박 사 학 위 논 문
Ph.D. Dissertation

심층학습을 위한 구조 독립적 불변성

Architecture-Agnostic Invariances for Deep Learning

2026

김 진 우 (金 眞 友 Kim, Jinwoo)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

심층학습을 위한 구조 독립적 불변성

2026

김 진 우

한 국 과 학 기 술 원

전산학부

심층학습을 위한 구조 독립적 불변성

김 진 우

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2025년 12월 11일

심사위원장	홍 승 훈	(인)
심 사 위 원	조 경 현	(인)
심 사 위 원	양 홍 석	(인)
심 사 위 원	안 성 수	(인)
심 사 위 원	Siamak Ravanbakhsh	(인)

Architecture-Agnostic Invariances for Deep Learning

Jinwoo Kim

Advisor: Seunghoon Hong

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computing

Daejeon, Korea
December 11, 2025

Approved by

Seunghoon Hong
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DCS 김진우. 심층학습을 위한 구조 독립적 불변성. 전산학부 . 2026년. 214+x 쪽. 지도교수: 홍승훈. (영문 논문)
Jinwoo Kim. Architecture-Agnostic Invariances for Deep Learning. School of Computing . 2026. 214+x pages. Advisor: Seunghoon Hong. (Text in English)

초 록

본 논문은 심층학습 모델이 훈련 데이터를 벗어난 환경에서도 일반화 성능을 확보해야 하는 오랜 문제를 다룬다. 이를 위해 기반 모델을 포함한 임의의 구조를 가진 신경망에 임의의 대칭성에 대한 불변성과 등변성을 부여하여, 변환된 새로운 입력에 대해서도 일반화 성능을 향상시키는 실용적인 확률론적 방법론들을 제안한다. 제안된 방법들은 확장성이 높은 범용 신경망으로 하여금 시각 및 언어에서 그래프로의 전이 학습, 지식 그래프에서의 제로샷 추론, 별도의 훈련이 필요 없는 시각적 왜곡 하에서의 컴퓨터 비전, 그리고 새로운 초기 조건하에서의 편미분 방정식 풀이 등의 과제를 수행할 수 있도록 한다. 이러한 기여를 통해, 본 연구는 불변성에 대한 사전 지식이 가용한 다양한 도메인에서 심층학습의 적용 범위와 효용성을 한층 향상시킨다.

핵 심 낱 말 기하학적 심층학습, 불변성 및 등변성, 대칭성, 일반화

Abstract

A longstanding problem in deep learning is making models generalize better out of their training data. This thesis proposes practical probabilistic methods for endowing neural networks of any architecture such as foundation models with invariance and equivariance with respect to arbitrary symmetries to improve their generalizations on novel transformed inputs. These methods enable neural networks with scalable, all-purpose architectures to perform transfer learning from vision and language to graphs, zero-shot reasoning on knowledge graphs, training-free warp-invariant vision, and neural partial differential equations solving for unseen initial conditions. These contributions help advance the scope and utility of deep learning across diverse domains whenever relevant knowledge of invariance is available.

Keywords Geometric deep learning, invariance and equivariance, symmetry, generalization

Contents

List of Tables

List of Figures

Chapter 1. Introduction

A longstanding challenge in deep learning is building models that can learn from limited data and generalize robustly to previously unseen circumstances. While modern neural networks have achieved remarkable success across vision and language, they require an enormous amount of data and struggle when inputs are presented in unfamiliar ways. One principled approach to this challenge is to incorporate our knowledge of invariance, specifying how the semantics of data remain unchanged under structured transformations. A classifier that knows rotations preserve object identity can reliably generalize from upright examples to rotated ones without seeing them during training. This insight has driven a decade of research into methods exploiting symmetry groups of geometric transformations, with applications including robotics, physics simulation, drug discovery, and recommendation systems.

The dominant paradigm leverages symmetries through specialized architectures, such as group-convolutional and message-passing networks, whose weight-sharing is constrained to respect specific symmetries. While effective, this approach has significant limitations. Each new symmetry requires custom architecture design, often incurring substantial implementation complexity and restricted expressive power. More critically, these specialized structures are incompatible with general-purpose architectures like transformers, and thus with the valuable ability to transfer knowledge from data-abundant domains to problems with different symmetries. This motivates the central question of this thesis: *Can we harness the generalization benefits of symmetry while remaining fully compatible with arbitrary neural architectures?*

This thesis answers this question affirmatively by developing architecture-agnostic methods for endowing neural networks with invariance and equivariance to arbitrary symmetries. By decoupling symmetry from architectural design, these methods achieve robust generalization while leveraging the scalability and transferable knowledge of general-purpose models. We make three principal contributions. In Chapter 2, we show that standard transformers can express permutation-equivariant computations via appropriate tokenization and input augmentations, without modifying the attention mechanism. This yields graph and hypergraph transformers compatible with conventional implementations, achieving competitive performance on large-scale computational chemistry benchmarks. In Chapter 3, we introduce probabilistic symmetrization, an approach that achieves equivariance not through architectural constraints but through stochastic data transformations. Applicable to any base model and symmetry group, this framework enables zero-shot warp-invariance in vision, improved neural PDE solving for unseen initial conditions, and effective transfer of models pre-trained on natural images to graphs. In Chapter 4, we introduce random walk neural networks, which convert graphs into sequences via stochastic traversals, enabling any sequence model to process graph data with provable isomorphism invariance. This allows pre-trained language models to be directly applied to graph reasoning, classification on scientific and social networks, and protein property prediction. We further develop a knowledge graph foundation model capable of zero-shot link prediction for entirely unseen entities and relations.

Together, these contributions expand geometric deep learning beyond specialized architectures, enabling practitioners to incorporate symmetry into any model and opening new research directions at the intersection of symmetry and generalization.

Chapter 2. Higher-order Transformers

The transformer architecture has become the dominant backbone for deep learning, powering state-of-the-art models in vision and language. Its success stems largely from self-attention, which captures global dependencies between elements. However, standard transformers lack explicit inductive biases for structured data such as graphs and hypergraphs, where permutation symmetry should be respected. Learning on such structures has conventionally relied on message-passing networks. While satisfying permutation equivariance by construction, these architectures often struggle with long-range dependencies due to issues in signal propagation and are limited in expressivity by the Weisfeiler-Lehman hierarchy. This chapter addresses this problem by developing principled extensions of transformers to sets, graphs, and hypergraphs, and ultimately demonstrating that standard transformers, given appropriate input representations, can match or exceed the expressivity of specialized architectures.

This chapter is adapted from the following papers:

- "Transformers Generalize DeepSets and Can be Extended to Graphs and Hypergraphs", which originally appeared at NeurIPS 2021 and is joint work with Saeyoon Oh, and Seunghoon Hong,
- "Equivariant Hypergraph Neural Networks", which originally appeared at ECCV 2022 and is joint work with Saeyoon Oh, Sungjun Cho, and Seunghoon Hong,
- "Pure Transformers are Powerful Graph Learners", which originally appeared at NeurIPS 2022 and is joint work with Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong.

The first two works develop attention mechanisms over tensorized queries and keys with equivariant masking, extending transformers to graphs and hypergraphs while exceeding the expressivity of their message-passing counterparts. The third work shows that these mechanisms can be implicitly realized by standard transformers through appropriate tokenization. In Section 2.1, we develop higher-order transformers for graphs and hypergraphs, along with sparse and kernelized variants that scale linearly with edge count. In Section 2.2, we present equivariant hypergraph neural networks that support hyperedges of arbitrary size via symmetric tensor representations and hypernetwork-based parameter sharing. In Section 2.3, we introduce tokenized graph transformers, demonstrating that a standard transformer with node and edge tokens augmented by node identifiers can express higher-order attention, effectively converting a general-purpose architecture into a powerful graph learner without architectural modification.

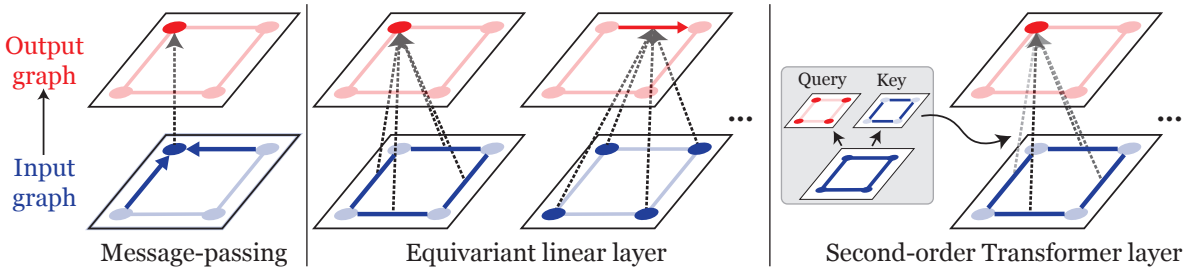


Figure 2.1: Illustrated operations of a message-passing GNN, an equivariant linear layer, and a second-order transformer layer. A single output node is highlighted.

2.1 Higher-order transformers

Graph is a universal data modality used to model social networks [Qiu et al., 2018], chemical compounds [Gilmer et al., 2017], biological structures [Fout et al., 2017], and interactions in particle physics [Kipf et al., 2018, Serviansky et al., 2020]. Graph neural networks (GNNs) adopt a message-passing scheme [Zhou et al., 2018, Zhang et al., 2018b, Wu et al., 2021b], where the node features are propagated and aggregated recurrently according to the neighborhood structure given in the input graph. Despite the simplicity, the local and recurrent nature makes them unable to discover dependency between any two nodes with a distance longer than the message-passing steps [Gu et al., 2020]. The locality of message passing is also known to be related to the over-smoothing problem that prevents scaling of GNNs [Li et al., 2018, Cai and Wang, 2020, Oono and Suzuki, 2020].

An alternative approach is using a more general set of operations that involve global interactions while respecting the permutation symmetry of graphs. Such operations either produce the same output regardless of the node ordering of input graph (permutation invariance), or commute with node reordering (permutation equivariance). Message passing is an equivariant operation, but is restricted to local neighborhoods defined by the input graph. Recently, Maron et al. [2019b] characterized the full space of invariant and equivariant linear layers. It turned out that these layers span not only the local neighborhood interactions of message-passing GNNs, but also global interactions such as the one between disconnected nodes, and even edge-to-node, or node-to-edge interactions (Figure 2.1). Notably, their formulation naturally extends to layers with different input & output orders (e.g., edge in, node out), and higher-order layers for hypergraphs. In theory, an invariant MLP composed of these layers should be more powerful than message-passing GNNs as they can model long-range dependency between nodes (Figure 2.1). However, they are currently not widely adopted due to relatively low performance and high memory cost.

Contributions We present higher-order transformers that address the low performance and high complexity of invariant MLPs. First, based on an observation that transformer encoder generalizes first-order equivariant linear layers or deep sets [Zaheer et al., 2017], we formulate higher-order transformer layers that generalize equivariant linear layers by extending self-attention to higher orders (Figure 2.1). Second, while transformer layers with input and output orders k, l has $\mathcal{O}(n^{k+l})$ complexity, we show that by leveraging the sparsity of inputs, we can obtain $\mathcal{O}(m^2)$ complexity given input with m hyperedges. Further adopting kernel attention approaches, we propose a variant that reduces the complexity to $\mathcal{O}(m)$ and theoretically prove that they are more expressive than message-passing networks. Finally, we test higher-order transformers on a range of tasks, and show that they achieve significant improvements over invariant MLPs and are highly competitive in performance and scalability to the state-of-the-art GNNs.

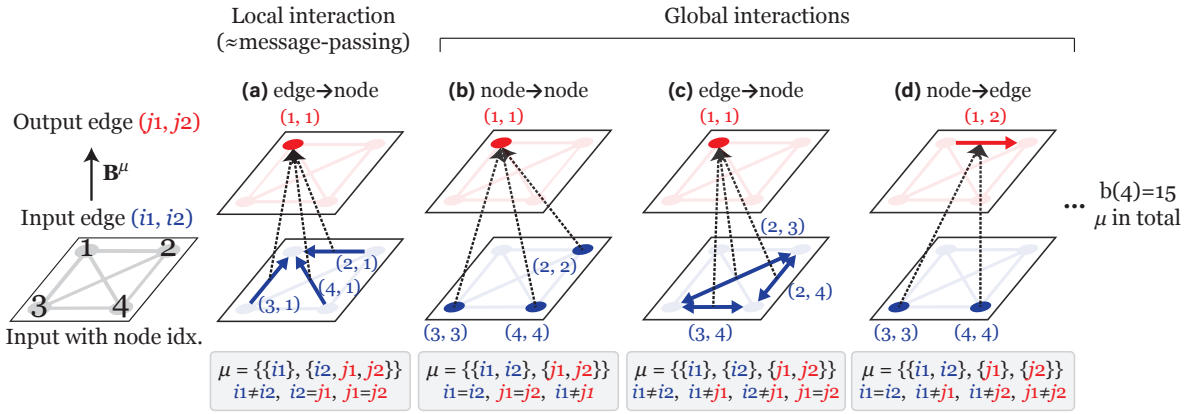


Figure 2.2: Example operations in a second-order layer $L_{2\to 2}$. We illustrate how input edges $(i_1, i_2) = \mathbf{i}$ are aggregated to an output edge $(j_1, j_2) = \mathbf{j}$ in various equivalence classes $(\mathbf{i}, \mathbf{j}) \in \mu$ via basis tensor $\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}$. Note that a loop ($(i_1, i_1)$ or (j_1, j_1)) represents a node.

2.1.1 Preliminary

In this section, we first describe how (hyper)graphs can be treated as higher-order tensors. We then describe linear layers that operate on higher-order tensors while respecting node permutation symmetry. In particular, we analyze operations within linear layers for second-order tensors (graphs) and show they involve global interactions on top of local aggregation.

Let us define some notations. We denote a set as $\{a, ..., b\}$, a tuple as $(a, ..., b)$, and denote $[n] = \{1, ..., n\}$. We denote the space of order-k tensors as $\mathbb{R}^{n^k \times d}$ where d is feature dimension. For an order-k tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, we use multi-index $\mathbf{i} = (i_1, ..., i_k) \in [n]^k$ to denote $\mathbf{A}_{\mathbf{i}} = \mathbf{A}_{i_1,...,i_k} \in \mathbb{R}^d$. Let S_n be the set of all permutations of $[n]$. $\pi \in S_n$ acts on $\mathbf{i}$ by $\pi(\mathbf{i}) = (\pi(i_1), ..., \pi(i_k))$, and acts on an order-$k$ tensor $\mathbf{A}$ by $(\pi \cdot \mathbf{A})_{\mathbf{i}} = \mathbf{A}_{\pi^{-1}(\mathbf{i})}$.

(Hyper)graphs as tensors Generally, a (hyper)graph data G can be represented as a tuple $(V, \mathbf{A})$, where V is a set of n nodes and $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ encodes features attached to hyperedges. The type of the hypergraph is indicated by the order k of the tensor $\mathbf{A}$. First-order tensor is a set of features (e.g., point cloud, bag-of-words) where $\mathbf{A}_i$ is the feature of node i. Second-order tensor encodes edge features (e.g., adjacency) where $\mathbf{A}_{i_1,i_2}$ is the feature of edge (i_1, i_2). Generally, an order-k tensor encodes hyperedge features (e.g., mesh) where $\mathbf{A}_{i_1,...,i_k}$ is the feature of hyperedge $(i_1, ..., i_k)$.

Permutation invariance and equivariance Our problem of interest is building a functional relation $f(\mathbf{A}) \approx T$ between tensor $\mathbf{A}$ and target T. If T is a single output vector, we often require that f is *permutation invariant*, that it satisfies $f(\pi \cdot \mathbf{A}) = f(\mathbf{A})$; if T is a tensor $T = \mathbf{T}$, we often require that f is *permutation equivariant*, satisfying $f(\pi \cdot \mathbf{A}) = \pi \cdot f(\mathbf{A})$, for all $\pi \in S_n$ and $\mathbf{A} \in \mathbb{R}^{n^k \times d}$. In a typical design setup where a neural network f is built using linear layers and non-linear activations, the construction of f reduces to finding invariant and equivariant *linear* layers.

Invariant and equivariant linear layers We summarize invariant linear layers $L_{k\to 0} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{d'}$ and equivariant linear layers $L_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ identified in Maron et al. [2019b]. Note that invariant layer is a special case of $L_{k\to l}$ with $l = 0$. In summary, given an input $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, the order-$l$

output of an equivariant layer $L_{k\to l}$ can be written with $\mathbf{i} \in [n]^k, \mathbf{j} \in [n]^l$ as:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{j}} b_{\lambda}, \tag{2.1}$$

where $w_\mu \in \mathbb{R}^{d\times d'}$, $b_\lambda \in \mathbb{R}^{d'}$ are weight and bias parameters, $\mathbf{B}^\mu \in \mathbb{R}^{n^{k+l}}$ and $\mathbf{C}^\lambda \in \mathbb{R}^{n^l}$ are *basis tensors* (will be defined), and μ and λ are *equivalence classes* of order-$(k+l)$ and order-l multi-indices, respectively. The equivalence classes are defined upon equivalence relation $\sim$ that, for multi-indices $\mathbf{i}, \mathbf{j} \in [n]^k$, $\mathbf{i} \sim \mathbf{j}$ iff $(i_1, ..., i_k) = (\pi(j_1), ..., \pi(j_k))$ for some node permutation $\pi \in S_n$. Then, a multi-index $\mathbf{i}$ and all members $\mathbf{j}$ in its equivalence class have the same (permutation-invariant) equality pattern: $\mathbf{i}_a = \mathbf{i}_b \Leftrightarrow \mathbf{j}_a = \mathbf{j}_b$ for all $a, b \in [k]$. Consequently, each equivalence class μ (or λ) is a distinct set of all order-$(k+l)$ (or order-l) multi-indices having a specific equality pattern.

Notably, we can represent each equivalence class of order-k multi-indices as a unique partition of $[k]$ regardless of n, where the partition specifies the equality pattern. *e.g.,* with $k = 2$, we have two partitions and respective equivalent classes: $\mu_1 = \{\{i_1, i_2\}\}$ the set of all (i_1, i_2) with $i_1 = i_2$, and $\mu_2 = \{\{i_1\}, \{i_2\}\}$ the set of all (i_1, i_2) with $i_1 \neq i_2$. Thus, with $\mathrm{b}(k)$ the k-th Bell number or number of partitions of $[k]$, we have $\mathrm{b}(k+l)$ equivalence classes μ for the weight and $\mathrm{b}(l)$ equivalence classes λ for the bias. We guide the readers interested in derivation to Maron et al. [2019b].

With the equivalence classes, basis tensors are defined as follows:

$$\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} = \begin{cases} 1 & (\mathbf{i},\mathbf{j}) \in \mu \\ 0 & \text{otherwise} \end{cases}; \qquad \mathbf{C}^{\lambda}_{\mathbf{j}} = \begin{cases} 1 & \mathbf{j} \in \lambda \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

In Equation (2.1), each equivalence class μ determines which (hyper)edges participate and how they interact in summation $\sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}}$. As an example, let us consider $L_{2\to 2}$ that maps input edges $\mathbf{i} = (i_1, i_2)$ to output edges $\mathbf{j} = (j_1, j_2)$. An equivalence class $\mu = \{\{i_1, i_2\}, \{j_1, j_2\}\}$ represents all $(\mathbf{i}, \mathbf{j}) \in \mu$ that $i_1 = i_2$, $j_1 = j_2$, and $i_1 \neq j_1$. Then, due to masking by $\mathbf{B}^\mu$, only input elements $\mathbf{A}_{i_1,i_2}$ with $i_1 = i_2$ participate in the summation and gives output $L_{2\to 2}(\mathbf{A})_{j_1,j_2}$ for $j_1 = j_2$, $i_1 \neq j_1$. Intuitively, this is analogous to computing a node feature by sum-pooling all the other nodes (Figure 2.2(b)). Different μ accounts for other interactions as shown in Figure 2.2. Notably, it contains a richer set of operations beyond local interactions modeled by message passing (Figure 2.2(a)), such as global interaction across all nodes (Figure 2.2(b)), and edge-to-node (Figure 2.2(c)), node-to-edge interactions (Figure 2.2(d)).

We finish the section by writing out the first-order equivariant layer $L_{1\to 1}$. As $\mathrm{b}(2) = 2$, the layer has two equivalence classes $\mu_1 = \{\{i_1, j_1\}\}$ and $\mu_2 = \{\{i_1\}, \{j_1\}\}$. Then, we have $\mathbf{B}^{\mu_1} = I_n$ and $\mathbf{B}^{\mu_2} = 1_n 1_n^\top - I_n$, with $1_n \in \mathbb{R}^n$ vector of ones. Then, given a set of features $\mathbf{A} \in \mathbb{R}^{n\times d}$,

$$L_{1\to 1}(\mathbf{A}) = I_n \mathbf{A} w_1' + (1_n 1_n^\top - I_n)\mathbf{A} w_2' + 1_n b^\top \tag{2.3}$$

$$= I_n \mathbf{A} w_1 + 1_n 1_n^\top \mathbf{A} w_2 + 1_n b^\top, \tag{2.4}$$

where $w_1, w_2, w_1', w_2' \in \mathbb{R}^{d\times d'}$, $b \in \mathbb{R}^{d'}$. $L_{1\to 1}$ is analogous to a combination of elementwise feedforward (μ_1) and sum-pooling of set elements (μ_2), and is also known as a deep set layer [Zaheer et al., 2017].

2.1.2 Method

In Section 2.1.1, we introduced higher-order linear equivariant layers $L_{k\to l}$, and showed that they contain various global and node/edge interactions that are not covered by message passing. In this section,

we establish a connection between the first-order equivariant layer $L_{1\to1}$ and self-attention of transformer encoder layers [Vaswani et al., 2017]. Then, we extend the relationship to higher orders by tensorizing queries and keys, and formulate higher-order transformer layers. We finish the section by proposing a principled parameter reduction for queries and keys, which reduces a fair amount of computation. All missing proofs can be found in Appendix A.1.1.

Transformers generalize deep sets

As we've shown, first-order linear layer, or deep set, has a simple structure composed of feedforward and sum-pooling (Equation (2.4)). Although it is theoretically proven to be a universal approximator of permutation-invariant functions [Zaheer et al., 2017], static sum-pooling could be limited in capturing interactions of set elements, motivating the use of sophisticated pooling. In particular, the self-attention mechanism of transformer encoder [Vaswani et al., 2017] was shown to achieve a large performance gain in set modeling via context-aware weighted pooling [Lee et al., 2019, Yun et al., 2020]. To see this, let us write out the transformer encoder layers.

A transformer encoder layer is a function $\mathrm{Enc} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ consisting of two layers: a self-attention layer $\mathrm{Attn} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ and an elementwise feedforward layer $\mathrm{MLP} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$. For a set of n input vectors $X \in \mathbb{R}^{n\times d}$, a transformer layer computes the following:

$$\mathrm{Enc}(X)_i = \mathrm{Attn}(X)_i + \mathrm{MLP}(\mathrm{Attn}(X))_i, \qquad \mathrm{Attn}(X)_i = X_i + \sum_{h=1}^{H}\sum_{j=1}^{n} \alpha_{ij}^h X_j w_h^V w_h^O,$$

where $\alpha^h = \sigma(X w_h^Q (X w_h^K)^\top)$ is an attention coefficient with an activation σ, H is the number of heads, d_H is head size, d_F is hidden dimension, and $w_h^O \in \mathbb{R}^{d_H\times d}$, $w_h^V, w_h^K, w_h^Q \in \mathbb{R}^{d\times d_H}$.[1]

Now, we show that transformer layers are generalized first-order linear equivariant functions. By setting $\alpha_{ij}^h = 1$ and assuming that $\mathrm{MLP}(Y)$ approximates a linear layer $Y w^F + b^{F\top}$ following the universal approximation theorem [Hornik et al., 1989], the transformer layer reduces to the following[2]:

$$\mathrm{Enc}(X)_i = X_i(I_n + w^F) + \sum_{j=1}^{n} X_j w^{VO}(I_n + w^F) + b^{F\top},$$

where $w^{VO} = \sum_{h=1}^{H} w_h^V w_h^O$. This is equivalent to a deep set layer in Equation (2.4) with $w_1 = I_n + w^F$, $w_2 = w^{VO}(I_n + w^F)$, and $b = b^F$. In other words, we can convert deep sets to transformers by changing the static pooling to attention and replacing elementwise linear mapping with an MLP. We generalize this approach to higher-order input and output, which leads to the formulation of higher-order transformers for graphs and hypergraphs.

Higher-order transformer layers

We've seen that transformer layers are generalized first-order equivariant linear layers $L_{1\to1}$. Notably, the generalization procedure was equivalent to changing static pooling to attention and adding feedforward MLP. In this section, we generalize the approach to $L_{k\to l}$ with arbitrary orders k and l and formulate higher-order transformer layer $\mathrm{Enc}_{k\to l}$.

In general, we define higher-order transformer layer as a function $\mathrm{Enc}_{k\to l} : \mathbb{R}^{n^k\times d} \to \mathbb{R}^{n^l\times d}$ with two layers: a higher-order self-attention layer $\mathrm{Attn}_{k\to l} : \mathbb{R}^{n^k\times d} \to \mathbb{R}^{n^l\times d}$ and a feedforward layer

[1] We omit normalization after $\mathrm{Attn}(\cdot)$ and $\mathrm{MLP}(\cdot)$ for simplicity as in Yun et al. [2020], Hanin and Sellke [2017].
[2] In practice, transformer employs softmax in attention and deviates from Deepsets due to normalization.

$\mathrm{MLP}_{l\to l} : \mathbb{R}^{n^l\times d} \to \mathbb{R}^{n^l\times d}$. For an input tensor $\mathbf{A} \in \mathbb{R}^{n^k\times d}$, a transformer layer computes:

$$\begin{aligned}\mathrm{Enc}_{k\to l}(\mathbf{A}) &= \mathrm{Attn}_{k\to l}(\mathbf{A}) + \mathrm{MLP}_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A})),\\ \mathrm{MLP}_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A})) &= L^2_{l\to l}\left(\mathrm{ReLU}(L^1_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A})))\right),\end{aligned} \tag{2.5}$$

where $L^1_{l\to l} : \mathbb{R}^{n^l\times d} \to \mathbb{R}^{n^l\times d_F}$ and $L^2_{l\to l} : \mathbb{R}^{n^l\times d_F} \to \mathbb{R}^{n^l\times d}$ are equivariant linear layers with hidden dimension d_F. Remaining question is how to define and compute higher-order self-attention $\mathrm{Attn}_{k\to l}(\mathbf{A})$.

To design $\mathrm{Attn}_{k\to l}$, we remove the bias from Equation (2.1) and introduce attention coefficients. This is done by changing each $\mathbf{B}^\mu \in \mathbb{R}^{n^{k+l}}$ to attention coefficient tensor $\boldsymbol{\alpha}^{h,\mu} \in \mathbb{R}^{n^{k+l}}$ with multiple heads:

$$\mathrm{Attn}_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{h=1}^{H}\sum_{\mu}\sum_{\mathbf{i}} \boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}}\mathbf{A}_{\mathbf{i}} w^V_{h,\mu} w^O_{h,\mu}, \tag{2.6}$$

where $w^O_{h,\mu} \in \mathbb{R}^{d_H\times d}$, $w^V_{h,\mu} \in \mathbb{R}^{d\times d_H}$ are learnable parameters, H denotes the number of heads, and d_H denotes head size. Then similar to first-order case, we can show the following:

Theorem 1. *$\mathrm{Enc}_{k\to l}$ (Equation (2.5)) is a generalization of $L_{k\to l}$ (Equation (2.1)).*

Proof. Let $\boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} = 1$ for all h, μ, and $(\mathbf{i},\mathbf{j}) \in \mu$. This renders $\boldsymbol{\alpha}^{h,\mu} = \mathbf{B}^\mu$ from the definition of $\mathbf{B}^\mu$. Additionally, let $\mathrm{MLP}_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A}))_{\mathbf{j}} = \sum_\lambda \mathbf{C}^\lambda_{\mathbf{j}} b_\lambda$ for all $\mathbf{j} \in [n]^l$. That is, $\mathrm{MLP}_{l\to l}$ ignores input and reduces to an invariant bias in Equation (2.1). Then, Equation (2.5) reduces to the following:

$$\mathrm{Enc}_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu}\sum_{\mathbf{i}} \mathbf{B}^\mu_{\mathbf{i},\mathbf{j}}\mathbf{A}_{\mathbf{i}} \sum_{h=1}^{H} w^V_{h,\mu} w^O_{h,\mu} + \sum_{\lambda} \mathbf{C}^\lambda_{\mathbf{j}} b_\lambda,$$

which is equivalent to Equation (2.1) with $w_\mu = \sum_{h=1}^{H} w^V_{h,\mu} w^O_{h,\mu}$. □

Now, we describe how to compute each attention tensor $\boldsymbol{\alpha}^\mu \in \mathbb{R}^{n^{k+l}}$ from input $\mathbf{A} \in \mathbb{R}^{n^k\times d}$ (Equation (2.6); head index h dropped for brevity). We obtain them from higher-order query and key:

$$\boldsymbol{\alpha}^\mu_{\mathbf{i},\mathbf{j}} = \begin{cases} \sigma(\mathbf{Q}^\mu_{\mathbf{j}}, \mathbf{K}^\mu_{\mathbf{i}})/Z_{\mathbf{j}} & (\mathbf{i},\mathbf{j}) \in \mu \\ 0 & \text{otherwise}\end{cases} \qquad \text{where} \qquad \mathbf{Q}^\mu = L^\mu_{k\to l}(\mathbf{A}), \mathbf{K}^\mu = L^\mu_{k\to k}(\mathbf{A}), \tag{2.7}$$

where $Z_{\mathbf{j}} = \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \sigma(\mathbf{Q}^\mu_{\mathbf{j}}, \mathbf{K}^\mu_{\mathbf{i}})$ is a normalization constant. We note that query and key tensors are computed from the input $\mathbf{A}$ using the equivariant linear layers in Equation (2.1). This leads to permutation equivariance (or invariance) of transformer encoder layer $\mathrm{Enc}_{k\to l}$ in Equation (2.5).

Reducing orders of query and key Although Equations (2.6) and (2.7) provide a simple and generic definition of higher-order self-attention, we observe that there exist a lot of unnecessary computations. Specifically, there exist elements of query $\mathbf{Q}^\mu$ and key $\mathbf{K}^\mu$ that are unused in computation of attention coefficient $\boldsymbol{\alpha}^\mu_{\mathbf{i},\mathbf{j}}$, as it depends only on indices satisfying $(\mathbf{i},\mathbf{j}) \in \mu$. In fact, it turns out that the effective orders of query and key are much smaller than l and k, as we show below:

Proposition 1. *From Equation (2.7), let $u(\cdot)$ denote the number of unique entries in a multi-index. With $u_q = u(\mathbf{j})$, $u_k = u(\mathbf{i})$ for some $(\mathbf{i},\mathbf{j}) \in \mu$, we can always find suitable linear layers $L^\mu_{k\to u_q}$, $L^\mu_{k\to u_k}$ and index space mappings $f^\mu_q : [n]^l \to [n]^{u_q}$, $f^\mu_k : [n]^k \to [n]^{u_k}$ that satisfy the following.*

$$\boldsymbol{\alpha}^\mu_{\mathbf{i},\mathbf{j}} = \sigma(\tilde{\mathbf{Q}}^\mu_{\mathbf{j}'}, \tilde{\mathbf{K}}^\mu_{\mathbf{i}'})/\tilde{Z}_{\mathbf{j}} \;\forall (\mathbf{i},\mathbf{j}) \in \mu, \tag{2.8}$$

where $\tilde{Z}_{\mathbf{j}} = \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \sigma(\tilde{\mathbf{Q}}^{\mu}_{\mathbf{j}'}, \tilde{\mathbf{K}}^{\mu}_{\mathbf{i}'})$, $\tilde{\mathbf{Q}}^{\mu} = L^{\mu}_{k\to u_q}(\mathbf{A})$, $\tilde{\mathbf{K}}^{\mu} = L^{\mu}_{k\to u_k}(\mathbf{A})$, $\mathbf{j}' = f^{\mu}_q(\mathbf{j})$, $\mathbf{i}' = f^{\mu}_k(\mathbf{i})$.

Based on Proposition 1, we can compute query and key in Equation (2.7) in a much compact way using linear layers with output orders u_q and u_k instead of l and k. In our experiments, we observe that this optimization is very useful in reducing the number of parameters and memory footprint to a feasible level without affecting the effective model capacity.

Efficient higher-order transformers

In Section 2.1.2, we formulated higher-order transformer layers $\text{Enc}_{k\to l}$ and showed that they generalize linear equivariant layers $L_{k\to l}$. However, this capability comes with a cost; the high asymptotic complexity of the transformer encoder limits its practical merits. Specifically, we show the following:

Property 1. *Given input size n, the asymptotic complexity of a linear layer $L_{k\to l}$ (Equation (2.1)) is $\mathcal{O}(n^{k+l})$, and complexity of an encoder layer $Enc_{k\to l}$ (Equation (2.5)) is $\mathcal{O}(n^{k+l} + n^{2k} + n^{2l})$.*

Thus, in this section, we further analyze the encoder layer and propose a number of optimization and relaxation to reduce the asymptotic complexity with a minimal impact on capability. Notably, combining all our strategies reduces the complexity to $\mathcal{O}(m)$ given a hypergraph with m hyperedges. Even with such efficiency, we show that the reduced version of higher-order transformer is theoretically more expressive than message-passing neural networks.

Efficient linear layers A computation bottleneck within transformer encoder layer $\text{Enc}_{k\to l}$ is the higher-order linear layer, since it is used for key, query, and feedforward layer. By exploiting only a subset of basis, we show that we can implement lightweight version of linear layer with reduced asymptotic complexity.

Proposition 2. *Given a linear layer $L_{k\to l}$ in Equation (2.1), we can always find a nonempty subset $\mathcal{M}$ of equivalence classes such that computation of the following for all $\mathbf{j}$ has $\mathcal{O}(n^l)$ complexity.*

$$\bar{L}_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu\in\mathcal{M}} \sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{j}} b_{\lambda}, \tag{2.9}$$

We term the reduced linear layer $\bar{L}_{k\to l}$ in Equation (2.9) as a *lightweight* linear layer. In practice, we choose $\mathcal{M}$ among equivalence classes that do not involve summation over input. For instance, for $\bar{L}_{1\to 1}$, we use only the basis for elementwise mapping (I_n) and drop sum-pooling ($1_n 1_n^{\top}$) from Equation (2.4). This approximation effectively reduces the complexity of linear layers at the cost of losing inter-element dependencies. We employ the lightweight linear layers within the $\text{Enc}_{k\to l}$ (Equation (2.5)) to compute query and key embeddings, while the element dependency within $\text{Enc}_{k\to l}$ is handled by the higher-order self-attention using all equivalence classes as in Equation (2.7). This design is coherent to original (first-order) transformers, where the elements are first linearly projected to query/key with elementwise basis (I_n) and interaction is handled by attention (implicitly $1_n 1_n^{\top}$). Importantly, this does not hurt Theorem 1 as attention coefficients can still reduce to one and MLP can reduce to bias.

Proposition 3. *$Enc_{k\to l}$ (Equation (2.5)) with linear layers for key, query, and MLP changed to $\bar{L}$ still generalizes $L_{k\to l}$ (Equation (2.1)).*

Proof. This can be shown simply by plugging $\bar{L}$ into the proof of Theorem 1. We can still assume $\boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} = 1$ for all $(\mathbf{i},\mathbf{j}) \in \mu$ by setting $\bar{L}$ for key and query to output constants, and can reduce $\text{MLP}_{l\to l}$ composed

of $\bar{L}$ to an invariant bias as we subsample $\mu \in \mathcal{M}$ but keep all λ for the bias. Thus, Equation (2.5) can reduce to Equation (2.1) and Theorem 1 holds. □

By using lightweight linear layers in $\text{Enc}_{k\to l}$, we can significantly reduce the computational cost. The complexity of $L^{\mu}_{k\to u_k}$, $L^{\mu}_{k\to u_q}$, and $\text{MLP}_{l\to l}$ in Equation (2.8) and Equation (2.5) reduces to $\mathcal{O}(n^k)$, $\mathcal{O}(n^l)$, and $\mathcal{O}(n^l)$, respectively, and as a result $\text{Enc}_{k\to l}$ becomes $\mathcal{O}(n^{k+l})$. As a result, we have higher-order transformer layers $\text{Enc}_{k\to l}$ that generalize $L_{k\to l}$ while retaining the complexity $\mathcal{O}(n^{k+l})$. From here we assume that all linear layers for key, query, and MLP are lightweight.

Sparse transformer layers Even with lightweight linear layers, $\mathcal{O}(n^{k+l})$ complexity of $\text{Enc}_{k\to l}$ is still impractical. For example, for graphs, complexity larger than $\mathcal{O}(n^2)$ is regarded prohibitive while $\text{Enc}_{2\to 2}$ is $\mathcal{O}(n^4)$. Fortunately, leveraging the sparsity inherent in real-world graphs can significantly reduce the complexity. As we show, it reduces the complexity of $\text{Enc}_{k\to l}$ to $\mathcal{O}(m^2)$ for processing a hypergraph with m hyperedges.

Let E the set of hyperedges of input hypergraph G. Each hyperedge is generally represented by a multi-index $\mathbf{i} \in [n]^k$, so we denote $E = \{\mathbf{i}_1, ..., \mathbf{i}_m\}$ with m hyperedges. Leveraging sparsity of E is straightforward when the order of the hypergraph is fixed (e.g., $L_{k\to k}$); we can perform computations with only respect to the existing hyperedges $\mathbf{i}, \mathbf{j} \in E$. However, in our framework, order can change by layers (*e.g.*, $L_{k\to l}$), making it difficult to directly transfer the sparsity structure E to different-order output. Inspired by network projection [Carletti et al., 2020], we remedy this by constructing E' such that for any $\mathbf{j} \in E'$, there exists $\mathbf{i} \in E$ containing all unique elements of $\mathbf{j}$. For example, for $L_{3\to 2}$, this corresponds to projection of third-order E to second-order E' by obtaining edges from sides of triangles. Despite simplicity, this simple heuristic works well in general and generalizes to any order.

Then, we integrate the hyperedge sets E, E' into computation of linear layer in Equation (2.9) as:

$$\bar{L}_{k\to l}(\mathbf{A}, E)_{\mathbf{j}} = \begin{cases} \sum_{\mu\in\mathcal{M}} \sum_{\mathbf{i}\in E} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{j}} b_{\lambda} & \mathbf{j} \in E' \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

Likewise, integrating E, E' into attention computation in Equations (2.6) and (2.7) we have:

$$\text{Attn}_{k\to l}(\mathbf{A}, E)_{\mathbf{j}} = \begin{cases} \sum_{h=1}^{H} \sum_{\mu} \sum_{\mathbf{i}\in E} \boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w^{V}_{h,\mu} w^{O}_{h,\mu} & \mathbf{j} \in E' \\ 0 & \text{otherwise} \end{cases}$$

$$\text{where } \boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} = \begin{cases} \sigma(\mathbf{Q}^{h,\mu}_{\mathbf{j}}, \mathbf{K}^{h,\mu}_{\mathbf{i}})/Z_{\mathbf{j}} & (\mathbf{i},\mathbf{j}) \in \mu, \mathbf{i} \in E, \mathbf{j} \in E' \\ 0 & \text{otherwise} \end{cases}, \tag{2.11}$$

where $\mathbf{Q}^{h,\mu} = \bar{L}^{h,\mu}_{k\to l}(\mathbf{A}, E)$, $\mathbf{K}^{h,\mu} = \bar{L}^{h,\mu}_{k\to k}(\mathbf{A}, E)$, $Z_{\mathbf{j}} = \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu\wedge\mathbf{i}\in E} \sigma(\mathbf{Q}^{h,\mu}_{\mathbf{j}}, \mathbf{K}^{h,\mu}_{\mathbf{i}})$.

Property 2. *When given E with m elements, the equivariant linear layer in Equation* (2.10) *has $\mathcal{O}(m)$ complexity, and self-attention computation in Equation* (2.11) *has $\mathcal{O}(m^2)$ complexity. Consequently, the computation done by a $\text{Enc}_{k\to l}$ composed of the layers has $\mathcal{O}(m^2)$ complexity.*

Kernel attention trick We've seen that, by constraining linear layers within $\text{Enc}_{k\to l}$ to have sparse input and output, we can reduce $\mathcal{O}(n^{k+l})$ complexity to quadratic $\mathcal{O}(m^2)$ to input size. Yet, even $\mathcal{O}(m^2)$ can be demanding with large or dense input. As the quadratic term comes from self-attention computation, we follow the prior work in kernel attention [Katharopoulos et al., 2020, Choromanski et al., 2021] and view attention coefficients as pairwise dot-product scores. As we will show, this allows us to further

reduce the complexity of $\text{Enc}_{k\to l}$ to $\mathcal{O}(n^k + n^l)$ and even $\mathcal{O}(m)$ for sparse version, at the cost of relaxing some modeling assumption.

We begin by approximating attention coefficient in Equation (2.7) using pairwise dot-product kernel:

$$\boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}} = \begin{cases} \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top}\phi(\mathbf{K}^{\mu}_{\mathbf{i}})/Z_{\mathbf{j}} & (\mathbf{i},\mathbf{j}) \in \mu \\ 0 & \text{otherwise} \end{cases} \quad \text{where} \quad Z_{\mathbf{j}} = \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top}\phi(\mathbf{K}^{\mu}_{\mathbf{i}}), \tag{2.12}$$

where $\phi : \mathbb{R}^{d_H} \to \mathbb{R}^{d_K}_{+}$ is kernel feature map. The choice of kernel can be flexible, and in our implementation we adopt the Performer kernel [Choromanski et al., 2021] with strong theoretical and empirical guarantee.

Substituting Equation (2.12) in Equation (2.6), we have:

$$\begin{aligned} \text{Attn}_{k\to l}(\mathbf{A})_{\mathbf{j}} &= \sum_{\mu} Z^{-1}_{\mathbf{j}} \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top}\phi(\mathbf{K}^{\mu}_{\mathbf{i}})\mathbf{A}_{\mathbf{i}} w^{V}_{\mu} w^{O}_{\mu} = \sum_{\mu} Z^{-1}_{\mathbf{j}} \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top} \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \phi(\mathbf{K}^{\mu}_{\mathbf{i}})\mathbf{A}_{\mathbf{i}} w^{V}_{\mu} w^{O}_{\mu}, \\ &\text{where } Z_{\mathbf{j}} = \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top} \sum_{\mathbf{i}|(\mathbf{i},\mathbf{j})\in\mu} \phi(\mathbf{K}^{\mu}_{\mathbf{i}}). \end{aligned} \tag{2.13}$$

Here, the two inner-summations in Equation (2.13) are over the key index $\mathbf{i}$, which is *coupled* with the query index $\mathbf{j}$ by $(\mathbf{i},\mathbf{j}) \in \mu$. This coupling causes the major computational bottleneck, since the inner-summations should be computed for every $\mathbf{j}$-th query. We propose an approximation by decoupling the key and query indices and taking inner-summations over $\mathcal{I} = \bigcup_{\mathbf{j}} \{\mathbf{i}|(\mathbf{i},\mathbf{j}) \in \mu\}$:

$$\text{Attn}_{k\to l}(\mathbf{A})_{\mathbf{j}} \approx \sum_{\mu} Z^{-1}_{\mathbf{j}} \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top} \sum_{\mathbf{i}\in\mathcal{I}} \phi(\mathbf{K}^{\mu}_{\mathbf{i}})\mathbf{A}_{\mathbf{i}} w^{V}_{\mu} w^{O}_{\mu}, \text{ where } Z_{\mathbf{j}} \approx \phi(\mathbf{Q}^{\mu}_{\mathbf{j}})^{\top} \sum_{\mathbf{i}\in\mathcal{I}} \phi(\mathbf{K}^{\mu}_{\mathbf{i}}). \tag{2.14}$$

The approximation allows a query to attend to some additional keys (depending on μ), but it does not hurt the central requirement of attention that each query can assign different attention weights to the keys. With the approximation, we can compute the summations $\sum_{\mathbf{i}\in\mathcal{I}} \phi(\mathbf{K}^{\mu}_{\mathbf{i}})\mathbf{A}_{\mathbf{i}}$ and $\sum_{\mathbf{i}\in\mathcal{I}} \phi(\mathbf{K}^{\mu}_{\mathbf{i}})$ only once and reuse them across all query indices $\mathbf{j}$ to reduce the cost. Specifically, we show:

Property 3. *The encoder $\text{Enc}_{k\to l}$ with approximation in Equation (2.14) has a complexity of $\mathcal{O}(n^k + n^l)$. Exploiting sparsity further reduces the complexity to $\mathcal{O}(m)$, linear to the number of hyperedges m.*

Theoretical analysis

We showed that exploiting sparsity and adopting kernel attention reduces the computational complexity of the transformer encoder to linear to input edges. Considering graphs ($k = l = 2$), this complexity is equivalent to or better than $\mathcal{O}(n + m)$ complexity of the message passing operation[3]. Still, we show that our (approximate) encoder is theoretically more expressive than message passing.

Specifically, we show the following:

Theorem 2. *A composition of two sparse transformer layers $\text{Enc}_{2\to 2}$ with kernel attention can approximate any message-passing algorithm [Gilmer et al., 2017] to arbitrary precision, while the opposite is not true.*

This leads to the following corollary:

Corollary 1. *Second-order sparse transformers with kernel attention are more expressive than any message-passing neural networks within the framework of Gilmer et al. [2017].*

[3]In practice, we place node features on the diagonals of the adjacency matrix, leading to $\mathcal{O}(n + m)$.

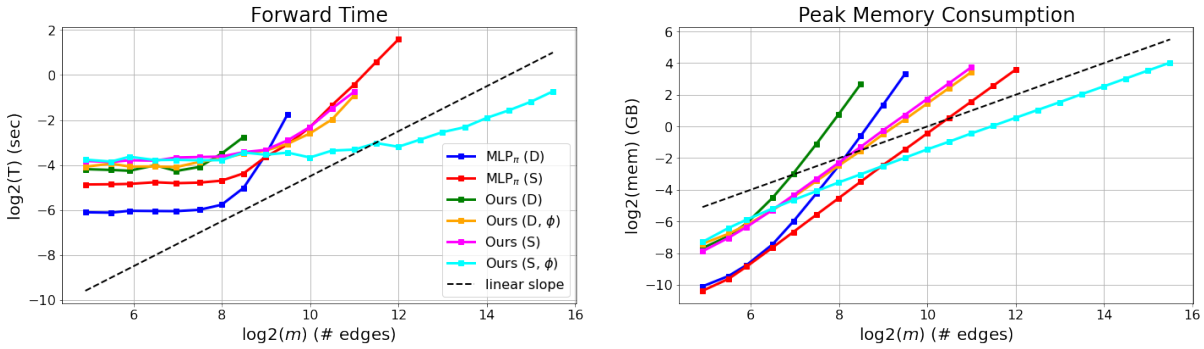


Figure 2.3: A comparison of all second-order models in terms of forward time, memory consumption, and maximal possible input size. Plots are shown until each model runs into out-of-memory error on a RTX 6000 GPU with 22GB.

In the proof, we note that local information propagation in message-passing GNNs is carried out in $\text{Enc}_{2\to 2}$ by a single $\mu = \{\{i_1\}, \{i_2, i_3, i_4\}\}$. Other types of μ would carry out different operations, which provides intuition on the powerfulness of transformers.

2.1.3 Experiments

We demonstrate the capability of higher-order transformers on a variety of tasks including synthetic data, large-scale graph regression, and set-to-(hyper)graph prediction. Specifically, we use a synthetic node classification dataset from Gu et al. [2020], a molecular graph regression dataset from Hu et al. [2021], two set-to-graph prediction datasets from Serviansky et al. [2020], and three hyperedge prediction datasets used in Zhang et al. [2020]. We implemented invariant MLP in Maron et al. [2019b] as one of baselines, which we abbreviate as MLP_π. We build our model by gradually adding lightweight linear layers, sparse linear layers, and kernel attention, and denote them by (D), (S), and (ϕ), respectively.

Runtime and memory cost

To experimentally verify our claims on linear complexity, we conducted a runtime and memory consumption analysis on all second-order models using random graphs. The results are shown in Figure 2.3. Consistent with our theoretical claims, sparse second-order transformer with kernel attention (Ours (S, ϕ)) is the only variant that linearly scales to input size in terms of both time and memory. Also, it is the only one that successfully scales to graphs with $>$ 50k edges, while still very fast in smaller graphs.

Synthetic chains

To test the ability of higher-order transformers in modeling long-range interactions in graphs, we used a synthetic dataset where the task is node classification in chain graphs. Binary class information is provided in a terminal node, so a model is required to propagate the information across the chain by handling long-range dependency. We used training and test sets with chains of length 20 and 200 respectively. As baselines, we used 3 message-passing networks, a sparse invariant MLP,

Table 2.1: Chain node classification results.

Method	Micro F1 (%) ↑	Macro F1 (%) ↑
GCN	47.78 ± 4.17	33.58 ± 1.86
GIN-0	53.72 ± 4.17	36.22 ± 1.86
GAT	47.78 ± 4.17	33.58 ± 1.86
MLP_π (S)	53.5 ± 4.16	36.04 ± 1.97
Ours (S) w/o global	53.72 ± 4.17	36.22 ± 1.86
Ours (S, ϕ) w/o global	50.77 ± 5.15	35.22 ± 2.17
Ours (S)	$\mathbf{100 \pm 0.00}$	$\mathbf{100 \pm 0.00}$
Ours (S, ϕ)	$\mathbf{100 \pm 0.00}$	$\mathbf{100 \pm 0.00}$

and ablated versions of sparse second-order transformers where μ for global pooling are removed (w/o global).

The results are in Table 2.1. We first note that second-order transformers successfully capture long-range dependency up to 200 nodes apart, while message-passing networks fail. Importantly, when the subset of basis that accounts for global pooling is eliminated, the performance of transformers drops similar to message-passing networks, showing their importance in modeling long-range interaction. Yet, simply having global basis is not enough, as seen in the failure of MLP_π. This indicates fine-grained interaction modeling via attention is essential even in this simple task.

Large-scale graph regression

To further evaluate higher-order transformers in large-scale setting, we used PCQM4M-LSC from Open Graph Benchmark [Hu et al., 2021], which is a graph-level regression dataset composed of 3.8M molecular graphs. As test data is unavailable, we report the Mean Absolute Error (MAE) on validation dataset. In addition to the baselines from the benchmark, we also report the performances of second-order invariant MLP and a vanilla (first-order) transformer[4] for comparison.

The results are in Table 2.2. Second-order transformer outperforms the message-passing GNNs (GCN, GIN) by a large margin, including the ones with a virtual node that can model long-range interactions (GCN-VN, GIN-VN). It suggests that higher-order attention is potentially better in handling long-range interactions on graphs than the current practice of augmenting GNNs with a virtual node. Furthermore, second-order transformer outperforms invariant MLP, indicating that replacing sum-pooling with attention is important for scale-up. Finally, second-order transformer significantly outperforms vanilla transformer with Laplacian graph embeddings. This is presumably because node embeddings are insufficient to utilize features associated with edges, while second-order transformers can naturally use all edge information. We also note that invariant MLP and vanilla transformer have worse complexity than the second-order transformer ($\mathcal{O}(m^2)$ and $\mathcal{O}(n^2)$, respectively), while the second-order transformer has $\mathcal{O}(m)$ complexity identical to GCN.

Table 2.2: PCQM4M-LSC large-scale graph regression results. * indicates results are obtained with a shorter schedule (10% of the full iterations).

Method	Val. MAE ↓
MLP [Hu et al., 2021]	0.2044
GCN [Hu et al., 2021]	0.1684
GIN [Hu et al., 2021]	0.1536
GCN-VN [Hu et al., 2021]	0.1510
GIN-VN [Hu et al., 2021]	0.1396
Transformer + LapPE*	0.2162
MLP_π (S)*	0.1464
Ours (S, ϕ; small)*	0.1376
Ours (S, ϕ)*	0.1294
Ours (S, ϕ)	**0.1263**

Set-to-graph prediction

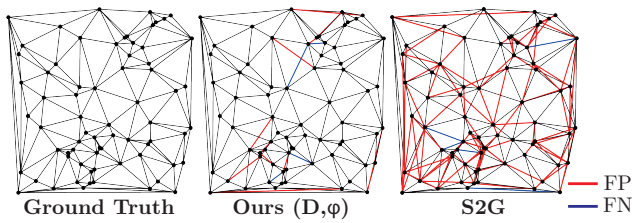
An important advantage of our framework distinguished from most existing GNNs is that, by design, it can be applied to any learning scenario with different input and output orders (mixed-order). To demonstrate this, we tested higher-order transformers in set-to-graph prediction tasks where the goal is to predict edge structure of a graph given a set of node features. We used two datasets following the prior work [Serviansky et al., 2020]. The first dataset Jets originates from particle physics experiments, where collision of high-energy particles gives a set of observed particles. The task is to partition the feature set of observed particles according to their origin. By viewing each subset of particles as a fully-connected graph, the problem is cast as a set-to-graph prediction. For the second dataset Delaunay, the task is to

[4]As vanilla transformer operates on node features only, we used Laplacian graph embeddings [Belkin and Niyogi, 2003, Dwivedi and Bresson, 2020] as positional embeddings so that the model can consider edge structure information.

Table 2.3: Set-to-graph results. Lower-right panel shows Delaunay (20-80) sample from ours and S2G.

	Method	F1 ↑	RI ↑	ARI ↑
Jets (B)	AVR	0.565	0.612	0.318
	MLP	0.533	0.643	0.315
	SIAM	0.606	0.675	0.411
	SIAM-3	0.597	0.673	0.396
	GNN	0.586	0.661	0.381
	S2G	0.646	0.736	0.491
	S2G+	0.655	0.747	0.508
	Ours (D)	0.667	0.746	0.520
	Ours (D, ϕ)	**0.670**	**0.751**	**0.526**
Jets (C)	AVR	0.695	0.650	0.326
	MLP	0.686	0.658	0.319
	SIAM	0.729	0.695	0.406
	SIAM-3	0.719	0.710	0.421
	GNN	0.720	0.689	0.390
	S2G	0.747	0.727	0.457
	S2G+	0.751	0.733	0.467
	Ours (D)	0.755	0.732	0.469
	Ours (D, ϕ)	**0.757**	**0.735**	**0.473**
Jets (L)	AVR	0.970	0.965	0.922
	MLP	0.960	0.957	0.894
	SIAM	0.973	0.970	0.925
	SIAM-3	0.895	0.876	0.729
	GNN	0.972	0.970	0.929
	S2G	0.972	0.970	0.931
	S2G+	0.971	0.969	0.929
	Ours (D)	**0.974**	**0.972**	**0.935**
	Ours (D, ϕ)	**0.974**	**0.972**	**0.935**

	Method	Acc ↑	Prec ↑	Rec ↑	F1 ↑
Delaunay (50)	SIAM	0.939	0.766	0.653	0.704
	SIAM-3	0.911	0.608	0.538	0.570
	GNN0	0.826	0.384	0.966	0.549
	GNN5	0.809	0.363	**0.985**	0.530
	GNN10	0.759	0.311	0.978	0.471
	S2G	0.984	0.927	0.926	0.926
	S2G+	0.983	0.927	0.925	0.926
	Ours (D)	**0.994**	**0.981**	0.967	**0.974**
	Ours (D, ϕ)	0.991	0.967	0.952	0.959
Delaunay (20-80)	SIAM	0.919	0.667	0.764	0.687
	SIAM-3	0.895	0.578	0.622	0.587
	GNN0	0.810	0.387	0.946	0.536
	GNN5	0.777	0.352	**0.975**	0.506
	GNN10	0.746	0.322	0.970	0.474
	S2G	0.947	0.736	0.934	0.799
	S2G+	0.947	0.735	0.934	0.798
	Ours (D)	**0.993**	**0.982**	0.960	**0.971**
	Ours (D, ϕ)	0.989	0.948	0.956	0.952



predict Delaunay triangulation [Delaunay, 1934] given a set of points in 2D space. Two datasets are used for this task, one containing 50 points and the other containing varying number of points $\in \{20, ..., 80\}$. For the baselines, we take the scores reported in Serviansky et al. [2020], which includes GNNs and invariant MLPs (S2G, S2G+) with $L_{1\to1}$ and $L_{1\to2}$. Our model is made by substituting the linear layers with $\text{Enc}_{1\to1}$ and $\text{Enc}_{1\to2}$ (mixed-order) respectively.

The results are outlined in Table 2.3. Mixed-order transformers, both softmax and kernel attention, have favorable scores over all baselines. Especially, they outperform all baselines by a large margin in Delaunay: note that GNNs fall into trivial solution with high recall but very low precision. We particularly note that the transformers' performance in Delaunay (20-80) is comparable to Delaunay (50), with 0.3-0.7% drop in F1 score. Compared with S2G that exhibits $\sim 12\%$ drop, this indicates attention mechanism within $\text{Enc}_{1\to2}$ is helpful in modeling varying number of nodes.

k-uniform hyperedge prediction

One major advantage of our framework is that it naturally extends to higher-order data (hypergraphs). To demonstrate this, we consider higher-order extension of set-to-graph prediction task, where the goal is predicting k-uniform hyperedges (e.g., user-location-activity) from node features. For evaluation, we used three datasets for transductive 3-edge prediction following the prior work [Zhang et al., 2020]. The first dataset GPS derives from a GPS network [Zheng et al., 2010], and contains (user-location-activity) hyperedges. The second dataset MovieLens is a social network dataset of tagging activities [Harper and Konstan, 2016], containing (user-movie-tag) hyperedges. The third dataset Drug comes from a medicine

Table 2.4: k-uniform hyperedge prediction results. For Hyper-SAGNN, we reproduced the scores using the open-sourced code. For the other baselines, we take the scores reported in Zhang et al. [2020].

	GPS		MovieLens		Drug	
	AUC ↑	AUPR ↑	AUC ↑	AUPR ↑	AUC ↑	AUPR ↑
node2vec-mean	0.563	0.191	0.562	0.197	0.670	0.246
node2vec-min	0.570	0.185	0.539	0.186	0.684	0.258
DHNE	0.910	0.668	0.877	0.668	0.925	0.859
Hyper-SAGNN-E	0.947	0.788	0.922	**0.792**	0.963	0.897
Hyper-SAGNN-W	0.907	0.632	0.909	0.683	0.956	0.890
S2G+ (S)	0.943	0.726	0.918	0.737	0.963	0.898
Ours (S, ϕ)	**0.952**	**0.804**	**0.923**	0.771	**0.964**	**0.901**

network from FAERS[5], containing (user-drug-reaction) hyperedges. As baselines, we consider higher-order invariant MLP (S2G+) and the state-of-the-art, self-attention based Hyper-SAGNN [Zhang et al., 2020]. We implemented higher-order transformer and S2G+ by substituting $\mathrm{Enc}_{1\to 2}$ and $L_{1\to 2}$ in set-to-graph architectures to $\mathrm{Enc}_{1\to 3}$ and $L_{1\to 3}$, respectively.

The results are in Table 2.4. Higher-order transformer outperforms S2G+ in all datasets, and Hyper-SAGNN in the most cases. The results suggest that higher-order self-attention is effective in learning higher-order representation beyond second-order graphs. The results are encouraging especially because we did not introduce any form of task-specific heuristics, while some of the baselines such as Hyper-SAGNN depend on many inductive biases (static/dynamic branches, Hadamard power, etc.).

2.1.4 Discussion

We proposed a generalization of transformers to higher-orders, and applied a number of design strategies that reduce their complexity to a feasible level. Higher-order transformers are attractive, both in theory and application. In theoretical aspect, it inherits the theoretical completeness and expressive power of invariant MLPs. In application aspect, it is potentially more powerful than message-passing neural networks due to global interaction modeling, and can be extended to a variety of useful mixed-order tasks involving sets, graphs, and hypergraphs.

At the same time, our work has some limitations that need to be addressed in future work. First, although complexity to input size can be lowered to linear, the number of basis grows rapidly with increasing order due to $\mathcal{O}((0.792k/\ln(k+1))^k)$ asymptotic formula of k-th Bell number [Berend and Tassa, 2010], still making the model infeasible in higher orders. Improvement approaches such as finding a compact subset of basis that retains universality [Serviansky et al., 2020] and exploiting unorderedness [Maron et al., 2019b] are promising in this direction. Second, our work builds upon tensor-based representation of graphs, which makes it difficult to be directly extended to hypergraphs containing edges with varying orders (e.g., co-citation networks). This is a common challenge to all tensor-based methods [Hartford et al., 2018, Maron et al., 2019b,a, Keriven and Peyré, 2019, Serviansky et al., 2020], and we believe addressing this would be an important future research direction.

[5] www.fda.gov/Drugs/

2.2 Equivariant hypergraph neural networks

Reasoning about a system that involves a set of entities and their relationships requires relational data structures. Graph represents relational data with nodes and edges, where a node corresponds to an entity and an edge represents a relationship between a pair of nodes. Pairs are often insufficient to represent complex relationships. For instance, geometric configurations of entities such as angles and areas can only be captured via higher-order relationships between three or more nodes. Hypergraph is a general data structure that represents such relationships with hyperedges, i.e., edges associating more than two nodes at a time [Berge and Minieka., 1981]. Thus, it is widely used to represent visual data such as scenes [Gao et al., 2012, Kim et al., 2020a], feature correspondence [Wang et al., 2021, Belongie et al., 2002, Ray, 2005, Lowe, 1999] and polygonal mesh [Botsch et al., 2008, Milano et al., 2020, Yavartanoo et al., 2021], as well as general relational data such as social and networks [Tan et al., 2011, Li et al., 2013, Bu et al., 2010], biological networks [Gu et al., 2017, Klimm et al., 2021], linguistic structures [Ding et al., 2020] and combinatorial optimization problems [Kalai, 1997].

To learn deep representation of hypergraphs, recent works developed specialized hypergraph neural networks by generalizing message-passing GNNs [Feng et al., 2019, Bai et al., 2021, Dong et al., 2020, Huang and Yang, 2021, Arya et al., 2020, Chien et al., 2022]. In these networks, node and (hyper)edge features are updated recurrently by aggregating features of neighbors according to the connectivity in the input. Despite such simplicity, message passing has limitations. Notably, their local and recurrent characteristics prevent them from handling dependencies between any nodes with a distance larger than the propagation steps. This is known to be related to oversmoothing which hinders the use of deep networks [Li et al., 2018, Cai and Wang, 2020, Oono and Suzuki, 2020, Huang and Yang, 2021].

A more general and potentially powerful approach is to find *all* possible permutation-equivariant linear operations on the input (hyper)graph and use them to form linear layers of *equivariant GNNs* [Maron et al., 2019b]. While message passing is a specific, locally restricted case of equivariant operation, the maximal set extends further to various global interactions over possibly disconnected nodes and (hyper)edges [Kim et al., 2021a]. This naturally extends to higher-order layers that can handle hypergraphs in principle and even mixed-order layers where input and output are of different orders (e.g., graph in, hypergraph out). Yet, the actual usage of equivariant GNNs has been mainly limited to sets and graphs [Zaheer et al., 2017, Maron et al., 2019b, Serviansky et al., 2020, Kim et al., 2021a], and have not been realized for general hypergraph learning. This is due to the prohibitive parameter counts of higher-order layers and a bound in the input and output hyperedge orders that comes from fixed-order tensor representation.

We propose equivariant hypergraph neural network (EHNN) as the first attempt to realize equivariant GNNs for hypergraphs. We establish a connection between sparse, general hypergraphs and dense, fixed-order tensors, from which we derive the maximally expressive equivariant linear layers for undirected hypergraphs. Then we impose an intra-layer parameter sharing via hypernetworks [Ha et al., 2017], which retains expressiveness, bounds the parameter count, and allows processing hyperedges of arbitrary and possibly unseen orders. The resulting layer (EHNN-MLP) turns out to be a simple augmentation of an MLP-based message passing with hyperedge order embedding and global pooling, allowing an efficient implementation. We further develop a transformer counterpart (EHNN-transformer) by introducing self-attention to achieve a higher expressivity with the same asymptotic cost. In a challenging synthetic k-edge identification task, we show that EHNN performs fine-grained global reasoning to perfectly solve the task, and generalizes towards unseen hyperedge orders. We demonstrate state-of-the-art results in several hypergraph learning tasks, including semi-supervised classification and visual keypoint matching.

2.2.1 Preliminary

We introduce some preliminary of permutation equivariant learning [Maron et al., 2019b, Serviansky et al., 2020, Kim et al., 2021a]. We first describe higher-order tensors and then maximally expressive permutation equivariant linear layers that compose equivariant GNNs [Maron et al., 2019b].

We denote a set as $\{a, ..., b\}$, a tuple as $(a, ..., b)$, and $[n] = \{1, ..., n\}$. We denote the space of order-k tensors with d features as $\mathbb{R}^{n^k \times d}$. For $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, we use a multi-index $\mathbf{i} = (i_1, ..., i_k) \in [n]^k$ to index an element $\mathbf{A}_\mathbf{i} = \mathbf{A}_{i_1,...,i_k} \in \mathbb{R}^d$. Let S_n be all permutations of $[n]$. A node permutation $\pi \in S_n$ acts on a multi-index $\mathbf{i}$ by $\pi(\mathbf{i}) = (\pi(i_1), ..., \pi(i_k))$ and on a tensor $\mathbf{A}$ by $(\pi \cdot \mathbf{A})_\mathbf{i} = \mathbf{A}_{\pi^{-1}(\mathbf{i})}$.

Higher-order tensors Prior work on equivariant learning regard hypergraph data as $G = (V, \mathbf{A})$, with V a set of n nodes and $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ a tensor encoding hyperedge features. The order k of the tensor $\mathbf{A}$ indicates the type of hypergraph. First-order tensor encodes a set of features (e.g., point cloud) where $\mathbf{A}_i$ is the feature of node i. Second-order tensor encodes pairwise edge features (e.g., adjacency) where $\mathbf{A}_{i_1,i_2}$ is the feature of edge (i_1, i_2). Generally, an order-k tensor encodes *hyperedge* features (e.g., mesh normal) where $\mathbf{A}_{i_1,...,i_k}$ is the feature of hyperedge $(i_1, ..., i_k)$. We begin our discussion from the tensors, but will arrive at the familiar notion of hypergraphs with any-order undirected hyperedges.

Permutation invariance and equivariance In (hyper)graph learning, we are interested in building a function f that takes a (higher-order) tensor $\mathbf{A}$ as input and outputs some value. Since the tensor representation of a graph changes dramatically with the permutation of node numbering, the function f should be invariant or equivariant under node permutations. Formally, if the output is a single vector, f is required to be *invariant*, always satisfying $f(\pi \cdot \mathbf{A}) = f(\mathbf{A})$; if the output is a tensor, f is required to be *equivariant*, always satisfying $f(\pi \cdot \mathbf{A}) = \pi \cdot f(\mathbf{A})$. As a neural network f is often built as a stack of linear layers and non-linearities, its construction reduces to finding invariant and equivariant *linear* layers.

Invariant and equivariant linear layers Many (hyper)graph neural networks rely on message passing [Gilmer et al., 2017, Feng et al., 2019], which is a restricted equivariant operator. Alternatively, *maximally expressive* linear layers for higher-order tensors have been characterized by Maron et al. [2019b]. Specifically, invariant linear layers $L_{k\to 0} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{d'}$ and equivariant linear layers $L_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ were identified. Given an order-k input $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, the order-$l$ output of an equivariant linear layer $L_{k\to l}$ is written as follows, with indicator $\mathbb{1}$ and multi-indices $\mathbf{i} \in [n]^k, \mathbf{j} \in [n]^l$:

$$L_{k\to l}(\mathbf{A})_\mathbf{j} = \sum_\mu \sum_\mathbf{i} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\mu} \mathbf{A}_\mathbf{i} w_\mu + \sum_\lambda \mathbb{1}_{\mathbf{j}\in\lambda} b_\lambda, \tag{2.15}$$

where $w_\mu \in \mathbb{R}^{d\times d'}$, $b_\lambda \in \mathbb{R}^{d'}$ are weight and bias parameters, and μ and λ are *equivalence classes* of order-$(k + l)$ and order-l multi-indices, respectively.

The equivalence classes can be interpreted as a partitioning of a multi-index space: the order-$(k + l)$ equivalence classes μ specifies a partitioning of $[n]^{k+l}$, and order-l equivalence classes λ specifies a partitioning of $[n]^l$. The number of the equivalence classes (the size of partitioning) depends only on orders k and l. With $\mathrm{b}(k)$ the k-th Bell number, there are $\mathrm{b}(k + l)$ equivalence classes μ for the weight and $\mathrm{b}(l)$ equivalence classes λ for the bias. For the first-order layer $L_{1\to 1}$, there are $\mathrm{b}(2) = 2$ equivalence classes μ_1, μ_2 for the weight, specifying the partitioning of $[n]^2$ as $\{\mu_1, \mu_2\}$ where $\mu_1 = \{(i, j)|i = j\}$ and $\mu_2 = \{(i, j)|i \neq j\}$. More details can be found in Maron et al. [2019b], Kim et al. [2021a].

Related work

Equivariant GNNs Based on the maximally expressive equivariant linear layers (Equation (2.15)), a bouquet of permutation invariant or equivariant neural networks was formulated. A representative example is *equivariant GNN* [Maron et al., 2019b] (also called k-IGN [Maron et al., 2019a, Chen et al., 2020b]), built by stacking the equivariant linear layers and non-linearity. Their theoretical expressive power has been extensively studied [Zaheer et al., 2017, Keriven and Peyré, 2019, Maron et al., 2019a,c, Chen et al., 2020b], leading to successful variants in set and graph learning [Maron et al., 2020, Serviansky et al., 2020, Kim et al., 2021a]. In particular, practical variants such as higher-order transformer [Kim et al., 2021a] unified the equivariant GNNs and the transformer architecture [Vaswani et al., 2017, Lee et al., 2019], surpassing the performance of message-passing GNNs in large-scale molecular graph regression.

Challenges in hypergraph learning Despite the potential advantages, equivariant GNNs were rarely considered for hypergraph learning outside Albooyeh et al. [2019] and not implemented except for the restricted k-uniform hyperedge prediction [Serviansky et al., 2020, Kim et al., 2021a]. We find two challenges. First, despite the recent tricks to reduce computational cost [Kim et al., 2021a], the parameter count still grows rapidly to Bell number of input order [Berend and Tassa, 2010]. This makes any $L_{k\to l}$ with $k+l>4$ challenging to use, as $k+l=5$ already leads to 52 weight matrices. Second, in inductive learning [Hamilton et al., 2017, Zhang et al., 2018a] where a model is tested on unseen nodes or hypergraphs, the model can be required to process unseen-order hyperedges that surpass the maximal order in the training data. This is not straightforward for equivariant GNNs, because the fixed-order tensors that underlie $L_{k\to l}$ require pre-specifying the max hyperedge order (k,l) that can be processed.

2.2.2 Method

We now proceed to our framework on practical equivariant GNNs for general hypergraph data. All missing proofs can be found in Appendix A.2.1. In practical setups that assume undirected hypergraphs [Feng et al., 2019, Yadati et al., 2019, Arya et al., 2020, Dong et al., 2020, Bai et al., 2021, Chien et al., 2022], a hypergraph $G=(V,E,\mathbf{X})$ is defined by a set of n nodes V, a set of m hyperedges E, and features $\mathbf{X}\in\mathbb{R}^{m\times d}$ of the hyperedges. Each hyperedge $e\in E$ is a subset of node set V, and its order $|e|$ indicates its type. For example, a first-order edge $\{i\}$ represents an i-th node; a second-order edge $\{i,j\}$ represents a pairwise link of i-th and j-th nodes; in general, an order-k edge $\{i_1,...,i_k\}$ represents a hyperedge that links k nodes. By $\mathbf{X}_e\in\mathbb{R}^d$ we denote the feature attached to a hyperedge e. We assume that node and hyperedge features are both d-dimensional [Maron et al., 2019b,a, Kim et al., 2021a, Chen et al., 2020b]; to handle different dimensionalities, we simply let $d=(d_v+d_e)$ and place node features at first d_v channels and hyperedge features at last d_e channels.

Note that the hypergraphs $(V,E,\mathbf{X})$ do not directly align with the higher-order tensors $\mathbf{A}\in\mathbb{R}^{n^k\times d}$ described in Section 2.2.1. Unlike them, hypergraphs of our interest are sparse, their hyperedges are undirected, and each hyperedge contains unique node indices. As equivariant GNNs (Section 2.2.1) build upon tensors, it is necessary to establish a connection between the hypergraphs and the tensors.

A hypergraph is a sequence of tensors

To describe hypergraphs $(V,E,\mathbf{X})$ using higher-order tensors $\mathbf{A}\in\mathbb{R}^{n^k\times d}$, it is convenient to introduce *k-uniform hypergraphs*. A hypergraph is k-uniform if all of its hyperedges are exactly of order-k. For

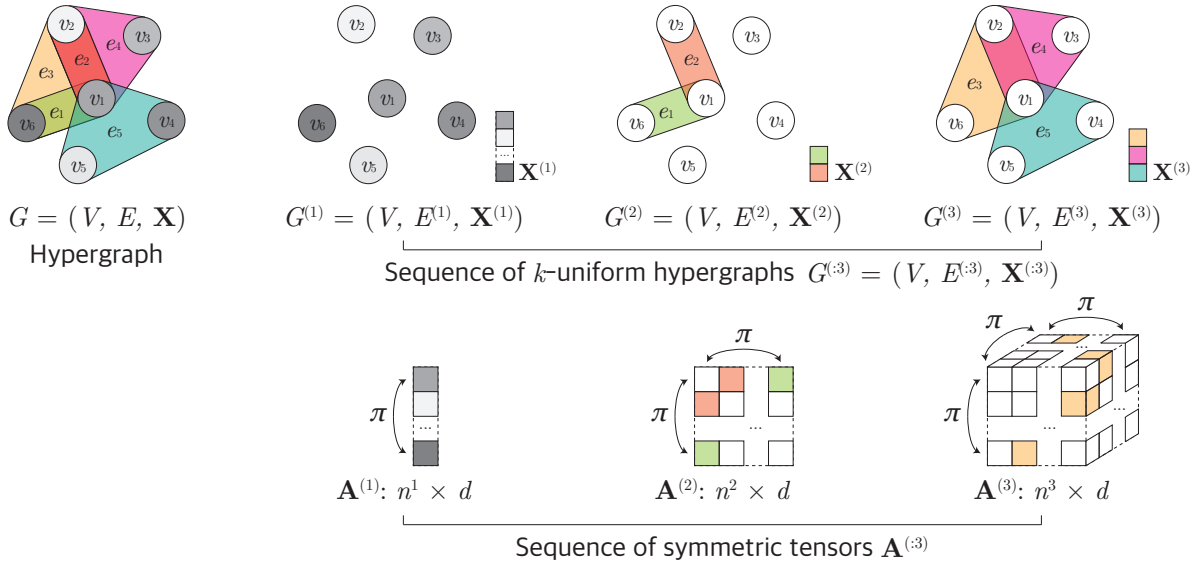


Figure 2.4: A hypergraph as a sequence of k-uniform hypergraphs (Definition 1), or equivalently, as a sequence of symmetric tensors (Definition 3). Note that nodes are handled as first-order hyperedges.

example, a graph without self-loops is 2-uniform, and a triangle mesh is 3-uniform. From that, we can define equivalent representation of a hypergraph as a *sequence* of k-uniform hypergraphs:

Definition 1. *The* sequence representation *of a hypergraph* $(V, E, \mathbf{X})$ *with maximum hyperedge order* K *is a sequence of* k*-uniform hypergraphs with* $k \leq K$*, written as* $(V, E^{(k)}, \mathbf{X}^{(k)})_{k \leq K} = (V, E^{(:K)}, \mathbf{X}^{(:K)})$*, where* $E^{(k)}$ *as the set of all order-*k *hyperedges in* E *and* $\mathbf{X}^{(k)}$ *as a row stack of features* $\{\mathbf{X}_e | e \in E^{(k)}\}$*.*

As the collection $(E^{(k)})_{k \leq K}$ forms a partition of E, we can retrieve the original hypergraph $(V, E, \mathbf{X})$ from its sequence representation $(V, E^{(k)}, \mathbf{X}^{(k)})_{k \leq K}$ by using the union of $(E^{(k)})_{k \leq K}$ for E and the concatenation of $(\mathbf{X}^{(k)})_{k \leq K}$ for $\mathbf{X}$.

The concept of uniform hypergraph is convenient as we can draw an equivalent representation as a *symmetric* tensor [Chien et al., 2022, Kofidis and Regalia, 2002]. An order-k tensor $\mathbf{A}$ is symmetric if its entries are invariant under re-ordering of indices, e.g., $\mathbf{A}_{ij} = \mathbf{A}_{ji}$, $\mathbf{A}_{ijk} = \mathbf{A}_{kij} = ...$, and so on. From that, we define the equivalent representation of a k-uniform hypergraph as an order-k symmetric tensor[6]:

Definition 2. *The* tensor representation *of* k*-uniform hypergraph* $(V, E^{(k)}, \mathbf{X}^{(k)})$ *is an order-*k *symmetric tensor* $\mathbf{A}^{(k)} \in \mathbb{R}^{n^k \times d}$ *defined as follows:*

$$\mathbf{A}^{(k)}_{(i_1,...,i_k)} = \begin{cases} \mathbf{X}^{(k)}_e & \text{if } e = \{i_1, ..., i_k\} \in E^{(k)} \\ 0 & \text{otherwise} \end{cases} . \tag{2.16}$$

From $\mathbf{A}^{(k)}$, we can retrieve the original k-uniform hypergraph $(V, E^{(k)}, \mathbf{X}^{(k)})$ by identifying the indices of all nonzero entries of $\mathbf{A}^{(k)}$ to construct $E^{(k)}$, and then using $E^{(k)}$ to index $\mathbf{A}^{(k)}$ to obtain $\mathbf{X}^{(k)}$.

Now, directly combining Definitions 1 and 2, we define the equivalent representation of a hypergraph as a sequence of higher-order tensors:

Definition 3. *The* tensor sequence representation *of a hypergraph* $(V, E, \mathbf{X})$ *with maximum hyperedge order* K *is a sequence of symmetric higher-order tensors* $(\mathbf{A}^{(k)})_{k \leq K} = \mathbf{A}^{(:K)}$*, where each* $\mathbf{A}^{(k)}$ *is the*

[6]Higher-order tensors can in principle represent directed hypergraphs as well; we constrain them to be symmetric to specifically represent undirected hypergraphs.

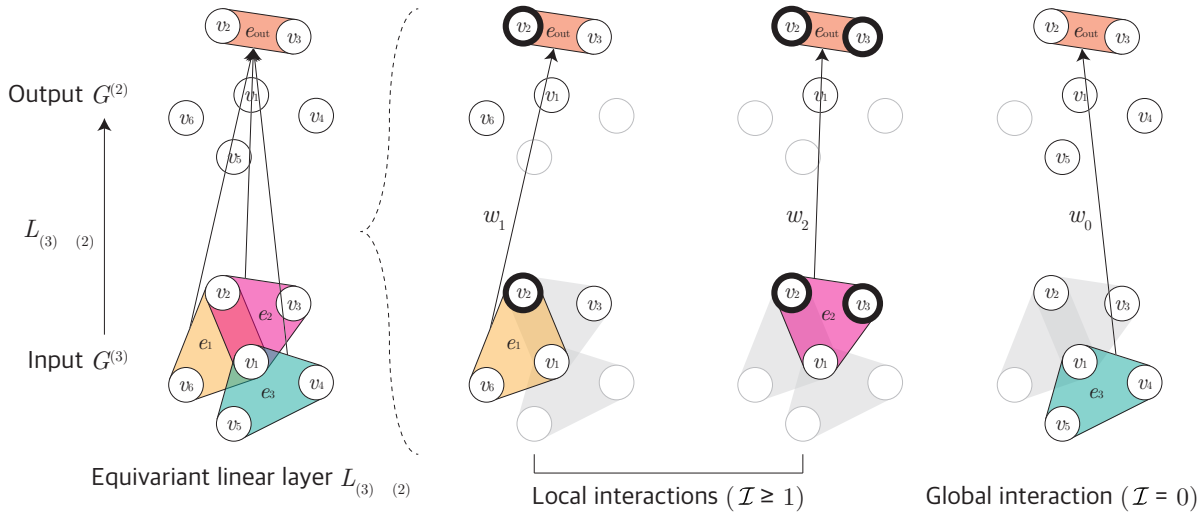


Figure 2.5: Illustration of an equivariant linear layer $L_{(3)\to(2)}$ as in Equation (2.17). The layer uses different weights $w_{\mathcal{I}}$ for different overlaps $\mathcal{I}$ between input and output hyperedges. This gives rise to local interactions similar to message passing ($\mathcal{I} \geq 1$) and global interactions ($\mathcal{I} = 0$) as global sum-pooling.

tensor representation (Definition 2) of each k-uniform hypergraph $(V, E^{(k)}, \mathbf{X}^{(k)})$ that comes from the sequence representation of the hypergraph $(V, E^{(k)}, \mathbf{X}^{(k)})_{k\leq K} = (V, E^{(:K)}, \mathbf{X}^{(:K)})$ (Definition 1).

An illustration is in Figure 2.4. Note that we can include node features as $\mathbf{A}^{(1)}$. Now, our problem of interest reduces to identifying a function f that operates on sequences of tensors $\mathbf{A}^{(:K)}$ that represent hypergraphs. Invariance and equivariance (Section 2.2.1) applies similarly. A node permutation $\pi \in S_n$ acts on a tensor sequence $\mathbf{A}^{(:K)}$ by jointly acting on each tensor, $\pi \cdot \mathbf{A}^{(:K)} = (\pi \cdot \mathbf{A}^{(k)})_{k\leq K}$. An invariant f always satisfies $f(\pi \cdot \mathbf{A}^{(:K)}) = f(\mathbf{A}^{(:K)})$, and an equivariant f always satisfies $f(\pi \cdot \mathbf{A}^{(:K)}) = \pi \cdot f(\mathbf{A}^{(:K)})$.

Equivariant linear layers for hypergraphs

In Definition 3, we represented a hypergraph as a sequence of symmetric tensors $(\mathbf{A}^{(k)})_{k\leq K}$, each $\mathbf{A}^{(k)}$ representing a k-uniform hypergraph. We now use equivariant linear layers $L_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ in Equation (2.15) to formalize linear layers that input and output hypergraphs. The idea is to combine all pairwise linear maps between tensors (i.e., k-uniform hypergraphs) of input and output sequences. Although simple, we prove that this gives the *maximally expressive* equivariant linear layer for hypergraphs.

Linear layers for k-uniform hypergraphs In Section 2.2.1, we argued that the equivariant linear layer $L_{k\to l}$ cannot be practically used due to the prohibitive number of $\mathrm{b}(k+l)$ weights and $\mathrm{b}(l)$ biases. Yet, when the input and output tensors are restricted to k- and l-uniform hypergraphs respectively, we can show that the layer reduces to $\mathcal{O}(k+l)$ weights and a single bias:

Proposition 4. *Assume that the input and output of equivariant linear layer $L_{k\to l}$ (Equation (2.15)) are constrained to symmetric tensors that represent k- and l-uniform hypergraphs respectively (Equation (2.16)). Then it reduces to $L_{(k)\to(l)}$ below:*

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\mathcal{I}=1}^{\min(k,l)}\sum_{\mathbf{i}}\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\mathcal{I}} + \sum_{\mathbf{i}}\mathbf{A}^{(k)}_{\mathbf{i}}w_0 + b_l\right), \tag{2.17}$$

where $w_o, w_{\mathcal{I}} \in \mathbb{R}^{d \times d'}$, $b_l \in \mathbb{R}^{d'}$ *are weight and bias,* $|\mathbf{i}|$ *is number of distinct elements in* $\mathbf{i}$*, and* $|\mathbf{i} \cap \mathbf{j}|$ *is number of distinct intersecting elements in* $\mathbf{i}$ *and* $\mathbf{j}$*.*

The idea for the proof is that, if the input and output are constrained to tensors that represent uniform hypergraphs (Equation (2.16)), a large number of parameters are tied to adhere to the symmetry. This leads to much fewer parameters compared to the original layer $L_{k \to l}$. Still, $L_{(k) \to (l)}$ (Equation (2.17)) is a maximally expressive linear layer as it produces identical outputs to (unreduced) $L_{k \to l}$.

Notably, Equation (2.17) reveals that maximal expressiveness comprises sophisticated local message passing augmented with global interaction. In the first term of Equation (2.17), the constraint $\mathbb{1}_{|\mathbf{i} \cap \mathbf{j}| > 0}$ specifies local dependency between *incident* input and output hyperedges having at least one overlapping node. This local interaction is more *fine-grained* than conventional message passing, as it uses separate weights $w_{\mathcal{I}}$ for different numbers of overlapping nodes $\mathcal{I}$ (Figure 2.5). This is reminiscent of recent work in subgraph message passing [Bevilacqua et al., 2022a] that improves expressivity of GNNs. On top of that, the layer contains intrinsic global interaction via pooling in the second term of Equation (2.17), which reminds of the virtual node or global attention [Li et al., 2017, Ishiguro et al., 2019, Knyazev et al., 2019, Louis et al., 2020, Wu et al., 2021a] that improves expressive power [Puny et al., 2020].

Linear layers for hypergraphs We now construct maximally expressive equivariant linear layers for undirected hypergraphs. As in Definition 3, a hypergraph can be represented as a sequence of tensors $(\mathbf{A}^{(k)})_{k \leq K} = \mathbf{A}^{(:K)}$. Thus, we construct the linear layer $L_{(:K) \to (:L)}$ to input and output those tensor sequences while being equivariant $L_{(:K) \to (:L)}(\pi \cdot \mathbf{A}^{(:K)}) = \pi \cdot L_{(:K) \to (:L)}(\mathbf{A}^{(:K)})$. For this, we simply use all pairwise linear layers $L_{(k) \to (l)}$ (Equation (2.17)) between tensors of input and output sequences:

$$L_{(:K) \to (:L)}(\mathbf{A}^{(:K)}) = \left(\sum_{k \leq K} L_{(k) \to (l)}(\mathbf{A}^{(k)}) \right)_{l \leq L}. \tag{2.18}$$

For better interpretation, we plug Equation (2.17) into Equation (2.18) and rewrite it with respect to $\mathbf{j}$-th entry of l-th (order-l) output tensor:

$$L_{(:K) \to (:L)}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \sum_{k \leq K} \sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i} \cap \mathbf{j}|=\mathcal{I}} \mathbf{A}^{(k)}_{\mathbf{i}} w_{k,l,\mathcal{I}} + \mathbb{1}_{|\mathbf{j}|=l} \sum_{k \leq K} \sum_{\mathbf{i}} \mathbf{A}^{(k)}_{\mathbf{i}} w_{k,l,0} + \mathbb{1}_{|\mathbf{j}|=l} b_l. \tag{2.19}$$

Note that we added subscripts (k, l) to w_0, $w_{\mathcal{I}}$ to differentiate between weights from each sublayer $L_{(k) \to (l)}$ as they are involved in different computations. On the other hand, the biases from sublayers $(L_{(k) \to (l)})_{k \leq K}$ carry out precisely the same computation, and can be merged to a single bias b_l. As a result, $L_{(:K) \to (:L)}$ contains $\sum_{l \leq L, k \leq K}(1 + \min(k, l))$ weights and L biases, achieving better scalability than the original $L_{K \to L}$ that has exponentially many weights and biases.

Similar to sublayers $L_{(k) \to (l)}$ (Equation (2.17)), we see that the combined layer for general hypergraphs $L_{(:K) \to (:L)}$ (Equation (2.19)) is a mixture of fine-grained local message passing and global interaction. The local interactions utilize different weights $w_{k,l,\mathcal{I}}$ for each triplet $(k, l, \mathcal{I})$ that specifies dependency between order-k input and order-l output hyperedges with $\mathcal{I}$ overlapping nodes. Similarly, global interactions (pooling) utilize different weights $w_{k,l,0}$ for each pair (k, l), specifying global dependency between all order-k input and order-l output hyperedges. Finally, different biases b_l are assigned for each output hyperedge order l. Importantly, we can show the following:

Theorem 3. *$L_{(:K)\to(:L)}$ (Equation (2.18)) is the maximally expressive equivariant linear layer for undirected hypergraphs represented as tensor sequences.*

Similar as in Proposition 4, the idea for the proof is to appropriately constrain the input and output of the maximally expressive equivariant linear layer $L_{K\to L}$, and observe that most of its parameters are tied and reduced, leading to $L_{(:K)\to(:L)}$. Still, the layer retains the maximal expressiveness of the original layer $L_{K\to L}$ as it produces the identical output.

Equivariant hypergraph neural networks

We have introduced equivariant linear layers for general undirected hypergraphs $L_{(:K)\to(:L)}$ by composing order-specific sublayers $L_{(k)\to(l)}$ for $k \leq K$, $l \leq L$ and proved their maximal expressiveness. However, these layers are still unsuitable to be used in practice because they cannot input or output hypergraphs with orders exceeding (K, L), and the number of weights and biases grows at least linearly to (K, L) that can reach several hundred in practice. To resolve the problems jointly, we propose *Equivariant Hypergraph Neural Network* (EHNN) that introduces intrinsic trainable parameter sharing via *hypernetworks* [Ha et al., 2017]. More specifically, we impose parameter sharing within $L_{(:K)\to(:L)}$ and across all sublayers $L_{(k)\to(l)}$ via two hypernetworks, each for weights and biases[7].

As a result, an EHNN layer is defined as follows, with hypernetworks $\mathcal{W} : \mathbb{N}^3 \to \mathbb{R}^{d\times d'}$, $\mathcal{B} : \mathbb{N} \to \mathbb{R}^{d'}$ inferring all weights $w_{k,l,\mathcal{I}}$ and biases b_l (Equation (2.19)) from the subscripts $(k, l, \mathcal{I})$ and (l) respectively:

$$\begin{aligned}\mathrm{EHNN}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \sum_{k\leq K} \sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} \mathbf{A}^{(k)}_{\mathbf{i}} \mathcal{W}(k, l, \mathcal{I}) \\ + \mathbb{1}_{|\mathbf{j}|=l} \sum_{k\leq K} \sum_{\mathbf{i}} \mathbf{A}^{(k)}_{\mathbf{i}} \mathcal{W}(k, l, 0) + \mathbb{1}_{|\mathbf{j}|=l} \mathcal{B}(l). \end{aligned} \tag{2.20}$$

In principle, this preserves maximal expressiveness of $L_{(:K)\to(:L)}$ when $\mathcal{W}$ and $\mathcal{B}$ are parameterized as MLPs, as by universal approximation they can learn any lookup table that maps subscripts to weights and biases [Hornik et al., 1989]. Furthermore, as hypernetworks $\mathcal{W}$ and $\mathcal{B}$ can produce weights for arbitrary hyperedge orders $(k, l, \mathcal{I})$, we can remove the bound in hyperedge orders from the specification of the layer and use a single EHNN layer with bounded parameters to any hypergraphs with unbounded or unseen hyperedge orders. Conclusively, EHNN layer is by far the first attempt that is maximally expressive (i.e., is able to model $L_{(:K)\to(:L)}$ (Theorem 3) and thus exhausts the full space of equivariant linear layers on undirected hypergraphs) while being able to process arbitrary-order hypergraphs by construction.

Practical realizations

The EHNN layer in Equation (2.20) is conceptually elegant, but in practice it can be costly as we need to explicitly hold all output matrices of the hypernetwork $\mathcal{W}(k, l, \mathcal{I}) \in \mathbb{R}^{d\times d'}$ in memory. This motivates us to seek for simpler realizations of EHNN that can be implemented efficiently while retaining the maximal expressiveness. To this end, we propose EHNN-MLP that utilizes three consecutive MLPs to approximate the role of the weight hypernetwork, and also propose its extension EHNN-transformer with self-attention. Then, we finish the section by providing a comparative analysis of EHNN-MLP and EHNN-transformer with respect to the existing message-passing hypergraph neural networks.

[7]Note that we do not share the hypernetworks across different levels of layers.

Realization with MLP We first introduce *EHNN-MLP*, a simple realization of EHNN with three elementwise MLPs $\phi_{1:3}$ where each $\phi_p : \mathbb{N} \times \mathbb{R}^{d_p} \to \mathbb{R}^{d'_p}$ takes a positive integer as an auxiliary input. The intuition here is to *decompose* the weight application with hypernetwork $\mathcal{W}(k, l, \mathcal{I})$ into three consecutive MLPs $\phi_1(k, \cdot)$, $\phi_2(\mathcal{I}, \cdot)$, and $\phi_3(l, \cdot)$, eliminating the need to explicitly store the inferred weights for each triplet $\mathcal{W}(k, l, \mathcal{I})$. We characterize EHNN-MLP as follows:

$$\text{EHNN-MLP}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \phi_3\left(l, \sum_{\mathcal{I}\geq 0} \phi_2\left(\mathcal{I}, \sum_{k\leq K}\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}})\right)\right) + \mathcal{B}(l),$$
$$\text{where } \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} = \begin{cases} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} & \text{if } \mathcal{I} \geq 1 \\ 1 & \text{if } \mathcal{I} = 0 \end{cases}, \tag{2.21}$$

where we omit the output constraint $\mathbb{1}_{|\mathbf{j}|=l}$ for brevity, and introduce a binary scalar $\mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}}$ to write local ($\mathcal{I} \geq 1$) and global ($\mathcal{I} = 0$) interactions together.

Now we show that an EHNN-MLP layer can realize any EHNN layer:

Theorem 4. *An EHNN-MLP layer (Equation* (2.21)*) can approximate any EHNN layer (Equation* (2.20)*) to an arbitrary precision.*

The proof is done by leveraging the universal approximation property [Hornik et al., 1989] to model appropriate functions with the MLPs $\phi_{1:3}$, so that the output of EHNN-MLP (Equation (2.21)) accurately approximates the output of EHNN (Equation (2.20)). As a result, with EHNN-MLP, we now have a practical model that can approximate the maximally expressive linear layer for general undirected hypergraphs.

In our implementation of the MLPs $\phi_{1:3}$, we first transform the input order (k, l or $\mathcal{I}$) into a continuous vector called *order embedding*, and combine it with the input feature through concatenation. This way, the order embeddings are served similarly as the positional encoding used in transformer [Vaswani et al., 2017] with a subtle difference that it indicates the order of the input or output hyperedges. We employ sinusoidal encoding [Vaswani et al., 2017] to obtain order embedding due to its efficiency and, more importantly, to aid extrapolation to unseen hyperedge orders in testing.

Realization as a transformer While EHNN-MLP (Equation (2.21)) theoretically inherits the high expressive power of EHNN, in practice, its static sum-pooling can be limited in accounting for relative importance of input hyperedges. A solution for this is to introduce more sophisticated pooling. In particular, the attention mechanism of transformers [Vaswani et al., 2017] was shown to offer a large performance gain in set and (hyper)graph modeling [Lee et al., 2019, Kim et al., 2021a, Chien et al., 2022] via dynamic weighting of input elements. Thus, we extend EHNN-MLP with multihead attention coefficients $\boldsymbol{\alpha}^{h,\mathcal{I}}_{\mathbf{i},\mathbf{j}}$ and introduce *EHNN-transformer*, an advanced realization of EHNN:

$$\text{EHNN-transformer}(\mathbf{A}^{(:K)}) = \text{Attn}(\mathbf{A}^{(:K)}) + \text{MLP}(\text{Attn}(\mathbf{A}^{(:K)})),$$
$$\text{where Attn}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \phi_3\left(l, \sum_{\mathcal{I}\geq 0} \phi_2\left(\mathcal{I}, \sum_{h=1}^{H}\sum_{k\leq K}\sum_{\mathbf{i}} \boldsymbol{\alpha}^{h,\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}}) w^V_h\right)\right), \tag{2.22}$$

where we omit the output constraint $\mathbb{1}_{|\mathbf{j}|=l}$ and bias $\mathcal{B}(l)$ for brevity. H denotes the number of heads and $w^V_h \in \mathbb{R}^{d\times d_v}$ denotes the value weight matrix. To compute attention coefficients $\boldsymbol{\alpha}^{h,\mathcal{I}}_{\mathbf{i},\mathbf{j}}$ from the input, we introduce additional query and key (hyper)networks $\mathcal{Q} : \mathbb{N} \to \mathbb{R}^{H\times d_H}$ and $\mathcal{K} : \mathbb{N} \times \mathbb{R}^d \to \mathbb{R}^{H\times d_H}$ and

characterize scaled dot-product attention [Vaswani et al., 2017] as follows:

$$\boldsymbol{\alpha}_{\mathbf{i},\mathbf{j}}^{h,\mathcal{I}} = \begin{cases} \sigma\left(\mathcal{Q}(\mathcal{I})_h \mathcal{K}\left(\mathcal{I}, \phi_1(k, \mathbf{A}_{\mathbf{i}}^{(k)})\right)_h^\top / \sqrt{d_H} \cdot \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\right) & \text{if } \mathcal{I} \geq 1 \\ \sigma\left(\mathcal{Q}(0)_h \mathcal{K}\left(\mathcal{I}, \phi_1(k, \mathbf{A}_{\mathbf{i}}^{(k)})\right)_h^\top / \sqrt{d_H}\right) & \text{if } \mathcal{I} = 0 \end{cases}, \tag{2.23}$$

where $\sigma(\cdot)$ denotes activation, often chosen as softmax normalization. Note that the query $\mathcal{Q}(\mathcal{I})$ is agnostic to output index $\mathbf{j}$, following prior works on set and (hyper)graph attention [Lee et al., 2019, Chien et al., 2022]. Although this choice of attention mechanism has a drawback that assigning importance to input ($\mathbf{i}$) depending on output ($\mathbf{j}$) is not straightforward, we choose it in favor of scalability.

Analysis

We finish the section by providing a comparative analysis of EHNNs with respect to the existing message passing networks for hypergraphs. We specifically compare against *AllSet* [Chien et al., 2022], as it represents a highly general framework that subsumes most existing hypergraph neural networks. Their MLP-based characterization *AllDeepSets* can be written with two MLPs ϕ_1 and ϕ_2 as follows:

$$\text{AllDeepSets}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \phi_2 \left(\sum_{k\leq K} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|\geq 1} \phi_1(\mathbf{A}_{\mathbf{i}}^{(k)}) \right). \tag{2.24}$$

We show the below by reducing EHNN-MLP to AllDeepSets through ablation:

Theorem 5. *An AllDeepSets layer (Equation* (2.24)*) is a special case of EHNN-MLP layer (Equation* (2.21)*), while the opposite is not true.*

Theorem 5 leads to the following corollary:

Corollary 2. *An EHNN-MLP layer is more expressive than an AllDeepSets layer and also all hypergraph neural networks that AllDeepSets subsumes.*

In the proof of Theorem 5, we leverage a simple property that message passing of AllDeepSets cannot account for long-range dependency modeling of EHNN. Then a natural question is: if we rule out the presence of global interaction, can the *local message passing* of EHNN be potentially better than that of current message-passing networks? Our analysis suggests so. More specifically, in practical scenarios when local message aggregation cannot be multiset universal, the explicit hyperedge order conditioning of EHNN can help. To demonstrate this, we bring a powerful, transformer-based characterization of AllSet framework called AllSetTransformer:

$$\begin{aligned} &\text{AllSetTransformer}(\mathbf{A}^{(:K)}) = \text{AllSetAttn}(\mathbf{A}^{(:K)}) + \text{MLP}(\text{AllSetAttn}(\mathbf{A}^{(:K)})), \\ &\text{where AllSetAttn}(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l} \sum_{h=1}^{H} \sum_{k\leq K} \sum_{\mathbf{i}} \boldsymbol{\alpha}_{\mathbf{i},\mathbf{j}}^h \phi_1(\mathbf{A}_{\mathbf{i}}^{(k)}), \end{aligned} \tag{2.25}$$

with H the number of heads, and attention coefficients $\boldsymbol{\alpha}_{\mathbf{i},\mathbf{j}}^h$ computed with a query matrix $Q \in \mathbb{R}^{H\times d_H}$, a key network $\mathcal{K} : \mathbb{R}^d \to \mathbb{R}^{H\times d_H}$, and activation $\sigma(\cdot)$:

$$\boldsymbol{\alpha}_{\mathbf{i},\mathbf{j}}^h = \sigma\left(Q_h \mathcal{K}\left(\mathbf{A}_{\mathbf{i}}^{(k)}\right)_h^\top / \sqrt{d_H} \cdot \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\right).$$

Table 2.5: Results for synthetic k-edge identification. We show averaged best test accuracy (%) over 5 runs with standard deviation.

	Test involves only seen k	Test involves unseen k	
		Interpolation	Extrapolation
AllDeepSets	76.99 ± 0.98	79.6 ± 0.86	79.01 ± 2.82
AllSetTransformer	77.61 ± 2.27	78.61 ± 2.367	77.35 ± 1.89
EHNN-MLP	98.02 ± 0.73	90.70 ± 2.90	85.65 ± 2.89
EHNN-transformer	**99.69 ± 0.31**	**92.31 ± 1.47**	**90.19 ± 5.51**

An interesting fact is that when we take the activation for attention $\sigma(\cdot)$ as *any* kinds of normalization including softmax, attention mechanism of AllSetTransformer can fail to model simple multiset functions. A simple example is that they cannot *count* the number of input hyperedges that overlap with each output hyperedge. As a simple demonstration, if input hypergraph is a node set $\mathbf{A}^{(1)} \in \mathbb{R}^{n \times d}$ with identical features $\mathbf{A}^{(1)}_i = \mathbf{1}$, an AllSetTransformer layer that outputs features of hyperedges E cannot learn node counting *i.e.*, for output $\mathbf{Y}$, $\mathbf{Y}_e = |e|$ for $e \in E$. This is because, as attention normalizes over keys, all outputs of AllSetAttn (Equation (2.25)) is always $H\phi_1(\mathbf{1})$. On the other hand, the hyperedge order-aware message passing of EHNN-transformer (Equation (2.22)) can easily solve the problem by forwarding l.

We finish with stronger expressiveness of EHNN-transformer compared to AllSetTransformer:

Theorem 6. *An AllSetTransformer layer (Equation* (2.25)*) is a special case of EHNN-transformer layer (Equation* (2.22)*), while the opposite is not true.*

2.2.3 Experiments

We test EHNN on a range of inductive and transductive hypergraph learning problems, including synthetic node classification problem, real-world semi-supervised classification, and visual keypoint matching. For the real-world tasks, we use 10 semi-supervised classification datasets used in Chien et al. [2022] and two visual keypoint matching datasets used in Wang et al. [2021].

Synthetic k-edge identification

We devise a simple but challenging synthetic node classification task termed k-edge identification to demonstrate how the expressive power of EHNN can help learn complex hypergraph functions. In an input hypergraph, we pick a random hyperedge and mark its nodes with a binary label. The task is to identify all other nodes whose hyperedge order is as same as the marked one. The model is required to propagate the information of marked hyperedge globally while also reasoning about the fine-grained structure of individual hyperedges for comparison. We use 100 train and 20 test hypergraphs, each with 100 nodes and randomly wired 10 hyperedges of orders $\in \{2, ..., 10\}$. To test generalization to unseen orders, we add two train sets where hyperedges are sampled *excluding* orders $\{5, 6, 7\}$ (interpolation) or orders $\{8, 9, 10\}$ (extrapolation). We evaluate EHNN-MLP/-transformer with AllDeepSets/AllSetTransformer [Chien et al., 2022] as message-passing baselines.

The test performances are in Table 2.5. EHNN achieves significant improvement over message passing, producing almost perfect predictions. The result advocates that even for simple tasks there are cases where the high expressive power of a network is essential. Furthermore, we observe evidence that the model can interpolate or even extrapolate to unseen hyperedge orders. This supports the use of hypernetworks to infer parameters for potentially unseen orders.

Table 2.6: Results for semi-supervised node classification. Average accuracy (%) over 20 runs are shown. Gray shade indicate the best result, and blue shade indicate results within one standard deviation of the best. Baseline scores are taken from Chien et al. [2022].

	Zoo	20Newsgroups	mushroom	NTU2012	ModelNet40	Yelp	House(1)	Walmart(1)	House(0.6)	Walmart(0.6)	avg. rank (↓)
MLP	87.18	**81.42**	**100.00**	85.52	96.14	31.96	67.93	45.51	81.53	63.28	6.4
CEGCN	51.54	OOM	95.27	81.52	89.92	OOM	62.80	54.44	64.36	59.78	11.5
CEGAT	47.88	OOM	96.60	82.21	92.52	OOM	69.09	51.14	77.25	59.47	10.5
HNHN	93.59	81.35	**100.00**	89.11	97.84	31.65	67.80	47.18	78.78	65.80	5.9
HGNN	92.50	80.33	98.73	87.72	95.44	33.04	61.39	62.00	66.16	77.72	7.8
HCHA	93.65	80.33	98.70	87.48	94.48	30.99	61.36	62.45	67.91	77.12	8.1
HyperGCN	N/A	81.05	47.90	56.36	75.89	29.42	48.31	44.74	78.22	55.31	12.4
UniGCNII	93.65	81.12	99.96	89.30	98.07	31.70	67.25	54.45	80.65	72.08	5.8
HAN (full batch)	85.19	OOM	90.86	83.58	94.04	OOM	71.05	OOM	83.27	OOM	9.9
HAN (minibatch)	75.77	79.72	93.45	80.77	91.52	26.05	62.00	48.57	82.04	63.1	10.6
AllDeepSets	95.39	81.06	99.99	88.09	96.98	30.36	67.82	64.55	80.70	78.46	5.4
AllSetTransformer	**97.50**	81.38	**100.00**	88.69	98.20	**36.89**	69.33	65.46	83.14	78.46	2.4
EHNN-MLP	91.15	81.31	99.99	87.35	97.74	35.80	67.41	65.65	82.29	78.80	5.0
EHNN-transformer	93.27	**81.42**	**100.00**	**89.60**	**98.28**	36.48	**71.53**	**68.73**	**85.09**	**80.05**	**1.6**

Table 2.7: Visual keypoint matching accuracy (%) on Willow test set.

	car	duck	face	motor	wine	avg.
GMN	38.85	38.75	78.85	28.08	45.00	45.90
NGM	77.50	85.87	99.81	77.50	89.71	86.08
NHGM	69.13	83.08	99.81	73.37	88.65	82.81
NMGM	74.95	81.33	99.83	78.26	92.06	85.29
IPCA-GM	79.58	80.20	99.70	73.37	83.75	83.32
CIE-H	9.37	8.87	9.88	11.84	9.84	9.96
BBGM	96.15	90.96	**100.00**	96.54	**99.23**	96.58
GANN-MGM	92.11	90.11	**100.00**	96.21	98.26	95.34
NGM-v2	94.81	89.04	**100.00**	96.54	95.87	95.25
NHGM-v2	89.33	83.17	**100.00**	92.60	95.96	92.21
EHNN-MLP	94.71	91.92	**100.00**	97.21	97.79	96.33
EHNN-transformer	**97.02**	**92.69**	**100.00**	**97.60**	98.08	**97.08**

Semi-supervised classification

To test EHNN in real-world hypergraph learning, we use 10 transductive semi-supervised node classification datasets [Chien et al., 2022]. The data is randomly split into 50% training, 25% validation, and 25% test. We run the experiment 20 times with random splits and initialization and report aggregated performance.

The results are in Table 2.6. Our methods often achieve favorable scores over strong baselines, e.g., AllDeepSets/AllSetTransformer, improving the state-of-the-art by 3.27% in Walmart (1), 1.82% in House (0.6), and 1.54% in Walmart (0.6). Notably, EHNN-transformer holds the best in most cases. This supports the notion that attention strengthens equivariant networks [Kim et al., 2021a, Lee et al., 2019], and also implies that the high expressivity of EHNN makes it strong in general hypergraph learning setups involving not only social networks but also vision and graphics (NTU2012 and ModelNet40).

Visual keypoint matching

To test EHNN in computer vision problems represented as hypergraph learning, we tackle visual keypoint matching. The task is considered challenging due to the discrepancy between the two images in terms of viewpoint, scale, and lighting. Following previous work [Wang et al., 2021], we view the problem as *hypergraph matching*, where keypoints of each image form a hypergraph. This is considered helpful as the hyperedge features can capture rotation- and scale-invariant geometric features such as angles.

Table 2.8: Visual keypoint matching accuracy (%) on PASCAL-VOC test set.

	aero	bike	bird	boat	botl	bus	car	cat	chair	cow	desk
GMN	40.67	57.62	58.19	51.38	77.55	72.48	66.90	65.04	40.43	61.56	65.17
PCA-GM	51.46	62.43	64.70	58.56	81.94	75.18	69.56	71.05	44.53	65.81	39.00
NGM	12.09	10.01	17.44	21.73	12.03	21.40	20.16	14.26	15.10	12.07	14.50
NHGM	12.09	10.01	17.44	21.73	12.03	21.40	20.16	14.26	15.10	12.07	14.50
IPCA-GM	50.78	62.29	63.87	58.94	79.46	74.18	72.60	71.52	41.42	64.12	36.67
CIE-H	52.26	66.79	69.09	59.76	83.38	74.61	69.93	71.04	43.36	69.20	76.00
BBGM	**60.06**	71.32	78.21	78.97	88.63	**95.57**	**89.52**	80.53	**59.34**	**77.80**	76.00
GANN-MGM	14.75	32.20	21.31	24.43	67.23	36.35	21.09	17.20	25.73	21.00	37.50
NGM-v2	42.88	61.70	63.63	75.62	84.66	90.58	75.34	72.26	44.42	66.67	74.50
NHGM-v2	57.04	71.88	76.06	**79.96**	**89.79**	93.70	86.16	80.76	56.36	76.70	74.33
EHNN-MLP	57.34	**73.89**	76.41	78.41	89.40	94.51	85.58	79.83	56.39	76.56	**91.00**
EHNN-transformer	60.04	72.36	**78.25**	78.59	87.61	93.77	87.99	**80.78**	58.76	76.29	81.17

	dog	horse	mbk	prsn	plant	sheep	sofa	train	tv	avg.
GMN	61.56	62.18	58.96	37.80	78.39	66.89	39.74	79.84	90.94	61.66
PCA-GM	67.82	65.18	65.71	46.21	83.81	70.51	49.88	80.87	93.07	65.36
NGM	12.83	12.05	15.69	09.76	21.00	17.10	15.12	31.11	24.88	16.52
NHGM	12.83	12.05	15.67	09.76	21.00	17.10	14.66	31.11	24.83	16.49
IPCA-GM	69.11	66.05	65.88	46.97	83.09	68.97	51.83	79.17	92.27	64.96
CIE-H	69.68	71.18	66.14	46.76	87.22	71.08	59.16	82.84	92.60	69.10
BBGM	**80.39**	77.80	76.48	**65.99**	**98.52**	**78.07**	76.65	97.61	94.36	**80.09**
GANN-MGM	16.16	20.16	25.92	19.20	53.76	18.34	26.16	46.30	72.32	30.85
NGM-v2	67.83	68.92	68.86	47.40	96.69	70.57	70.01	95.13	92.49	71.51
NHGM-v2	76.75	77.45	**76.81**	58.56	98.21	75.34	76.42	98.10	94.80	78.76
EHNN-MLP	76.57	**78.65**	75.54	58.92	98.31	76.53	**81.14**	98.08	**95.01**	79.90
EHNN-transformer	78.30	76.91	75.79	63.78	97.60	76.47	78.04	**98.53**	93.83	79.74

We then cast hypergraph matching to binary node classification on a single association hypergraph as in previous work [Wang et al., 2021].

We use two standard datasets [Wang et al., 2021]: Willow ObjectClass [Cho et al., 2013] and PASCAL-VOC [Everingham et al., 2010, Bourdev and Malik, 2009]. The Willow dataset consists of 256 images with 5 object categories. The PASCAL-VOC dataset contains 11,530 images with 20 object categories and is considered challenging due to the large variance in illumination and pose. We follow the training setup of NHGM-v2 [Wang et al., 2021] and only replace the hypergraph neural network module with EHNN-MLP/-transformer. The key difference between NHGM-v2 and our models is that NHGM-v2 utilizes two *separate* message passing networks, one on 2-edges and another on 3-edges, and aggregates node features as a weighted sum. In contrast, EHNN *mixes* the information from 2-edges and 3-edges extensively via shared MLP hypernetworks and global interactions.

The results are in Tables 2.7 and 2.8. On Willow, EHNN-transformer gives the best performance, improving over NHGM-v2 by 4.87%. On PASCAL-VOC, EHNNs improve over NHGM-v2 by $\sim 1\%$ and are competitive to the best (BBGM; 0.19% gap) that relies on a sophisticated combinatorial solver [Rolinek et al., 2020]. We conjecture that intrinsic and global mixing of 2-edge (distance) and 3-edge (angle) features improves hypergraph learning, and consequently benefits keypoint matching.

Ablation study

To identify the source of performance improvements, we perform an ablation study on k-edge identification. We gradually ablate each component of EHNN-MLP: the order conditioning in the elementwise MLPs (Equation (2.21)) and the global interactions (Equation (2.21)), until it reduces to message passing ($\approx$ AllDeepSets [Chien et al., 2022]). We also compare against naïve EHNN (Section 2.2.2)

Table 2.9: Ablation study on k-edge identification. We show averaged best test accuracy (%) over 5 runs with standard deviation. The results for AllDeepSets and EHNN-MLP are taken from Table 2.5.

	Global interaction	Order emb.	MLP realization	Seen k	Unseen k	
					Interpolation	Extrapolation
AllDeepSets	×	×	◯	76.99±0.98	79.6±0.86	79.01±2.82
EHNN-MLP (ablated)						
• w/o global and order	×	×	◯	78.94±1.18	78.6±1.66	77.97±1.83
• w/o global	×	◯	◯	80.34±2.87	77.86±2.38	79.56±3.03
• w/o order	◯	×	◯	84.05±1.77	81.30±3.56	80.17±3.34
EHNN (naïve)						
• Lookup table for $\mathcal{W}, \mathcal{B}$	◯	◯	×	87.09±2.49	84.09±1.29	80.20±2.21
• Hypernetwork for $\mathcal{W}, \mathcal{B}$	◯	◯	×	83.74±2.89	83.19±1.57	79.93±2.61
EHNN-MLP	◯	◯	◯	**98.02±0.73**	**90.70±2.90**	**85.65±2.89**

Table 2.10: Runtime and memory cost. We show aggregated results over 20 runs on a single A100 GPU.

	Forward (ms)	Backward (ms)	Peak memory (MB)
AllDeepSets	5.538±0.689	3.470±0.071	4.608±0.000
AllSetTransformers	6.883±0.657	5.037±0.681	5.077±0.000
EHNN (naïve)			
• Lookup table for $\mathcal{W}, \mathcal{B}$	28.68±0.535	71.71±3.597	14.43±0.000
• Hypernetwork for $\mathcal{W}, \mathcal{B}$	31.56±1.110	76.78±6.044	17.40±0.000
EHNN-MLP	7.517±0.865	7.179±0.548	7.361±0.000
EHNN-transformer	14.51±1.146	13.41±0.0962	11.98±0.000

which is maximally expressive but, unlike EHNN-MLP, is not realized as 3 elementwise MLPs. The results are in Table 2.9. Ablating any component of EHNN-MLP degrades performance until similar to AllDeepSets.

Runtime and memory cost

To test the computational efficiency of EHNN, we perform runtime and memory cost analysis using random hypergraphs with 1024 nodes and 128 randomly wired hyperedges with orders $\in \{2, ..., 10\}$. The results are outlined in Table 2.10. Naïve EHNN (Section 2.2.2) suffers from high time and memory cost as it requires to explicitly hold all order-specific weight matrices in memory. In contrast, EHNN-MLP/-transformer are more efficient, improving time and memory cost from 5-20× to 2-3× with respect to highly optimized message passing while still being maximally expressive.

2.2.4 Discussion

We proposed a class of hypergraph networks coined equivariant hypergraph neural network (EHNN). EHNN extends the foundations of equivariant GNNs to general undirected hypergraphs by representing a hypergraph as a sequence of tensors and combining equivariant linear layers on them. We further proposed EHNN-MLP/-transformer, practical realizations of EHNN based on hypernetworks. We showed that EHNN is more expressive than most message-passing networks and provided empirical evidence.

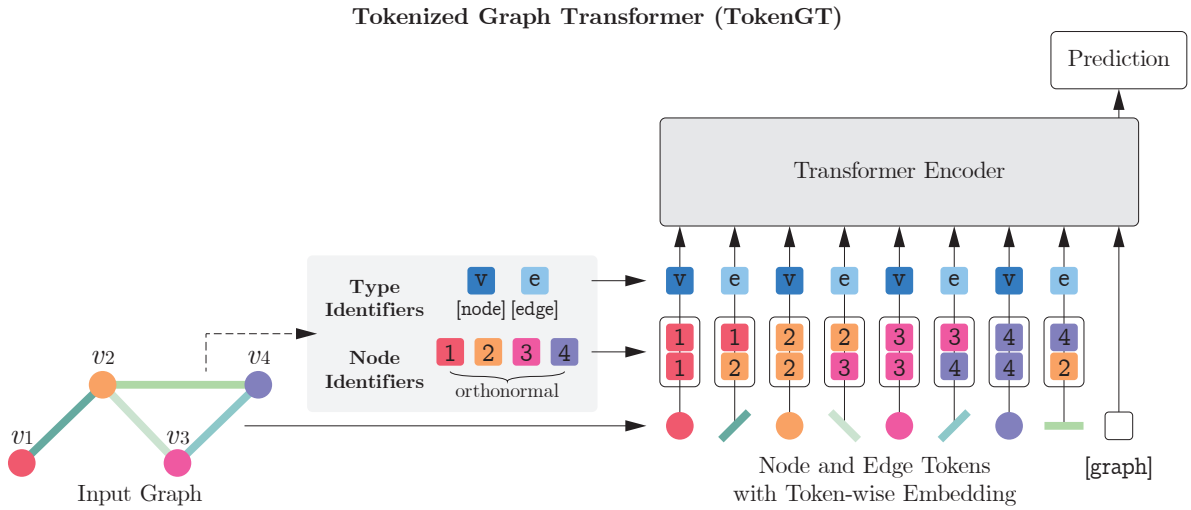


Figure 2.6: Overview of Tokenized Graph Transformer (TokenGT). We treat all nodes and edges of an input graph as independent tokens, augment them with orthonormal node identifiers and trainable type identifiers, and feed them to a standard transformer encoder. For graph-level prediction, we follow the common practice [Devlin et al., 2019, Dosovitskiy et al., 2021] of using a trainable `[graph]` token.

2.3 Tokenized graph transformer

In recent years, transformer [Vaswani et al., 2017] has served as a versatile architecture in a broad class of machine learning problems, such as natural language processing [Devlin et al., 2019, Brown et al., 2020], computer vision [Dosovitskiy et al., 2021], and reinforcement learning [Chen et al., 2021], to name a few. It is because the fully-attentional structure of transformer is general and powerful enough to take, process, and relate inputs and outputs of arbitrary structures, eliminating a need for data- and task-specific inductive bias to be baked into the network architecture. Combined with large-scale training, it opens up a new chapter for building a versatile model that can solve a wide range of problems involving diverse data modalities and even a mixture of modalities [Jaegle et al., 2021b,a, Reed et al., 2022].

In graph learning domain, inspired by the breakthroughs, multiple works tried combining self-attention into graph neural network (GNN) architecture where message passing was previously dominant [Min et al., 2022]. As global self-attention across nodes cannot reflect the graph structure, however, these methods introduce *graph-specific* architectural modifications. This includes restricting self-attention to local neighborhoods [Velickovic et al., 2018, Nguyen et al., 2022, Dwivedi and Bresson, 2020], using global self-attention in conjunction with message-passing GNN [Rong et al., 2020, Lin et al., 2021, Kim et al., 2021a], and injecting edge information into global self-attention via attention bias [Wang et al., 2019c, Ying et al., 2021, Hussain et al., 2021, Park et al., 2022]. Despite decent performance, such modifications can be a limiting constraint in terms of versatility, especially considering future integration to multi-task and multi-modal general-purpose attentional architectures [Jaegle et al., 2021b]. In addition, deviating from pure self-attention, these methods may inherit the issues of message-passing such as oversmoothing [Li et al., 2018, Cai and Wang, 2020, Oono and Suzuki, 2020], and become incompatible with useful engineering techniques e.g., linear attention [Tay et al., 2020] developed for standard self-attention.

Instead, we explore the opposite direction of *applying a standard transformer directly for graphs*. For this, we treat all nodes and edges as independent tokens, augment them with appropriate token-wise

embeddings, and feed the tokens as input to the standard transformer. The model operates identically to transformers used in language and vision; each node or edge is treated as a token, identical to the words in a sentence or patches of an image [Vaswani et al., 2017, Dosovitskiy et al., 2021]. Perhaps surprisingly, we show that this simple approach yields a powerful graph learner both in theory and practice.

As a key result, we prove that with appropriate token-wise embeddings, self-attention over the node and edge tokens can approximate *any* permutation-equivariant linear operator on a graph [Maron et al., 2019b]. Remarkably, we show that a very simple choice of embedding composed of *node identifiers* and *type identifiers* is sufficient for accurate approximation. This provides a solid theoretical guarantee that, with the embeddings and enough attention heads, a transformer is at least as expressive as a second-order invariant graph network (2-IGN) [Maron et al., 2019b, Kim et al., 2021a], which is already more expressive than message-passing GNNs [Gilmer et al., 2017]. This also immediately grants the model with the expressive power at least as good as the 2-dimensional Weisfeiler-Lehman (WL) graph isomorphism test [Maron et al., 2019a], which is often sufficient for real-world graph data [Zopf, 2022]. We further extend our theoretical result to *hypergraphs* with order-k hyperedges, showing that a transformer with order-k generalized token embeddings is at least as expressive as k-IGN and, consequently k-WL test.

We test our model, named tokenized graph transformer (TokenGT), mainly on the PCQM4Mv2 large-scale quantum chemical property prediction dataset containing 3.7M molecular graphs [Hu et al., 2021]. Even though TokenGT involves minimal graph-specific architectural modifications, it performs significantly better than all GNN baselines, showing that the advantages of transformer architecture combined with large-scale training surpass the benefit of hard inductive bias of GNNs. Furthermore, TokenGT achieves competitive performance compared to transformer variants with strong graph-specific modifications [Ying et al., 2021, Hussain et al., 2021, Park et al., 2022]. Finally, we show that TokenGT can naturally utilize efficient approximations in transformers in contrast to these variants, using kernel attention [Choromanski et al., 2021] that enables linear cost without much degradation in performance.

2.3.1 Method

In this section, we present TokenGT, a pure transformer architecture for graphs with token-wise embeddings composed of *node identifiers* and *type identifiers* (Figure 2.6). Our goal in this section is to provide a practical overview. For theoretical analysis, we guide the readers to Section 2.3.2.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ an input graph with n nodes $\mathcal{V} = \{v_1, ..., v_n\}$ and m edges $\mathcal{E} = \{e_1, ..., e_m\} \subseteq \mathcal{V}^2$, associated with features $\mathbf{X}^{\mathcal{V}} \in \mathbb{R}^{n \times C}$ and $\mathbf{X}^{\mathcal{E}} \in \mathbb{R}^{m \times C}$, respectively. We treat each node and edge as an independent token (thus $(n+m)$ tokens in total) and construct their features by $\mathbf{X} = [\mathbf{X}^{\mathcal{V}}; \mathbf{X}^{\mathcal{E}}] \in \mathbb{R}^{(n+m) \times C}$. A naïve way to process a graph is to directly provide the tokens $\mathbf{X}$ as input to a transformer, but it would discard all connectivity. To properly represent graph structure, we augment the tokens $\mathbf{X}$ with token-wise embeddings, more specifically orthonormal *node identifiers* used for representing the connectivity of the tokens and trainable *type identifiers* that encode whether a token is a node or an edge. Despite the simplicity, we show that a transformer applied on these embeddings is a powerful graph learner.

Node identifiers The first component of token-wise embedding is the orthonormal node identifier that we use to represent the connectivity structure given in the input graph.

For a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we first produce n node-wise orthonormal vectors $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ that we refer to as node identifiers. Then, we augment the tokens $\mathbf{X}$ with node identifiers as follows.

- For each node $v \in \mathcal{V}$, we augment the token $\mathbf{X}_v$ as $[\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_v]$.

- For each edge $(u, v) \in \mathcal{E}$, we augment the token $\mathbf{X}_{(u,v)}$ as $[\mathbf{X}_{(u,v)}, \mathbf{P}_u, \mathbf{P}_v]$.

Intuitively, a transformer operating on the augmented tokens can fully recognize the connectivity structure of the graph since comparing the node identifiers between a pair of tokens reveals their *incidence* information. For instance, we can tell if an edge $e = (u, v)$ is connected with a node k through dot-product (attention) since $[\mathbf{P}_u, \mathbf{P}_v][\mathbf{P}_k, \mathbf{P}_k]^\top = 1$ if and only if $k \in (u, v)$ and 0 otherwise. This allows the transformer to identify and exploit the connectivity structure of a graph, for instance by putting more weights on incident pairs when the local operation is important.

Notably, as the node identifiers $\mathbf{P}$ are *only* required to be orthonormal, we have a large degree of freedom in implementation choices. We outline two practical methods below as examples.

- Orthogonal random features (ORFs), e.g., rows of random orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ obtained via QR decomposition of random Gaussian matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ [Yu et al., 2016, Choromanski et al., 2017].

- Laplacian eigenvectors obtained from eigendecomposition of graph Laplacian matrix, i.e., rows of $\mathbf{U}$ from $\mathbf{\Delta} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \mathbf{U}^\top \mathbf{\Lambda} \mathbf{U}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is adjacency matrix, $\mathbf{D}$ is degree matrix, and $\mathbf{\Lambda}$, $\mathbf{U}$ correspond to eigenvalues and eigenvectors respectively [Dwivedi et al., 2020].

Among the two methods, node identifiers generated as ORFs do not encode any information about the graph structure as they are entirely random. This means the transformer that operates on the ORF-based node identifiers needs to compile and recognize graph structure only from the incidence information provided by the node identifiers. Although this is challenging, perhaps surprisingly, we empirically show in Section 2.3.4 that transformers are strong enough to learn meaningful structural representations out of ORF-based node identifiers and outperform GNNs on large-scale task.

In contrast to ORFs, Laplacian eigenvectors provide a kind of graph positional embeddings (graph PEs) that describes the distance between nodes on a graph. Due to the positional information, it yields better performance compared to ORFs in our experiments in Section 2.3.4. One interesting aspect of Laplacian eigenvectors is that they can be viewed as a generalization of sinusoidal positional embeddings of NLP transformers to graphs, as the eigenvectors of 1D chain graphs are sine and cosine functions [Dwivedi et al., 2020]. Thus, by choosing Laplacian eigenvectors as node identifiers, our approach can be interpreted as a direct extension of the NLP transformer for inputs involving relational structures.

Type identifiers The second component of token-wise embedding is the trainable type identifier that encodes whether a token is node or edge. For a given input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we first prepare a trainable parameter matrix $\mathbf{E} = [\mathbf{E}^{\mathcal{V}}; \mathbf{E}^{\mathcal{E}}] \in \mathbb{R}^{2 \times d_e}$ that contains two type identifiers $\mathbf{E}^{\mathcal{V}}$ and $\mathbf{E}^{\mathcal{E}}$ for nodes and edges respectively. Then, we further augment the tokens with type identifiers as follows.

- For each node $v \in \mathcal{V}$, we augment the token $[\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_v]$ as $[\mathbf{X}_v, \mathbf{P}_v, \mathbf{P}_v, \mathbf{E}^{\mathcal{V}}]$.

- For each edge $(u, v) \in \mathcal{E}$, we augment the token $[\mathbf{X}_{(u,v)}, \mathbf{P}_u, \mathbf{P}_v]$ as $[\mathbf{X}_{(u,v)}, \mathbf{P}_u, \mathbf{P}_v, \mathbf{E}^{\mathcal{E}}]$.

These embeddings provide information on whether a given token is a node or an edge, which is critical, e.g., when an attention head tries to attend specifically to node tokens and ignore edge tokens.

Main transformer With node identifiers and type identifiers, we obtain augmented token features $\mathbf{X}^{in} \in \mathbb{R}^{(n+m) \times (C+2d_p+d_e)}$, which is further projected by a trainable matrix $w^{in} \in \mathbb{R}^{(C+2d_p+d_e) \times d}$ to be an input to transformer. For graph-level prediction, we prepend a special token `[graph]` with trainable

embedding $\mathbf{X}_{\texttt{[graph]}} \in \mathbb{R}^d$ similar to BERT [Devlin et al., 2019] and ViT [Dosovitskiy et al., 2021]. We utilize the feature of `[graph]` token at the output of the encoder as the graph representation, on which a linear prediction head is applied to produce the final graph-level prediction. Overall, the tokens $\mathbf{Z}^{(0)} = [\mathbf{X}_{\texttt{[graph]}}; \mathbf{X}^{in} w^{in}] \in \mathbb{R}^{(1+n+m)\times d}$ are used as the input to the main encoder. As an encoder, we adopt the standard transformer [Vaswani et al., 2017], which is an alternating stack of multihead self-attention layers (MSA) and feedforward MLP layers. We provide further details in Appendix A.3.1.

Inductive bias Similar to transformers in language and vision [Devlin et al., 2019, Dosovitskiy et al., 2021], TokenGT treats input nodes and edges as independent tokens and applies self-attention to them. This approach leads to much less inductive bias than current GNNs, where the sparse graph structure, or more fundamentally, *permutation symmetry* of graphs is deliberately baked into each layer [Gilmer et al., 2017, Maron et al., 2019b,a, Kim et al., 2021a]. For TokenGT, such information is provided entirely as a part of input using token-wise embeddings, and the model has to learn how to interpret and utilize the information from data. Although such weak inductive bias might raise questions on the expressiveness of the model, our theoretical analysis in Section 2.3.2 shows that TokenGT is a powerful graph learner thanks to the token-wise embeddings and expressive power of self-attention. For example, we show that TokenGT is more expressive than all message-passing GNNs under the framework of Gilmer et al. [2017].

2.3.2 Theoretical analysis

We now present our theory. Our key result is that TokenGT, a standard transformer with node and type identifiers presented in Section 2.3.1, is provably *at least* as expressive as the second-order invariant graph network (2-IGN [Maron et al., 2019b]), which is built upon *all* possible permutation-equivariant linear layers on a graph. This provides solid guarantees for TokenGT, such as being at least as powerful as the 2-WL graph isomorphism test and more expressive than all message-passing GNNs. Our theory is based on a general framework on hypergraphs represented as higher-order tensors, which leads to the formulation of order-k TokenGT that is at least as expressive as order-k IGN [Maron et al., 2019b].

Preliminary

We provide necessary background before presenting the main results.

Representing and processing sets and (hyper)graphs For a set of n nodes, we often represent their features as $\mathbf{X} \in \mathbb{R}^{n\times d}$ where $\mathbf{X}_i \in \mathbb{R}^d$ is the feature of the i-th node. The set is unordered and, therefore, should be treated invariant to the renumbering of the nodes. Let S_n the symmetric group or the group of permutations π on $[n] = \{1, ..., n\}$. By $\pi \cdot \mathbf{X}$ we denote permuting rows of $\mathbf{X}$ with π, i.e., $(\pi \cdot \mathbf{X})_i = \mathbf{X}_{\pi^{-1}(i)}$. Here, $\mathbf{X}$ and $\pi \cdot \mathbf{X}$ represent the identical set for all $\pi \in S_n$.

Generally, we consider (hyper)graphs represented as order-k tensor $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ with feature $\mathbf{X}_{\mathbf{i}} = \mathbf{X}_{i_1,...,i_k} \in \mathbb{R}^d$ attached to (hyper)edge represented as multi-index $\mathbf{i} = (i_1, ..., i_k) \in [n]^k$. Similar to sets, the tensor should be treated invariant to node renumbering by any $\pi \in S_n$ that acts on $\mathbf{X}$ by $(\pi \cdot \mathbf{X})_{\mathbf{i}} = \mathbf{X}_{\pi^{-1}(\mathbf{i})}$ where $\pi^{-1}(\mathbf{i}) = (\pi^{-1}(i_1), ..., \pi^{-1}(i_k))$. That is, $\mathbf{X}$ and $\pi \cdot \mathbf{X}$ represent the identical (hyper)graph for all π. Due to such symmetry, to build a function $F(\mathbf{X}) \approx T$ for tensor $\mathbf{X}$ and target T, a suitable way is to make them *invariant* $F(\pi \cdot \mathbf{X}) = F(\mathbf{X})$ when the target is a vector or *equivariant* $F(\pi \cdot \mathbf{X}) = \pi \cdot F(\mathbf{X})$ when the target is also a tensor, for all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and $\pi \in S_n$.

In our theoretical analysis, we work on order-k dense tensor representation $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ of a graph as they can represent node features ($k = 1$), edge features ($k = 2$), or hyperedge features ($k > 2$) in a

unified manner. This is interchangeable but slightly different from the sparse representation of a graph with edge set $\mathcal{E}$ used in Section 2.3.1. Nevertheless, in Section 2.3.4 we empirically verify that our key theoretical findings work equally well for dense and sparse graphs.

Invariant graph network We mainly develop our theoretical analysis upon *Invariant Graph Networks (IGNs)* [Maron et al., 2019a,b], a family of expressive graph networks derived from the permutation symmetry of tensor representation of graphs. Here we provide a summary. In general, we define:

Definition 4. *An order-k Invariant Graph Network (k-IGN) is a function* $F_k : \mathbb{R}^{n^k \times d_0} \to \mathbb{R}$ *written as:*

$$F_k = MLP \circ L_{k\to 0} \circ L^{(T)}_{k\to k} \circ \sigma \circ ... \circ \sigma \circ L^{(1)}_{k\to k},$$

where each $L^{(t)}_{k\to k}$ *is equivariant linear layer [Maron et al., 2019b] from* $\mathbb{R}^{n^k \times d_{t-1}}$ *to* $\mathbb{R}^{n^k \times d_t}$*,* σ *is activation function, and* $L_{k\to 0}$ *is a invariant linear layer from* $\mathbb{R}^{n^k \times d_T}$ *to* $\mathbb{R}$*.*

A body of previous work have shown appealing properties of k-IGN, including universal approximation [Maron et al., 2019c] and alignment to k-Weisfeiler-Lehman (k-WL) graph isomorphism test [Maron et al., 2019a, Chen et al., 2020b]. In particular, it is known that k-IGNs are at least as powerful as the k-WL test [Maron et al., 2019a]. It is also known that 2-IGNs are already more expressive [Maron et al., 2019b, Kim et al., 2021a] than all message-passing GNNs under the framework of Gilmer et al. [2017].

The core building block of IGN is invariant and equivariant *linear* layers [Maron et al., 2019b] with maximal expressiveness while respecting node permutation symmetry. The layers are defined as follows:

Definition 5. *An equivariant linear layer is a function* $L_{k\to l} : \mathbb{R}^{n^k \times d} \to \mathbb{R}^{n^l \times d'}$ *written as follows for order-k input* $\mathbf{X} \in \mathbb{R}^{n^k \times d}$*:*

$$L_{k\to l}(\mathbf{X})_{\mathbf{i}} = \sum_{\mu} \sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{X}_{\mathbf{j}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{i}} b_{\lambda}, \tag{2.26}$$

where $\mathbf{i} \in [n]^l, \mathbf{j} \in [n]^k$ *are multi-indices,* $w_\mu \in \mathbb{R}^{d\times d'}$*,* $b_\lambda \in \mathbb{R}^{d'}$ *are weight and bias parameters, and* $\mathbf{B}^\mu \in \mathbb{R}^{n^{l+k}}$ *and* $\mathbf{C}^\lambda \in \mathbb{R}^{n^l}$ *are binary basis tensors corresponding to order-*$(l+k)$ *and order-*l *equivalence classes* μ *and* λ*, respectively. Invariant linear layer is a special case of* $L_{k\to l}$ *with* $l = 0$*.*

We provide the definition of the equivalence classes and basis tensors in Appendix A.3.1. For now, it is sufficient to know that the basis tensors are binary tensors that form the orthogonal basis of the full space of linear equivariant layers. In general, in Equation (2.26) it is known that there exists bell$(k + l)$ number of basis tensors $\mathbf{B}^\mu$ for the weight and bell(l) number of basis tensors $\mathbf{C}^\lambda$ for the bias.

Can self-attention approximate equivariant basis?

Now, we present an intuition that connects transformer (Section 2.3.1) and equivariant linear layer (Definition 5). For that, we write out the multihead self-attention layer as follows:

$$\text{MSA}(\mathbf{X})_i = \sum_{h=1}^{H} \sum_{j} \boldsymbol{\alpha}^h_{ij} \mathbf{X}_j w^V_h w^O_h \text{ where } \boldsymbol{\alpha}^h = \text{softmax}\left(\frac{\mathbf{X} w^Q_h (\mathbf{X} w^K_h)^\top}{\sqrt{d_H}} \right), \tag{2.27}$$

where H is number of heads, d_H is head size, and $w^Q_h, w^K_h \in \mathbb{R}^{d\times d_H}$, $w^V_h \in \mathbb{R}^{d\times d_v}$ $w^O_h \in \mathbb{R}^{d_v \times d}$.

Our intuition is that the weighted sum of values with self-attention matrix $\boldsymbol{\alpha}^h$ in Equation (2.27) is analogous to the masked sum with basis tensor $\mathbf{B}^\mu$ in Equation (2.26) up to normalization. This

naturally leads to the following question: for a given equivariant layer $L_{k\to k} : \mathbb{R}^{n^k\times d} \to \mathbb{R}^{n^k\times d}$, can we use a transformer layer with multihead self-attention $\mathrm{MSA} : \mathbb{R}^{N\times d'} \to \mathbb{R}^{N\times d'}$ with $N = n^k$ to accurately approximate $L_{k\to k}$ by having $H = \mathrm{bell}(2k)$ attention heads approximate each equivariant basis $\mathbf{B}^\mu$?

We show that this can be possible, but only if we provide appropriate auxiliary information to input. For example, let us consider first-order layer $L_{1\to 1}$. The layer has $\mathrm{bell}(2) = 2$ basis tensors $\mathbf{B}^{\mu_1} = \mathbf{I}$ and $\mathbf{B}^{\mu_2} = \mathbf{1}\mathbf{1}^\top - \mathbf{I}$ for the weight, and $\mathrm{bell}(1) = 1$ basis tensor $\mathbf{C}^{\lambda_1} = \mathbf{1}$ for the bias. Given an input set $\mathbf{X} \in \mathbb{R}^{n\times d}$ it computes the following with $w_1, w_2 \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$:

$$L_{1\to 1}(\mathbf{X}) = \mathbf{I}\mathbf{X}w_1 + (\mathbf{1}\mathbf{1}^\top - \mathbf{I})\mathbf{X}w_2 + \mathbf{1}b^\top.$$

Now consider approximating basis tensor $\mathbf{B}^{\mu_1} = \mathbf{I}$ with an attention matrix $\boldsymbol{\alpha}^1$. The approximation is accurate when i-th query always only attends to i-th key and ignores the rest. To achieve the attention structure consistently, i.e., agnostic to input $\mathbf{X}$, we need to provide auxiliary input that self-attention can "latch onto" to faithfully approximate $\boldsymbol{\alpha}^1 \approx \mathbf{I}$. Without this, attention must entirely rely on the inputs $\mathbf{X}$, which is unreliable and can lead to approximation failure, e.g., when $\mathbf{X}$ has repeated rows.

For the auxiliary information, we prepare n node-wise orthonormal vectors $\mathbf{P} \in \mathbb{R}^{n\times d_p}$ (note that this is identical to node identifiers in Section 2.3.1), and augment the input to $\mathbf{X}^{in} = [\mathbf{X}, \mathbf{P}] \in \mathbb{R}^{n\times(d+d_p)}$. Let us assume that the query and key projections in Equation (2.27) ignore $\mathbf{X}$ and only leave $\mathbf{P}$ scaled by $\sqrt{a}$ with $a > 0$. Then attention matrix is computed as $\boldsymbol{\alpha}^1 = \mathrm{softmax}(\mathbf{S})$ where $\mathbf{S}_{ij} = a\mathbf{P}_i^\top\mathbf{P}_j$. Here, due to the orthonormality of $\mathbf{P}$, we have $\mathbf{P}_i^\top\mathbf{P}_j = 1$ only if $i = j$ and otherwise 0, which leads to $\mathbf{S} = a\mathbf{I}$. With $a \to \infty$ by scaling up the query and key projection weights, the softmax becomes arbitrarily close to the hardmax operator, and we obtain the following:

$$\boldsymbol{\alpha}^1 = \mathrm{softmax}(a\mathbf{I}) \to \mathbf{I} \qquad \text{as} \qquad a \to \infty.$$

Thus, self-attention can utilize the auxiliary information $\mathbf{P}$ to achieve an input-agnostic approximation of $\boldsymbol{\alpha}^1$ to $\mathbf{I}$. Notably, we can achieve a similar approximation for $\mathbf{B}^{\mu_2} = \mathbf{1}\mathbf{1}^\top - \mathbf{I}$ using the *same* $\mathbf{P}$ by flipping the sign of keys, which gives $\boldsymbol{\alpha}^2 = \mathrm{softmax}(-a\mathbf{I})$ due to orthonormality. By sending $a \to \infty$, now attention from the i-th query to the i-th key is suppressed, and we obtain the following:

$$\boldsymbol{\alpha}^2 = \mathrm{softmax}\left(-a\mathbf{I}\right) \to \frac{1}{n-1}(\mathbf{1}\mathbf{1}^\top - \mathbf{I}) \qquad \text{as} \qquad a \to \infty.$$

Note that this approximation is accurate only up to row normalization as rows of $\boldsymbol{\alpha}^2$ always sum to one due to softmax, while $\mathbf{B}^{\mu_2} = \mathbf{1}\mathbf{1}^\top - \mathbf{I}$ is binary. In our proofs of the theoretical results, we perform appropriate denormalization with MLP after MSA to achieve an accurate approximation.

Overall, we see that simple auxiliary input $\mathbf{P}$ suffices for two attention heads to approximate the equivariant basis of $L_{1\to 1}$ accurately. We now question the following. Given appropriate auxiliary information as input, can a transformer layer with $\mathrm{bell}(2k)$ attention heads accurately approximate $L_{k\to k}$ by having each head approximate each equivariant basis $\mathbf{B}^\mu$? What would be the sufficient auxiliary input? We answer the question by showing that, with (order-k generalized) node and type identifiers presented in Section 2.3.1, transformer layers can accurately approximate equivariant layers $L_{k\to k}$ via input-agnostic head-wise approximation of each equivariant basis.

Pure transformers are powerful graph learners

We now present our main theoretical results that extend the aforementioned discussions to any order k. Note that $k = 2$ corresponds to TokenGT for graphs presented in Section 2.3.1. With $k > 2$, we naturally extend TokenGT to hypergraphs. All proofs can be found in Appendix A.3.2.

We first introduce generalized node and type identifiers (Section 2.3.1) for order-k tensors $\mathbf{X} \in \mathbb{R}^{n^k \times d}$. We define the node identifier $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ as an orthonormal matrix with n rows, and the type identifier as a trainable matrix $\mathbf{E} \in \mathbb{R}^{\text{bell}(k) \times d_e}$ that contains bell(k) rows $\mathbf{E}^{\gamma_1}, ..., \mathbf{E}^{\gamma_{\text{bell}(k)}}$, each of which is designated for an order-k equivalence class γ. Then, we augment each entry of input tensor as $[\mathbf{X}_{i_1,...,i_k}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma}]$ where $(i_1, ..., i_k) \in \gamma$.

Let us exemplify. For $k = 1$ (sets), each i-th entry is augmented as $[\mathbf{X}_i, \mathbf{P}_i, \mathbf{E}^{\gamma_1}]$, consistent with our discussion for the graph case. For $k = 2$ (graphs), each (i, i)-th entry is augmented as $[\mathbf{X}_{ii}, \mathbf{P}_i, \mathbf{P}_i, \mathbf{E}^{\gamma_1}]$ and each (i, j)-th entry $(i \neq j)$ is augmented as $[\mathbf{X}_{ij}, \mathbf{P}_i, \mathbf{P}_j, \mathbf{E}^{\gamma_2}]$. This is consistent with TokenGT in Section 2.3.1, which augments nodes with $\mathbf{E}^{\mathcal{V}} = \mathbf{E}^{\gamma_1}$ and edges with $\mathbf{E}^{\mathcal{E}} = \mathbf{E}^{\gamma_2}$.

With node and type identifiers, we obtain augmented order-k tensor $\mathbf{X}^{in} \in \mathbb{R}^{n^k \times (d + kd_p + d_e)}$. We use a trainable projection $w^{in} \in \mathbb{R}^{(d+kd_p+d_e) \times d_{\mathcal{T}}}$ to map them to hidden dimension $d_{\mathcal{T}}$ of a transformer. We now show that self-attention on $\mathbf{X}^{in} w^{in}$ can accurately approximate equivariant basis:

Lemma 1. *For all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and their augmentation $\mathbf{X}^{in}$, self-attention coefficients $\boldsymbol{\alpha}^h$ (Equation (2.27)) computed with $\mathbf{X}^{in} w^{in}$ can approximate any basis tensor $\mathbf{B}^{\mu} \in \mathbb{R}^{n^{2k}}$ of order-k equivariant linear layer $L_{k \to k}$ (Definition 5) to arbitrary precision up to normalization.*

Consequently, with the node and type identifiers, a collection of bell$(2k)$ attention heads can approximate the collection of all basis tensors of order-k equivariant layer. This leads to the following:

Theorem 7. *For all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and their augmentation $\mathbf{X}^{in}$, a transformer layer with bell$(2k)$ self-attention heads that operates on $\mathbf{X}^{in} w^{in}$ can approximate an order-k equivariant linear layer $L_{k \to k}(\mathbf{X})$ (Definition 5) to arbitrary precision.*

While the approximation in Lemma 1 is only accurate up to normalization over inputs (keys) due to softmax normalization, for the approximation in Theorem 7 we perform appropriate denormalization using MLP after multihead self-attention and can obtain an accurate approximation.

By extending the result to multiple layers, we arrive at the following:

Theorem 8. *For all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and their augmentation $\mathbf{X}^{in}$, a transformer composed of T layers that operates on $\mathbf{X}^{in} w^{in}$ followed by sum-pooling and MLP can approximate an k-IGN $F_k(\mathbf{X})$ (Definition 4) to arbitrary precision.*

This directly leads to the following corollary:

Corollary 3. *A transformer on node and type identifiers in Theorem 8 is at least as expressive as k-IGN composed of order-k equivariant linear layers.*

Corollary 3 allows us to draw previous theoretical results on the expressiveness of k-IGN [Maron et al., 2019a,b, Kim et al., 2021a] and use them to lower-bound the provable expressiveness of a standard transformer:

Corollary 4. *A transformer on node and type identifiers in Theorem 8 is at least as powerful as k-WL graph isomorphism test and is more expressive than all message-passing GNNs within the framework of Gilmer et al. [2017].*

2.3.3 Related work

Equivariant neural networks A machine learning task is often invariant or equivariant to specific symmetry of input data, e.g., image classification is invariant to the translation of an input image. A large body of literature advocated baking the invariance or equivariance into a neural network as a type of inductive bias (e.g., translation equivariance of image convolution), showing that it reduces the number of parameters and improves generalization for a wide range of learning tasks involving various geometric structures [Cohen and Welling, 2016, 2017, Weiler et al., 2018, Thomas et al., 2018, Pan and Kondor, 2022, Bronstein et al., 2017]. Ravanbakhsh et al. [2017] showed that any equivariant layer for discrete group actions is equivalent to a specific parameter sharing. Zaheer et al. [2017], Maron et al. [2019b] derived the parameter sharing for node permutation-symmetric data (sets and (hyper)graphs), which gives the maximally expressive equivariant linear layers and k-IGN. The work on equivariant neural networks underlie our theory of how a standard transformer can be a powerful learner for sets and (hyper)graphs.

Expressive power of transformers and its connection to equivariance Recent work on transformers often focus on minimizing the domain- and task-specific inductive bias and scaling the model and data so that any useful computation structure can be learned [Dosovitskiy et al., 2021, Jaegle et al., 2021b,a, Brown et al., 2020, Devlin et al., 2019, Chen et al., 2021]. The success of this approach is, to some degree, attributed to the high expressivity of transformers that allows learning diverse functions suited for the data at hand [Yun et al., 2020, Bhattamishra et al., 2020, Bhojanapalli et al., 2020, Likhosherstov et al., 2021]. Recent theory has shown that transformers are expressive enough to model certain equivariant functions. Andreoli [2019] cast self-attention and convolution into a unified framework using basis tensors. Cordonnier et al. [2020] showed that transformers with relative positional encodings can approximate any image convolution layers. Lee et al. [2019], Kim et al. [2021a] showed that transformers can model equivariant linear layers for sets, which can be viewed as the first-order case of our theory. To our knowledge, our work is the first to show that standard transformers are expressive enough to provably model maximally expressive equivariant layers and k-IGN for (hyper)graphs with $k \geq 2$.

Transformers for graphs Unlike language and vision, developing transformers for graphs is challenging due to **(1)** the presence of edge connectivity and **(2)** the absence of canonical node ordering that prevents adopting simple positional encodings [Min et al., 2022]. To incorporate the connectivity, early methods restricted self-attention to local neighborhoods (thus reducing to message passing) [Dwivedi and Bresson, 2020, Nguyen et al., 2022, Velickovic et al., 2018] or used global self-attention with auxiliary message passing [Rong et al., 2020, Lin et al., 2021]. As message passing suffers from limited expressivity and oversmoothing, recent works often discard them and use global self-attention on nodes with heuristic modifications to process edges. Ying et al. [2021], Park et al. [2022] proposed to inject edge encoding based on shortest paths via attention bias. Kreuzer et al. [2021] proposed to incorporate edges into attention matrix via elementwise multiplication. We leave the self-attention unmodified and provide both nodes and edges with token-wise embeddings in Section 2.3.1 as its input. To inject graph structure into nodes, some works focus on developing graph positional encoding [Dwivedi et al., 2020, Lim et al., 2022, Kreuzer et al., 2021]. These can be directly incorporated into our work via auxiliary node identifiers, and hence are complementary. A work closely related to us is higher-order transformer [Kim et al., 2021a] that generalizes k-IGN with masked attention. While it is complex to implement due to hard-coded per-head masks, our method can be implemented effortlessly using any available transformer. Also, our model can flexibly use different heads to focus on a specific equivariant operator (e.g., local propagation) if needed.

Table 2.11: Second-order equivariant basis approximation. We report average and standard deviation of L2 error averaged over heads over 3 runs. For Random/ORF (first-order), we sample random embeddings independently for each token.

node id.	type id.	dense input train L2 ↓	dense input test L2 ↓	sparse input train L2 ↓	sparse input test L2 ↓
×	×	47.95 ± 0.600	53.93 ± 1.426	29.88 ± 0.450	34.70 ± 1.167
×	◯	32.38 ± 0.448	40.06 ± 1.202	15.92 ± 0.275	20.39 ± 0.765
Random (first-order)	◯	32.19 ± 0.476	32.49 ± 3.687	15.87 ± 0.247	16.56 ± 0.904
ORF (first-order)	◯	32.35 ± 0.369	39.87 ± 1.263	15.87 ± 0.247	16.56 ± 0.908
Random	×	5.909 ± 0.019	5.548 ± 0.090	8.152 ± 0.042	8.270 ± 0.285
ORF	×	5.472 ± 0.035	5.143 ± 0.078	7.167 ± 0.025	7.190 ± 0.217
Laplacian eigenvector	×	1.899 ± 3.050	1.702 ± 2.912	0.288 ± 0.019	0.064 ± 0.010
Random	◯	0.375 ± 0.009	0.234 ± 0.011	0.990 ± 0.108	0.875 ± 0.042
ORF	◯	0.080 ± 0.001	0.009 ± 5e-5	0.129 ± 0.002	**0.011 ± 0.002**
Laplacian eigenvector	◯	**0.053 ± 1.5e-5**	**0.005 ± 1e-4**	**0.101 ± 0.003**	0.019 ± 0.007

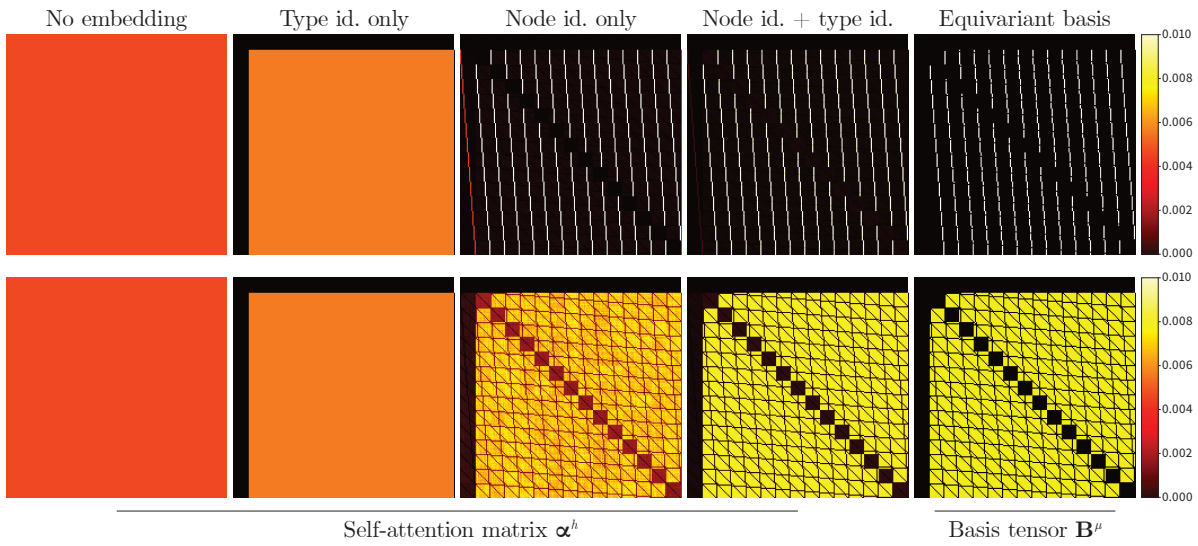


Figure 2.7: Self-attention maps learned under various node and type identifier configurations for two target equivariant basis tensors (out of 15). For better visualization, we clamp the entries by 0.01. Self-attention learns acute patterns coherent to equivariant basis when orthonormal node identifiers and type identifiers are both provided as input.

2.3.4 Experiments

We first conduct a synthetic experiment that directly confirms our key claims in Lemma 1. Then, we empirically explore the capability of TokenGT using the PCQM4Mv2 large-scale quantum chemistry regression dataset [Hu et al., 2020a].

Approximating second-order equivariant basis

As in Theorem 7 and 8, our argument on the expressive power of TokenGT relies on its capability to approximate order-k permutation-equivariant linear layers $L_{k\to k}$ (Definition 5). Specifically, Lemma 1 states that such capability depends on the ability of each self-attention head $\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^H$ (Equation (2.27)) to accurately approximate each equivariant basis $\mathbf{B}^{\mu_1}, ..., \mathbf{B}^{\mu_{\mathrm{bell}(2k)}}$ (Definition 5) up to normalization.

We verify this claim for $k = 2$ (second-order; graphs) in a synthetic setup using Barabási-Albert

random graphs. We use a multihead self-attention layer (Equation (2.27)) with $\text{bell}(2+2) = 15$ heads and explicitly supervise head-wise attention scores $\boldsymbol{\alpha}^h$ to approximate each (normalized) equivariant basis tensor $\mathbf{B}^{\mu_h}$ by minimizing L2 loss. Having the layer hyperparameters fixed, we provide different combinations of node and type identifiers, and test if multihead self-attention can jointly approximate *all* 15 equivariant basis on unseen graphs. We experiment with both dense and sparse graph representations; for graphs with n nodes and m edges, the dense graph considers all n^2 pair-wise edges as input as in Section 2.3.2, whereas the sparse graph considers only the present m edges as in Section 2.3.1.

We outline the results in Table 2.11. Consistent with Lemma 1, self-attention achieves accurate approximation of equivariant basis only when both the orthonormal node identifiers and type identifiers are given. Here, Laplacian eigenvectors (Lap, ◯) often yield slightly better results than orthogonal random features (ORF, ◯) presumably due to less stochasticity. Interestingly, we see that self-attention transfers the learned (pseudo-)equivariant self-attention structure to unseen graphs near perfectly. Non-orthogonal random embeddings lead to inaccurate approximation (Random, ◯), highlighting the importance of orthogonality of node identifiers. The approximation is also inaccurate when we sample ORF $\mathbf{P}_t$ independently for each token t (ORF (first-order), ◯) instead of using concatenated node identifiers $[\mathbf{P}_u, \mathbf{P}_v]$ for token (u, v). This supports our argument in Section 2.3.1 that the incidence information implicitly provided via node identifiers plays a key role in approximation. In Figure 2.7, we provide a visualization of self-attention maps learned under various node and type identifier choices.

Large-scale graph learning

An exclusive characteristic of TokenGT is its minimal graph-specific inductive bias, which requires it to learn internal computation structure largely from data. As such models are commonly known to work well with large-scale data [Vaswani et al., 2017, Dosovitskiy et al., 2021], we explore the capability of TokenGT on the PCQM4Mv2 quantum chemistry regression dataset [Hu et al., 2021], one of the current largest with 3.7M molecular graphs.

For TokenGT, we use both node and type identifiers, and use main transformer encoder configuration based on Graphormer [Ying et al., 2021] with 12 layers, 768 hidden dimension, and 32 attention heads. We try both ORF and Laplacian eigenvector as node identifiers, and denote corresponding models as **TokenGT (ORF)** and **TokenGT (Lap)** respectively. As an ablation, we also experiment with the same transformer without node and type identifiers, which we denote as **transformer**. Finally, we apply the kernel attention [Choromanski et al., 2021] that approximates the attention computation to linear cost (**TokenGT (Lap) + Performer**). We use AdamW optimizer with $(\beta_1, \beta_2) = (0.99, 0.999)$ and weight decay 0.1, and 60k learning rate warmup steps followed by linear decay over 1M iteration with batch size 1024. For fine-tuning, we use 1k warmup, 0.1M training steps, and cosine learning rate decay. We train the models on 8 RTX 3090 GPUs for 3 days.

The results are in Table 2.12. A standard transformer on the node and edge tokens cannot recognize graph structure and shows low performance (0.2340 valid MAE). Yet, the picture changes as soon as we augment the tokens with node and type identifiers. Notably, TokenGT (ORF) achieves 0.0962 MAE, which is already better than all GNN baselines. This is a somewhat surprising result, as both ORF and the transformer are not aware of graph structures. This implies transformer is strong enough to learn to interpret and reason over the incidence structure of tokens provided only implicitly by the node and type identifiers. By further switching to Laplacian eigenvectors that encode position on graphs [Dwivedi et al., 2020], we observe a performance boost to 0.0910 MAE, competitive to transformers with sophisticated graph-specific modifications (e.g., shortest path-based spatial encoding [Ying et al., 2021]). While such

Table 2.12: Results on PCQM4Mv2 large-scale graph regression benchmark. We report the Mean Absolute Error (MAE) on the validation set, and report MAE on the unavailable test set if possible.

	# parameters	valid MAE ↓	test-dev MAE ↓	asymptotics
Message-passing GNNs				
GCN [Hu et al., 2021]	2.0M	0.1379	0.1398	$\mathcal{O}(n+m)$
GIN [Hu et al., 2021]	3.8M	0.1195	0.1218	$\mathcal{O}(n+m)$
GAT	6.7M	0.1302	N/A	$\mathcal{O}(n+m)$
GCN-VN [Hu et al., 2021]	4.9M	0.1153	0.1152	$\mathcal{O}(n+m)$
GIN-VN [Hu et al., 2021]	6.7M	0.1083	0.1084	$\mathcal{O}(n+m)$
GAT-VN	6.7M	0.1192	N/A	$\mathcal{O}(n+m)$
GAT-VN (large)	55.2M	0.1361	N/A	$\mathcal{O}(n+m)$
Transformers with strong graph-specific modifications				
Graphormer [Shi et al., 2022]	48.3M	0.0864	N/A	$\mathcal{O}(n^2)$
EGT [Hussain et al., 2021]	89.3M	0.0869	0.0872	$\mathcal{O}(n^2)$
GRPE [Park et al., 2022]	46.2M	0.0890	0.0898	$\mathcal{O}(n^2)$
Pure transformers				
Transformer	48.5M	0.2340	N/A	$\mathcal{O}((n+m)^2)$
TokenGT (ORF)	48.6M	0.0962	N/A	$\mathcal{O}((n+m)^2)$
TokenGT (Lap)	48.5M	0.0910	0.0919	$\mathcal{O}((n+m)^2)$
TokenGT (Lap) + Performer	48.5M	0.0935	N/A	$\mathcal{O}(n+m)$

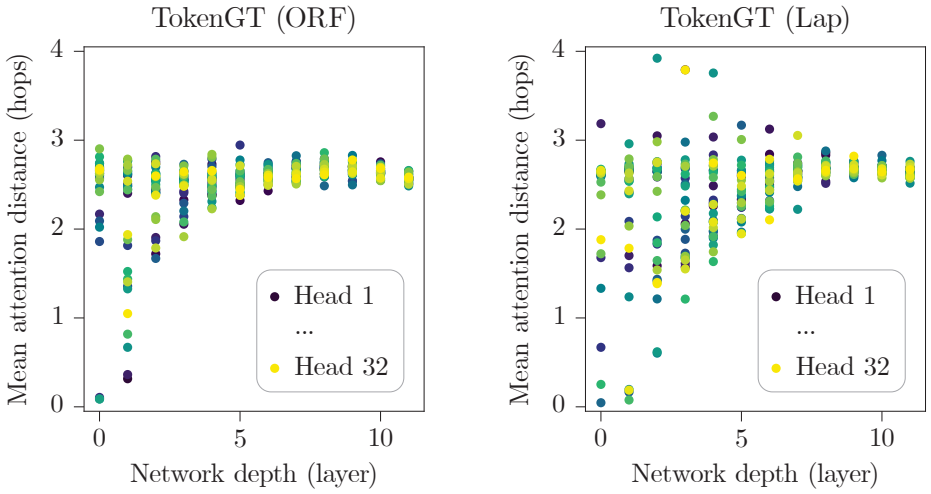


Figure 2.8: Attention distance by head and network depth. Each dot shows mean attention distance in hops across graphs of a head at a layer. The visualization is inspired by Dosovitskiy et al. [2021].

methods inject graph structure into attention matrix via bias term and therefore strictly require $\mathcal{O}(n^2)$ cost, TokenGT enables adopting kernelization for pure self-attention [Choromanski et al., 2021], resulting in TokenGT (Lap) + Performer with the best performance among $\mathcal{O}(n+m)$ models (0.0935 MAE).

While our theory in Section 2.3.2 *guarantees* that TokenGT can reduce to an equivariant layer by learning fixed equivariant basis at each attention head, in practice, it can freely utilize multihead self-attention to learn less restricted and more useful computation structure from data. To analyze such a structure, we compute the attention distance across heads and network depth by averaging pairwise token distances on a graph weighted by their attention scores (Figure 2.8). This distance is analogous to the number of hops in message-passing. In both TokenGT (ORF) and TokenGT (Lap), in the lowest layers, some heads attend globally over the graph while others consistently have small receptive fields (acting like a local message-passing operator). In deeper layers, the attention distances increase, and

most heads attend globally. Interestingly, this behavior is highly consistent with vision transformers on image patches [Dosovitskiy et al., 2021], suggesting that hybrid architectures based on convolution to aid ViT [Dai et al., 2021, Yuan et al., 2021] might also work well for graphs. While TokenGT (ORF) shows relatively consistent attention distance over heads, TokenGT (Lap) shows higher variance, implying that it learns more diverse attention patterns. Judging from the higher performance of TokenGT (Lap), this suggests that the graph structure information of the Laplacian eigenvector facilitates learning useful and diverse attention structures, which calls for future exploration of better node identifiers based on graph PEs [Kreuzer et al., 2021, Lim et al., 2022].

2.3.5 Discussion

We showed that transformers directly applied to graphs can work well in both theory and practice. In the theoretical aspect, we proved that with appropriate token-wise embeddings, a transformer on node and edge tokens is at least as expressive as k-IGN and k-WL test, making it more expressive than all message-passing GNNs. For such token-wise embeddings, we showed that a combination of simple orthonormal node identifiers and trainable type identifiers suffices, which we also verified with a synthetic experiment. In an experiment with PCQM4Mv2 large-scale dataset, we show that TokenGT performs significantly better than all GNNs and is competitive with transformer variants with strong graph-specific architectural components.

While the results suggest a promising research direction, there are challenges to be addressed in future work. First, treating each node and edge as tokens requires $\mathcal{O}((n+m)^2)$ asymptotic cost due to the quadratic nature of self-attention. While we address this to some degree with kernelization and achieve $\mathcal{O}(n+m)$ cost, other types of efficient transformers (e.g., sparse) that can deliver better performance are left to be tested. Another issue is slightly lower performance compared to the state-of-the-art. Adopting transformer engineering techniques from vision and language domains, such as data scaling [Brown et al., 2020, Dosovitskiy et al., 2021], deepening [Wang et al., 2022, Wies et al., 2021], hybrid architectures [Dai et al., 2021, Yuan et al., 2021], and self-supervision [Devlin et al., 2019, Brown et al., 2020, He et al., 2021a], are promising. In the societal aspect, to prevent the potential risky behavior in, e.g., decision making from graph-structured inputs, interpretability research regarding self-attention on graphs is desired.

We finish with interesting research directions that stem from our work. As our approach advocates viewing a graph as $(n+m)$ tokens [Kolter et al., 2019], it opens up new paradigms of graph learning, including autoregressive decoding, in-context learning, prompting, and multimodal learning. Another interesting direction is to extend our theory and use self-attention to approximate equivariant basis for general discrete group actions, which might be a viable approach for *learning equivariance from data*.

Chapter 3. Probabilistic Symmetrizations

The central question of this thesis is how to decouple symmetry from neural architecture. While the previous chapter explored how transformers can be adapted to permutation-symmetric data, in this chapter, we introduce a broader framework applicable to any base model and symmetry group. The approach is based on symmetrization, which makes a function equivariant by averaging its outputs over group transformations of the input. While theoretically powerful, its exact computation is problematic for large or continuous groups. To overcome this limitation, we develop probabilistic symmetrization, a method that designs or learns an input-conditional distribution over the symmetry group to perform sample-efficient symmetrization. This allows pre-trained foundation models to be applied to geometric tasks by externally endowing them with symmetry, rather than hard-coding it into the architecture.

This chapter is adapted from the following papers:

- "Learning Probabilistic Symmetrization for Architecture Agnostic Equivariance", which originally appeared at NeurIPS 2023 and is joint work with Tien Dat Nguyen, Ayhan Suleymanzade, Hyeokjun An, and Seunghoon Hong,
- "Learning Symmetrization for Equivariance with Orbit Distance Minimization", which originally appeared at the NeurIPS 2023 Workshop on Symmetry and Geometry in Neural Representations and is joint work with Tien Dat Nguyen, Hongseok Yang, and Seunghoon Hong,
- "Inverting Data Transformations via Diffusion Sampling", which is currently under review and is joint work with Sékou-Oumar Kaba, Jiyun Park, Seunghoon Hong, and Siamak Ravanbakhsh.

The first work introduces the core framework for a range of matrix groups, the second extends it to more general groups via a novel optimization objective, and the third develops a diffusion-based method applicable to arbitrary continuous groups. In Section 3.1, we present probabilistic symmetrization, show that it achieves equivariance in expectation and is capable of universal approximation, and demonstrate its effectiveness by transferring a pre-trained vision model to graph problems (permutation symmetry) and using a sequence transformer to model particle dynamics (permutation and Euclidean symmetry). In Section 3.2, we address the challenge of symmetrization for more general groups whose elements are difficult to parameterize; we introduce orbit distance minimization, a differentiable objective based on orbit-separating invariants that extends our framework to groups such as the Lorentz group. In Section 3.3, we propose transformation-inverting energy diffusion, a symmetrization method that uses diffusion sampling on arbitrary Lie groups, enabling training-free warp-invariant vision (affine and projective symmetries) and neural PDE solving under unseen initial conditions (Lie point symmetry).

3.1 Probabilistic symmetrization

Many perception problems in machine learning involve functions that are invariant or equivariant to certain symmetry group of transformations of data. Examples include learning on sets and graphs, point clouds, molecules, proteins, and physical data, to name a few [Bogatskiy et al., 2022, Bronstein et al., 2021]. Equivariant architecture design has emerged as a successful approach, where every building block of a model is carefully restricted to be equivariant to a symmetry group of interest [Battaglia et al., 2018, Bronstein et al., 2021, Deng et al., 2021]. However, equivariant architecture design faces fundamental limitations, as individual construction of models for each group can be laborious or computationally expensive [Thomas et al., 2018, Maron et al., 2019c, Morris et al., 2019b, Kim et al., 2021a], the architectural restrictions often lead to limited expressive power [Xu et al., 2019, Maron et al., 2019a, Zhang et al., 2023b, Joshi et al., 2023], and the knowledge learned from one problem cannot be easily transferred to others of different symmetries as the architecture would be incompatible.

This motivates us to seek a **symmetrization** solution that can achieve group invariance and equivariance with general-purpose, group-agnostic architectures such as an MLP or a transformer. As a basic form of symmetrization, any parameterized function $f_\theta : \mathcal{X} \to \mathcal{Y}$ on vector spaces $\mathcal{X}$, $\mathcal{Y}$ can be made invariant or equivariant by group averaging [Yarotsky, 2022, Murphy et al., 2019a], i.e., averaging over all possible transformations of inputs $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$ by a symmetry group $G = \{g\}$:

$$\phi_\theta(\mathbf{x}) = \frac{1}{|G|} \sum_{g \in G} g \cdot f_\theta(g^{-1} \cdot \mathbf{x}), \tag{3.1}$$

where ϕ_θ is equivariant or invariant to the group G. An important advantage is that the symmetrized function ϕ_θ can leverage the expressive power of the base function f_θ; it has been shown that ϕ_θ is a universal approximator of invariant or equivariant functions if f_θ is a universal approximator [Yarotsky, 2022], which includes an MLP [Hornik et al., 1989] or a transformer [Yun et al., 2020]. On the other hand, an immediate challenge is that for many practical groups involving permutation and rotations, the cardinality of the group $|G|$ is large or infinite, so the exact averaging is intractable. Due to this, existing symmetrization approaches often focus on small finite groups [Basu et al., 2022, Mouli and Ribeiro, 2021, van der Pol et al., 2020, Kicki et al., 2021], manually derive smaller subsets of the entire group e.g., a frame [Puny et al., 2022] to average over [Murphy et al., 2019a,b], or implement a relaxed version of equivariance [Kaba et al., 2023, Sannai et al., 2021].

An alternative method for tractability is to interpret Equation (3.1) as an expectation with uniform distribution $\mathrm{Unif}(G)$ over the compact group G [Mezzadri, 2006], and use sampling-based average to estimate it [Yarotsky, 2022, Murphy et al., 2019a,b]:

$$\phi_\theta(\mathbf{x}) = \mathbb{E}_{g \sim \mathrm{Unif}(G)} \left[g \cdot f_\theta(g^{-1} \cdot \mathbf{x}) \right], \tag{3.2}$$

where $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$ serves as an unbiased estimator of $\phi_\theta(\mathbf{x})$. While simple and general, this approach has practical issues that the base function f_θ is burdened to learn all equally possible group transformations, and the expectedly high variance of the estimator can lead to challenges in sampling-based training due to large variance of gradients as well as sample complexity of inference.

Our key idea is to replace the uniform distribution $\mathrm{Unif}(G)$ for the expectation in Equation (3.2) with a parameterized distribution $p_\omega(g|\mathbf{x})$ in a way that equivariance and expressive power are always guaranteed, and train it end-to-end with the base function f_θ to directly minimize task loss. We show

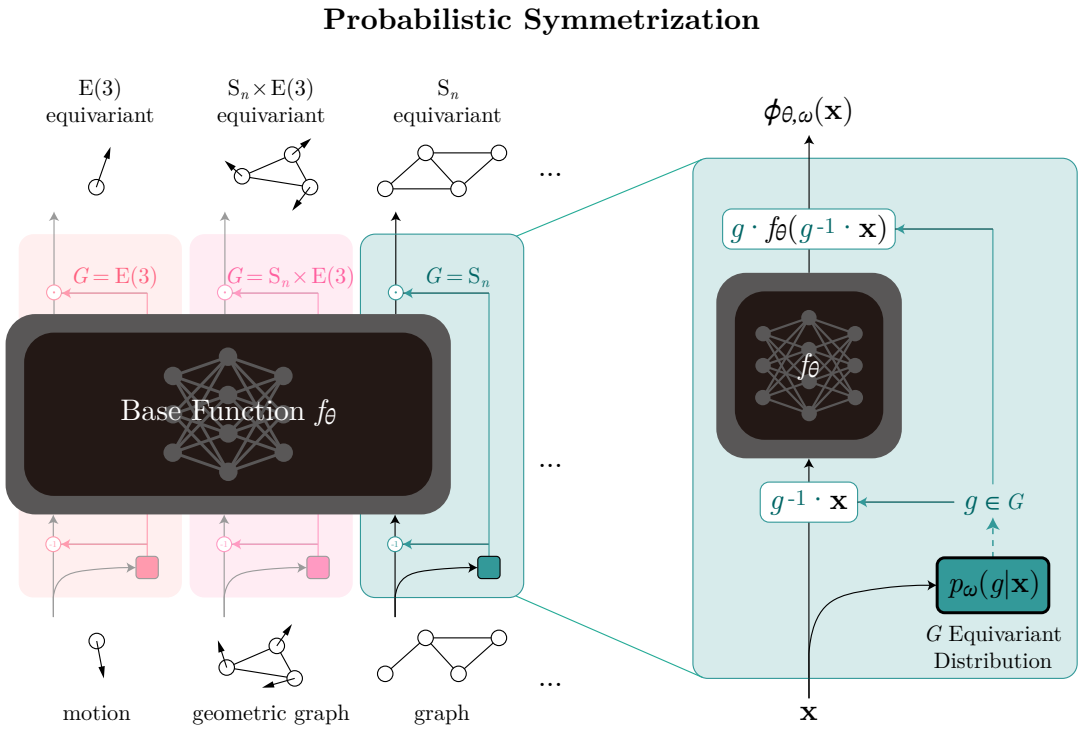


Figure 3.1: Overview of probabilistic symmetrization. We symmetrize an unconstrained base function f_θ into an equivariant function $\phi_{\theta,\omega}$ for group G using a learned equivariant distribution $p_\omega(g|\mathbf{x})$.

that the distribution $p_\omega(g|\mathbf{x})$ only needs to satisfy one simple condition to guarantee equivariance and expressive power: it has to be probabilistically equivariant [Bloem-Reddy and Teh, 2020]. This allows us to generally implement $p_\omega(g|\mathbf{x})$ as a noise-outsourced map $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto g$ with an invariant noise $\boldsymbol{\epsilon}$ and a small equivariant network q_ω, which enables gradient-based training with reparameterization [Kingma and Welling, 2014]. As p_ω is trained, it can enhance the learning of f_θ by producing group transformations of lower variance compared to Unif(G) so that f_θ is less burdened, and coordinating with f_θ to maximize task performance. We refer to our approach as probabilistic symmetrization. An overview is in Figure 3.1.

We implement and test our method with two general-purpose architectures as the base function f_θ: MLP and transformer. In particular, our transformer backbone is architecturally identical to patch-based vision transformers [Dosovitskiy et al., 2021], which allows us to initialize most of its parameters from ImageNet-21k pretrained weights [Steiner et al., 2021] and only replace the input and output projections to match task dimensions. We implement the conditional distribution $p_\omega(g|\mathbf{x})$ for a wide range of practical symmetry groups including permutation (S_n) and Euclidean groups ($O/SO(d)$ and $E/SE(d)$) and their product combinations (e.g., $S_n \times O(3)$), all of which are combinatorial or infinite groups. Empirical tests on a wide range of invariant and equivariant tasks involving graphs and motion data demonstrate competitive results against tailored equivariant architectures as well as existing symmetrization methods [Puny et al., 2022, Kaba et al., 2023]. This suggests the potential for learning invariant or equivariant functions for diverse groups with a group-agnostic and general-purpose backbone. We further show evidence that pretraining from non-symmetric modality (vision) leads to enhanced learning in symmetric modality (graphs), which indicates knowledge transfer across symmetries through weights θ.

3.1.1 Method

We introduce and analyze our approach called probabilistic symmetrization, which involves an equivariant distribution p_ω and group-agnostic base function f_θ. We then describe implementation of p_ω for practical groups including permutations and rotations in Section 3.1.1. Then, we describe our choice of base function f_θ focusing on MLP and transformers in particular.

Problem setup In general, our goal is to construct a function $\phi : \mathcal{X} \to \mathcal{Y}$ on finite vector spaces $\mathcal{X}, \mathcal{Y}$ that is invariant or equivariant to symmetry specified by a group $G = \{g\}$[1]. This is formally described by specifying how the group act as transformations on the input and output. A group representation $\rho : G \to \mathrm{GL}(\mathcal{X})$, where $\mathrm{GL}(\mathcal{X})$ is the set of all invertible matrices on $\mathcal{X}$, associates each group element $g \in G$ to an invertible matrix $\rho(g)$ that transforms a given vector $\mathbf{x} \in \mathcal{X}$ through $\mathbf{x} \mapsto g \cdot \mathbf{x} = \rho(g)\mathbf{x}$. Given that, a function $\phi : \mathcal{X} \to \mathcal{Y}$ is G-equivariant if:

$$\phi(\rho_1(g)\mathbf{x}) = \rho_2(g)\phi(\mathbf{x}), \qquad \forall \mathbf{x} \in \mathcal{X}, g \in G,$$

where the representations ρ_1 and ρ_2 are on the input and output, respectively. G-invariance is a special case of equivariance when the output representation is trivial, $\rho_2(g) = \mathbf{I}$.

Probabilistic symmetrization

To construct a G-equivariant function ϕ_θ, group averaging symmetrizes an arbitrary base function $f_\theta : \mathcal{X} \to \mathcal{Y}$ by taking expectation with uniform distribution over the group (Equation (3.2)). Instead, we propose to use an input-conditional parameterized distribution $p_\omega(g|\mathbf{x})$ and symmetrize f_θ as follows:

$$\phi_{\theta,\omega}(\mathbf{x}) = \mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right], \tag{3.3}$$

where the distribution $p_\omega(g|\mathbf{x})$ itself satisfies probabilistic G-equivariance:

$$p_\omega(g|\mathbf{x}) = p_\omega(g'g|\rho_1(g')\mathbf{x}), \qquad \forall \mathbf{x} \in \mathcal{X}, g, g' \in G.$$

Importantly, we show that probabilistic symmetrization with equivariant p_ω guarantees equivariance as well as expressive power of the symmetrized $\phi_{\theta,\omega}$.

Theorem 9. *If p_ω is G-equivariant, then $\phi_{\theta,\omega}$ is G-equivariant for arbitrary f_θ.*

Theorem 10. *If p_ω is G-equivariant and f_θ is a universal approximator, then $\phi_{\theta,\omega}$ is a universal approximator of G-equivariant functions.*

While the base function f_θ that guarantees universal approximation can be chosen in a group-agnostic manner e.g., an MLP [Hornik et al., 1989, Cybenko, 1989] or a transformer on token sequences [Yun et al., 2020], the distribution p_ω needs to be instantiated group specifically to satisfy G-equivariance. A simplistic choice is using uniform distribution $\mathrm{Unif}(G)$ for all inputs $\mathbf{x}$ with no parameterization (reducing to group averaging), which is technically equivariant and therefore guarantees equivariance and universality. However, appropriately parameterizing and learning p_ω can provide distinguished advantages compared to the uniform distribution, as it can **(1)** learn from data to collaborate with (pretrained) base

[1] In this work, we assume the group G to be compact.

function f_θ to maximize task performance and **(2)** learn to produce more consistent samples $g \sim p_\omega(g|\mathbf{x})$ that can offer more stable gradients for the base function f_θ during early training.

We now provide a generic blueprint of G-equivariant distribution $p_\omega(g|\mathbf{x})$ for any compact group G. Our goal is to sample $g \sim p_\omega(g|\mathbf{x})$ to obtain group representation $\rho(g)$ for symmetrization (Equation (3.3)) in a differentiable manner so that p_ω can be trained end-to-end. Since we only need sampling and there is no need to evaluate likelihoods, we simply implement $p_\omega(g|\mathbf{x})$ as a noise-outsourced, differentiable transformation $q_\omega(\mathbf{x}, \boldsymbol{\epsilon})$ of a noise variable $\boldsymbol{\epsilon} \in \mathcal{E}$ that directly outputs a group representation $\rho(g)$:

$$\rho(g) = q_\omega(\mathbf{x}, \boldsymbol{\epsilon}), \qquad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}), \tag{3.4}$$

where q_ω is G-equivariant and $p(\boldsymbol{\epsilon})$ is G invariant under an appropriate representation ρ':

$$q_\omega(\rho_1(g)\mathbf{x}, \rho'(g)\boldsymbol{\epsilon}) = \rho(g) q_\omega(\mathbf{x}, \boldsymbol{\epsilon}), \qquad p(\boldsymbol{\epsilon}) = p(\rho'(g)\boldsymbol{\epsilon}), \qquad \forall \mathbf{x} \in \mathcal{X}, \boldsymbol{\epsilon} \in \mathcal{E}, g \in G.$$

Given above implementation, we can show the G-equivariance of p_ω:

Theorem 11. *If q_ω is G-equivariant and $p(\boldsymbol{\epsilon})$ is G invariant under representation ρ' that $|\det \rho'(g)| = 1 \forall g \in G$, the distribution $p_\omega(g|\mathbf{x})$ characterized by $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \rho(g)$ is G-equivariant.*

In practice, one can use any available G-equivariant network to implement q_ω, e.g., a GNN for the symmetric group S_n, or an equivariant MLP which can be constructed for any matrix group [Finzi et al., 2021]. Since we expect most of the reasoning to be done by the base function f_θ, the equivariant network q_ω can be small and relatively less expressive. This allows us to get less affected by their known issues in expressiveness and scaling. For the noise $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, simple choices often suffice for G invariance. For example, standard normal $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}_n)$ provides invariance for the symmetric group S_n as well as the (special) orthogonal group $\mathrm{O}(n)$ and $\mathrm{SO}(n)$.

One important detail in designing q_ω is constraining its output to be a valid group representation $\rho(g)$. For this, we apply appropriate postprocessing to refine neural network features into group representations, e.g., Gram-Schmidt orthogonalization to obtain a representation $\rho(g) \in \mathbb{R}^{n \times n}$ of the orthogonal group $g \in \mathrm{O}(n)$. Importantly, to not break the G-equivariance of q_ω, this postprocessing needs to be equivariant itself, e.g., Gram-Schmidt process is itself $\mathrm{O}(n)$ equivariant [Kaba et al., 2023]. We discuss the implementations of p_ω for a range of practical symmetry groups in the following section.

Equivariant distribution p_ω

We present implementations of the G-equivariant distribution $p_\omega(g|\mathbf{x})$ for a range of practical groups demonstrated in our experiments (Section 3.1.3). Formal proofs of correctness are in Appendix A.4.1.

Symmetric group $\mathbf{S}_n$ The symmetric group S_n over a finite set of n elements contains all permutations of the set, which describes symmetry to ordering desired for learning set and graph data. The base representation is given by $\rho(g) = \mathbf{P}_g$ where $\mathbf{P}_g \in \{0, 1\}^{n \times n}$ is a permutation matrix for g.

To implement S_n-equivariant distribution $p_\omega(g|\mathbf{x})$ that provides permutation matrices $\mathbf{P}_g$ from graph data $\mathbf{x}$, we use the following. We first sample invariant noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times d}$ from i.i.d. uniform $\mathrm{Unif}[0, \eta]$ with noise scale η. For the S_n-equivariant map $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \rho(g)$, we first use a graph neural network (GNN) as an equivariant map $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{Z}$ that outputs nodewise scalar $\mathbf{Z} \in \mathbb{R}^n$. Then, assuming $\mathbf{Z}$ is tie-free[2], we

[2]This can be assumed since the invariant noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times d}$ serves as tiebreaker between the n nodes.

use argsort $\mathbf{Z} \mapsto \mathbf{P}_g$ below to get a permutation matrix [Prillo and Eisenschlos, 2020, Winter et al., 2021]:

$$\mathbf{P}_g = \mathrm{eq}(\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{Z})^\top),$$

where eq denotes elementwise equality indicator. The argsort operator is S_n-equivariant, i.e., it maps $\mathbf{P}_{g'}\mathbf{Z} \mapsto \mathbf{P}_{g'}\mathbf{P}_g$ for all $\mathbf{P}_{g'} \in \mathrm{S}_n$. To backpropagate through $\mathbf{P}_g$ during training, we use straight-through gradient estimator [Bengio et al., 2013] with an approximate permutation matrix $\hat{\mathbf{P}}_g \approx \mathbf{P}_g$ obtained from a differentiable relaxation of argsort operator [Mena et al., 2018, Grover et al., 2019, Winter et al., 2021].

Orthogonal groups $\mathbf{O}(n)$, $\mathbf{SO}(n)$ The orthogonal group $\mathrm{O}(n)$ contains all roto-reflections in $\mathbb{R}^n$ around origin, and the special orthogonal group $\mathrm{SO}(n)$ contains all rotations without reflections. These groups describe rotation symmetries desirable in learning geometric data. The base group representation for $\mathrm{O}(n)$ is given by $\rho(g) = \mathbf{Q}_g$ where $\mathbf{Q}_g$ is the orthogonal matrix for g. For $\mathrm{SO}(n)$, the representation is $\rho(g) = \mathbf{Q}_g^+$ where $\mathbf{Q}_g^+$ is the orthogonal matrix of g with determinant $+1$.

To implement $\mathrm{O}(n)/\mathrm{SO}(n)$-equivariant distribution $p_\omega(g|\mathbf{x})$ that provides orthogonal matrices given input data $\mathbf{x}$, we use the following design. We first sample invariant noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n\times d}$ from i.i.d. normal $\mathcal{N}(0, \eta^2)$ with noise scale η. For the $\mathrm{O}(n)/\mathrm{SO}(n)$-equivariant map $p_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \rho(g)$, we first use an equivariant neural network as a map $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{Z}$ that outputs n features $\mathbf{Z} \in \mathbb{R}^{n\times n}$. Then, assuming $\mathbf{Z}$ is full-rank[3], we use Gram-Schmidt process for orthogonalization, which is differentiable and $\mathrm{O}(n)$-equivariant [Kaba et al., 2023]. This completes the postprocessing $\mathbf{Z} \mapsto \mathbf{Q}_g$ for $\mathrm{O}(n)$. For $\mathrm{SO}(n)$, we can further use a simple scale operator $\mathbf{Q} \mapsto \mathbf{Q}_g^+$ to set the determinant of orthogonalized matrix to $+1$:

$$\text{scale} : \left[\begin{array}{c|c|c} \mathbf{Q}_1 & \ldots & \mathbf{Q}_n \end{array}\right] \mapsto \left[\begin{array}{c|c|c} \det(\mathbf{Q}) \cdot \mathbf{Q}_1 & \ldots & \mathbf{Q}_n \end{array}\right],$$

The scale operator is differentiable and $\mathrm{SO}(n)$-equivariant, i.e., it maps $\mathbf{Q}_{g'}^+\mathbf{Q} \mapsto \mathbf{Q}_{g'}^+\mathbf{Q}_g^+$ for all $\mathbf{Q}_{g'}^+ \in \mathrm{SO}(n)$, thereby completing the postprocessing $\mathbf{Z} \mapsto \mathbf{Q}_g^+$ for $\mathrm{SO}(n)$.

Euclidean groups $\mathbf{E}(n)$, $\mathbf{SE}(n)$ The Euclidean group $\mathrm{E}(n)$ contains all roto-translations and reflections in $\mathbb{R}^n$ and their combinations, and the special Euclidean group $\mathrm{SE}(n)$ contains all roto-translations without reflections. These groups are desired in learning physical systems such as a particle in motion. Formally, the Euclidean group is given as a combination of orthogonal group and translation group $\mathrm{E}(n) = \mathrm{O}(n) \ltimes \mathrm{T}(n)$, and similarly $\mathrm{SE}(n) = \mathrm{SO}(n) \ltimes \mathrm{T}(n)$. As the translation group $\mathrm{T}(n)$ is non-compact which violates our assumption for symmetrization, we handle it separately. Following prior work [Puny et al., 2022, Kaba et al., 2023], we subtract the centroid $\bar{\mathbf{x}}$ from the input data $\mathbf{x}$ as $\mathbf{x} - \bar{\mathbf{x}}$, and add it to the rotation symmetrized output as $\bar{\mathbf{x}} + g \cdot f_\theta(g^{-1} \cdot (\mathbf{x} - \bar{\mathbf{x}}))$ where g is sampled from $\mathrm{O}(n)/\mathrm{SO}(n)$-equivariant $p_\omega(g|\mathbf{x} - \bar{\mathbf{x}})$. This makes the overall symmetrized function $\mathrm{E}(n)/\mathrm{SE}(n)$-equivariant.

Product groups $H \times K$ While we have described several groups individually, in practice we often encounter product combinations of groups $G = H \times K$ that describe joint symmetry to each group H and K. For example, $\mathrm{S}_n \times \mathrm{O}(3)$ describes joint symmetry to permutations and rotations, which is desired in learning point clouds, molecules, and particle interactions. In general, an element of $H \times K$ is given as $g = (h, k)$ where $h \in H$ and $k \in K$, and group operations are applied elementwise $gg' = (h, k)(h', k') = (hh', kk')$.

[3] In practice, we add a random Gaussian matrix of a tiny magnitude to $\mathbf{Z}$ to prevent rank collapse.

The base group representation is accordingly given as pair of representations $\rho(g) = (\rho(h), \rho(k))$. While a common approach to handling $H \times K$ is *partially* symmetrizing on H and imposing K equivariance on the base architecture (e.g., rotational symmetrization of a graph neural network for $\mathrm{S}_n \times \mathrm{O}(3)$ equivariance [Puny et al., 2022, Kaba et al., 2023]), we extend to *full symmetrization* on $H \times K$ since our goal is not imposing any constraint on the base function f_θ.

To implement $H \times K$-equivariant distribution $p_\omega(g|\mathbf{x})$ that gives $\rho(g) = (\rho(h), \rho(k))$ from data $\mathbf{x}$, we use the following design. We first sample invariant noise $\boldsymbol{\epsilon}$ from i.i.d. normal $\mathcal{N}(0, \eta^2)$ with scale η. For the $H \times K$-equivariant map $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto (\rho(h), \rho(k))$, we employ a $H \times K$-equivariant neural network as a map $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto (\mathbf{Z}_H, \mathbf{Z}_K)$ such that the postprocessing for each group H and K provides maps $\mathbf{Z}_H \mapsto \rho(h)$ and $\mathbf{Z}_K \mapsto \rho(k)$ respectively, leading to full representation $\rho(g) = (\rho(h), \rho(k))$. For this whole procedure to be $H \times K$-equivariant, it is sufficient to have $\mathbf{Z}_H$ be K-invariant and $\mathbf{Z}_K$ be H-invariant. These are special cases of $H \times K$-equivariance, and is supported by a range of equivariant neural networks especially regarding $\mathrm{S}_n \times \mathrm{O}(3)$ or $\mathrm{S}_n \times \mathrm{SO}(3)$ equivariances [Deng et al., 2021].

Base function f_θ

We now describe the choice of group-agnostic base function $f_\theta : \mathbf{x} \mapsto \mathbf{y}$. As group symmetry is handled by the equivariant distribution $p_\omega(g|\mathbf{x})$, any symmetry concern is *hidden* from f_θ, allowing the inputs $\mathbf{x}$ and outputs $\mathbf{y}$ to be treated as plain multidimensional arrays. This allows us to implement f_θ with powerful general-purpose architectures, namely as an MLP that operates on flattened vectors of inputs and outputs, or a transformer as we describe below.

Let inputs $\mathbf{x} \in \mathcal{X} = \mathbb{R}^{n_1 \times \ldots \times n_a \times c}$ and outputs $\mathbf{y} \in \mathcal{Y} = \mathbb{R}^{n'_1 \times \ldots \times n'_b \times c'}$ be multidimensional arrays with c and c' channels, respectively. Our transformer base function $f_\theta : \mathbf{x} \mapsto \mathbf{y}$ is given as:

$$f_\theta = \text{detokenize} \circ \text{transformer} \circ \text{tokenize},$$

where $\text{tokenize} : \mathcal{X} \to \mathbb{R}^{m \times d}$ parses input array to a sequence of m tokens, $\text{transformer} : \mathbb{R}^{m \times d} \to \mathbb{R}^{m \times d}$ is a standard transformer encoder on tokens used in language and vision [Devlin et al., 2019, Dosovitskiy et al., 2021], and $\text{detokenize} : \mathbb{R}^{m \times d} \to \mathcal{Y}$ decodes encoded tokens to output array. For the tokenizer and detokenizer, we can use linear projections on flattened chunks of the array, which directly extends flattened patch projections in vision transformers to higher dimensions [Dosovitskiy et al., 2021, Shen et al., 2023, Shi et al., 2016]. This enables mapping between different dimensional inputs and outputs, *e.g*, for graph node classification with $\mathbf{x} \in \mathbb{R}^{n \times n \times c}$ and $\mathbf{y} \in \mathbb{R}^{n \times c'}$, it is possible to use 2D patch projection for the input and 1D projection for the output.

Above choice of f_θ offers important advantages including universal approximation [Yun et al., 2020] and ability to share and transfer the learned knowledge in θ over different domains of different group symmetries. Remarkably, this allows us to directly leverage large-scale pretrained parameters from data-abundant domains for learning on symmetric domains. In our experiments, we only replace the tokenizer and detokenizer of a vision transformer pretrained on ImageNet-21k [Wolf et al., 2019, Dosovitskiy et al., 2021] and fine-tune it to perform diverse S_n-equivariant tasks such as graph classification, node classification, and link prediction.

3.1.2 Related work

We discuss relation of our symmetrization method to prior ones, specifically group averaging [Yarotsky, 2022, Basu et al., 2022], frame averaging [Puny et al., 2022], and canonicalization [Kaba et al., 2023].

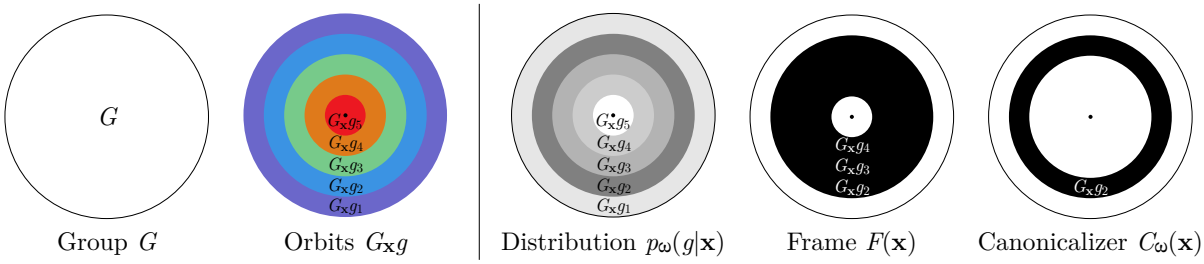


Figure 3.2: Visual illustration of the symmetrization methods based on probabilities assigned upon the partitioning of the group G into orbits $G_{\mathbf{x}}g$. Note that, while we use concentric circles of different perimeters to illustrate each orbit, all orbits actually have an identical cardinality $|G_{\mathbf{x}}g| = |G_{\mathbf{x}}|$.

Since the methods share a common formalization $\mathbf{y} = \mathbb{E}_g[g \cdot f_\theta(g^{-1} \cdot \mathbf{x})]$, one can expect a close theoretical relationship between them. We observe that probabilistic symmetrization is quite general; based on particular choices of the G-equivariant distribution $p_\omega(g|\mathbf{x})$, it can become most of the related symmetrization methods as special cases. This can be easily seen for group averaging [Yarotsky, 2022], as the distribution p_ω can reduce to the uniform distribution $\text{Unif}(G)$ over the group.

For frame averaging [Puny et al., 2022] and canonicalization [Kaba et al., 2023], we show two useful lemmas based on the concept of the stabilizer subgroup $G_{\mathbf{x}} = \{h \in G : \rho(h)\mathbf{x} = \mathbf{x}\}$. The first lemma shows that any group G is partitioned into disjoint orbits $G_{\mathbf{x}}g$ of group elements g under the action of $G_{\mathbf{x}}$ by left multiplication, as illustrated in the first and second panels of Figure 3.2.

Lemma 2. *Any group G is a union of disjoint orbits $G_{\mathbf{x}}g$ of equal cardinality.*

Proof. Let us consider the equivalence relation $\sim$ on G induced by the action of the stabilizer $G_{\mathbf{x}}$, defined as $g \sim h \iff h \in G_{\mathbf{x}}g$. The orbits $G_{\mathbf{x}}g$ are the equivalence classes under this relation, and the set of all orbits of G under the action of $G_{\mathbf{x}}$ forms a partition of G (i.e., the quotient $G/G_{\mathbf{x}}$). Furthermore, since $G_{\mathbf{x}} \leq G$ and right multiplication by some $g \in G$ is a faithful action of G on itself, we have $|G_{\mathbf{x}}g| = |G_{\mathbf{x}}|$ for all $g \in G$, which shows that all orbits $G_{\mathbf{x}}g$ have equal cardinality. □

The second lemma characterizes probability distributions over G that can be expressed with $p_\omega(g|\mathbf{x})$ under our method, which we illustrate in the third panel of Figure 3.2. Intuitively, $p_\omega(g|\mathbf{x})$ assigns constant probability densities over each of the orbit $G_{\mathbf{x}}g$ that partitions G as shown in Lemma 2.

Lemma 3. *G-equivariant $p_\omega(g|\mathbf{x})$ assigns identical probability to all elements on each orbit $G_{\mathbf{x}}g$.*

Proof. With equivariance, we have $p_\omega(g|\mathbf{x}) = p_\omega(hg|h \cdot \mathbf{x})$. Since $h \cdot \mathbf{x} = \mathbf{x}$ for all $h \in G_{\mathbf{x}}$, we have $p_\omega(g|\mathbf{x}) = p_\omega(hg|\mathbf{x})$ for all $h \in G_{\mathbf{x}}$; all elements on orbit $G_{\mathbf{x}}g$ have an identical probability. □

We now show that our method subsumes frame averaging [Puny et al., 2022] as a special case. Frame averaging also takes an average, but over a subset of group given by a frame $F : \mathcal{X} \to 2^G \setminus \emptyset$; importantly, it is required that the frame itself is G-equivariant $F(\rho(g)\mathbf{x}) = gF(\mathbf{x})$. We show that:

Proposition 5. *Probabilistic symmetrization with G-equivariant distribution $p_\omega(g|\mathbf{x})$ can become frame averaging [Puny et al., 2022] by assigning uniform density to a set of orbits $G_{\mathbf{x}}g$ for some group elements g.*

Proof. For a frame F, frame averaging is expressed as follows:

$$\frac{1}{|F(\mathbf{x})|} \sum_{g \in F(\mathbf{x})} \left[g \cdot f_\theta(g^{-1} \cdot \mathbf{x}) \right] = \mathbb{E}_{g \sim \text{Unif}(F(\mathbf{x}))} \left[g \cdot f_\theta(g^{-1} \cdot \mathbf{x}) \right].$$

From Puny et al. [2022, Theorem 3], we have that $F(\mathbf{x})$ is a disjoint union of equal size orbits $G_{\mathbf{x}}g$. Therefore, $\text{Unif}(F(\mathbf{x}))$ is a uniform probability distribution over the union of the orbits. This can be expressed by a G-equivariant distribution $p_\omega(g|\mathbf{x})$ by assigning identical probability over all orbits in the frame F and zero probability to all orbits not in the frame (illustrated in the fourth panel of Figure 3.2). Therefore, probabilistic symmetrization can become frame averaging. □

Canonicalization [Kaba et al., 2023] uses a *single* group element for symmetrization, produced by a trainable canonicalizer $C_\omega : \mathcal{X} \to G$. Here, it is required that the canonicalizer itself satisfies *relaxed* G-equivariance $C(\rho(g)\mathbf{x}) = ghC(\mathbf{x})$ up to an arbitrary stabilizer element $h \in G_{\mathbf{x}}$. We show:

Proposition 6. *Probabilistic symmetrization with G-equivariant distribution $p_\omega(g|\mathbf{x})$ can become canonicalization [Kaba et al., 2023] by assigning uniform density to a single orbit $G_{\mathbf{x}}g$ of some group element g.*

Proof. For some canonicalizer C_ω, canonicalization is defined as follows:

$$g \cdot f_\theta(g^{-1} \cdot \mathbf{x}), \qquad g = C_\omega(\mathbf{x}).$$

From relaxed G-equivariance, we have $C_\omega(\mathbf{x}) = hC_\omega(\mathbf{x})$ for every $h \in G_{\mathbf{x}}$. Then a valid choice for the canonicalizer C_ω is a stochastic map that samples from the uniform distribution over a single-orbit frame; $C_\omega(\mathbf{x}) \sim \text{Unif}(F_\omega(\mathbf{x}))$ where $F_\omega(\mathbf{x}) = G_{\mathbf{x}}g$ for some g. In this case, canonicalization is equivalent to a one-sample estimation of the expectation $\mathbb{E}_{g\sim\text{Unif}(F_\omega(\mathbf{x}))}\left[g \cdot f_\theta(g^{-1} \cdot \mathbf{x})\right]$. This can be expressed by $p_\omega(g|\mathbf{x})$ by assigning nonzero probability to the single orbit $G_{\mathbf{x}}g$ and zero to the rest (illustrated in the last panel of Figure 3.2). Therefore, probabilistic symmetrization can become canonicalization. □

Assuming that stabilizer $G_{\mathbf{x}}$ is trivial, our method can become canonicalization by removing random noise $\boldsymbol{\epsilon}$, which reduces p_ω to deterministic map $\rho(g) = q_\omega(\mathbf{x})$. We use this approach to implement canonicalizer for the S_n group, while Kaba et al. [2023] only provides canonicalizers for Euclidean groups.

3.1.3 Experiments

We demonstrate probabilistic symmetrization on a range of symmetry groups S_n, E(3) (O(3)), and the product $\mathrm{S}_n \times \mathrm{E}(3)$ ($\mathrm{S}_n \times \mathrm{O}(3)$), on a variety of invariant and equivariant tasks, with general-purpose base functions f_θ chosen as MLP and transformer optionally with pretraining from vision domain.

Graph isomorphism learning with MLP

Building expressive neural networks for graphs (S_n) has been considered important and challenging, as simple and efficient GNNs are often limited in expressive power to certain Weisfeiler-Lehman isomorphism tests like 1-WL [Xu et al., 2019, Maron et al., 2019a]. Since an MLP equipped with probabilistic symmetrization is in theory universal and S_n-equivariant, it has potential for graph learning that require high expressive power. To explicitly test this, we adopt the experimental setup of Puny et al. [2022] and use two datasets on graph separation task (S_n-invariant). GRAPH8c [Balcilar et al., 2021] consists of all non-isomorphic connected graphs with 8 nodes, and EXP [Abboud et al., 2021] consists of 3-WL distinguishable graphs that are not 2-WL distinguishable. We compare against standard GNNs as well as an MLP symmetrized with group averaging [Yarotsky, 2022], frame averaging [Puny et al., 2022], and canonicalization [Kaba et al., 2023]. Our method uses the same MLP architecture to these baselines, and its S_n-equivariant distribution p_ω for symmetrization is implemented using a 3-layer GIN [Xu et al., 2019] which is 1-WL expressive. We use 10 samples for symmetrization during both training and testing.

Table 3.1: Results for S_n-invariant graph separation. We use two tasks, one for counting pairs of graphs not separated by a model at random initialization (GRAPH8c and EXP), and one for learning to classify EXP to two classes (EXP-classify). For EXP-classify, we report the test accuracy at best validation accuracy. The columns arch. and sym. denote architectural and symmetrized equivariance, respectively. The results for baselines are from Puny et al. [2022] except for MLP-Canonical. which is tested by us.

	arch.	sym.	GRAPH8c ↓	EXP ↓	EXP-classify ↑
GCN [Kipf and Welling, 2017]	S_n	-	4755	600	50%
GAT [Velickovic et al., 2018]	S_n	-	1828	600	50%
GIN [Xu et al., 2019]	S_n	-	386	600	50%
ChebNet [Defferrard et al., 2016]	S_n	-	44	71	82%
PPGN [Maron et al., 2019a]	S_n	-	0	0	**100%**
GNNML3 [Balcilar et al., 2021]	S_n	-	0	0	**100%**
MLP-GA [Yarotsky, 2022]	-	S_n	0	0	50%
MLP-FA [Puny et al., 2022]	-	S_n	0	0	**100%**
MLP-Canonical.	-	S_n	0	0	50%
MLP-PS (Ours), fixed ϵ	-	S_n	0	0	79.5%
MLP-PS (Ours)	-	S_n	0	0	**100%**

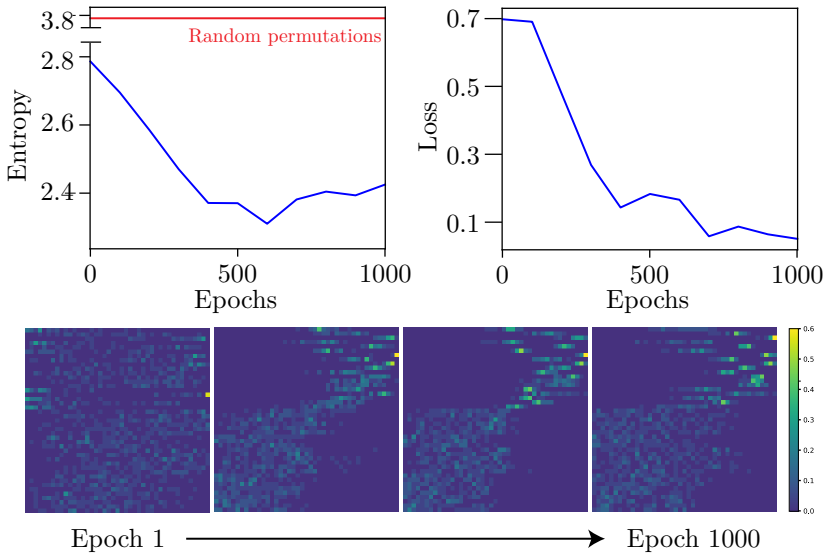


Figure 3.3: Learned $p_\omega(g|\mathbf{x})$ over time. The entropy of aggregated permutation matrices $\bar{\mathbf{P}} = \sum \mathbf{P}_g/N$ from $\mathbf{P}_g \sim p_\omega(g|\mathbf{x})$ for each input $\mathbf{x}$ drops in early training, indicating that the distribution learns to produce lower-variance permutations as in below visualizations.

The results are in Table 3.1. At random initialization, all symmetrization methods are able to provide perfect separation of all graphs, similar to PPGN [Maron et al., 2019a] and GNNML3 [Balcilar et al., 2021] that are equivariant networks carefully designed to be 3-WL expressive. However, when trained with gradient descent to solve classification problem, naïve symmetrization with group averaging fails, presumably because the MLP fails to adjust to equally possible 64! permutations of 64 nodes in maximum. On the other hand, our method is able to learn the task, achieving the same accuracy to frame averaging that utilizes costly eigendecomposition of graph Laplacian [Puny et al., 2022]. What makes our method work while group averaging fails? We conjecture this is since the distribution $p_\omega(g|\mathbf{x})$ can learn to provide more consistent permutations during early training, as we illustrate in Figure 3.3. In the figure, we measured the consistency of samples from $p_\omega(g|\mathbf{x})$ over training progress by sampling $N = 50$ permutation matrices $\mathbf{P}_g \sim p_\omega(g|\mathbf{x})$ and measuring the row-wise entropy of their average $\bar{\mathbf{P}} = \sum \mathbf{P}_g/N$

Table 3.2: Results for $S_n \times E(3)$-equivariant n-body problem. The columns arch. and sym. denote architectural and symmetrized equivariance, respectively. We report test MSE at best validation MSE, along with the standard deviation for GA and Ours where predictions are stochastic. The results for baselines are from Kaba et al. [2023] except symmetrized transformers which are tested by us.

	arch.	sym.	Position MSE ↓
SE(3) Transformer [Fuchs et al., 2020]	$S_n \times SE(3)$	-	0.0244
TFN [Thomas et al., 2018]	$S_n \times SE(3)$	-	0.0155
Radial Field [Köhler et al., 2019]	$S_n \times E(3)$	-	0.0104
EGNN [Satorras et al., 2021]	$S_n \times E(3)$	-	0.0071
GNN-FA [Puny et al., 2022]	S_n	E(3)	0.0057
GNN-Canonical. [Kaba et al., 2023]	S_n	E(3)	0.0043
Transformer-Canonical.	-	$S_n \times E(3)$	0.00508
Transformer-GA	-	$S_n \times E(3)$	0.00414 ± 0.00001
Transformer-PS (Ours)	-	$S_n \times E(3)$	$\mathbf{0.00401 \pm 0.00001}$

for each input $\mathbf{x}$. The more consistent the sampled permutations, the sharper their average, and lesser the entropy. As training progresses, p_ω learns to produce more consistent samples, which coincides with the initial increase in task performance. Given that, a natural question would be: if we enforce the samples $g \sim p_\omega(g|\mathbf{x})$ to be consistent from the first place, would it work? To answer this, we also tested a non-probabilistic version of our model that uses a single permutation per input $\rho(g) = q_\omega(\mathbf{x})$, which is a canonicalizer under relaxed equivariance [Kaba et al., 2023] as described in Section 3.1.2. As in Table 3.1, canonicalization fails, suggesting that probabilistic nature of $p_\omega(g|\mathbf{x})$ can be *beneficial* for learning. For another deterministic version of our model made by fixing the noise $\boldsymbol{\epsilon}$ (Equation (3.4)) at initialization, the performance drops to 79.5%, further implying that stochasticity of $p_\omega(g|\mathbf{x})$ has a role.

Particle dynamics learning with transformer

Learning sets or graphs attributed with position and velocity in 3D ($S_n \times E(3)$) is practically significant as they universally appear in physics, chemistry, and biology applications. While prior symmetrization methods employ an already S_n-equivariant base function and partially symmetrize the E(3) part, we attempt to symmetrize the entire product group $S_n \times E(3)$ and choose the base model f_θ as a sequence transformer to leverage its expressive power. For empirical demonstration, we adopt the experimental setup of Kaba et al. [2023] and use the n-body dataset [Satorras et al., 2021, Fuchs et al., 2020] where the task is predicting the position of $n = 5$ charged particles after certain time given their initial position and velocity in $\mathbb{R}^3$ ($S_n \times E(3)$-equivariant). We compare our method to $S_n \times E(3)$-equivariant networks and partial symmetrization methods applying E(3) symmetrization to GNNs. We also test prior symmetrization methods on the full group $S_n \times E(3)$ along our method, but could not test for frame averaging since equivariant frames for the full group $S_n \times E(3)$ was not available in current literature. Our method is implemented using a transformer with sequence positional encodings with around $2.3\times$ parameters of the baselines, and the $S_n \times E(3)$-equivariant distribution p_ω for symmetrization is implemented using a 2-layer Vector Neurons [Deng et al., 2021] that has around $0.03\times$ of parameters of the transformer. We use 20 samples for symmetrization during training, and use $10\times$ sample size for testing since the task is regression where appropriate variance reduction is necessary.

The results are in Table 3.2. We observe simple group averaging exhibits a surprisingly strong performance, as it achieves 0.00414 MSE and already outperforms previous state-of-the-art 0.0043 MSE. This is as the permutation component of the symmetry is small, $n = 5$ particles interacting with each other,

Table 3.3: Results for S_n-equivariant node classification on PATTERN. We report test accuracy at the best validation accuracy, along with the standard deviation for GA and Ours where predictions are stochastic. The results for GNN baselines are from Dwivedi et al. [2020].

	pretrain.	Accuracy ↑
GCN [Kipf and Welling, 2017], 16 layers	-	85.614
GAT [Velickovic et al., 2018], 16 layers	-	78.271
GatedGCN [Bresson and Laurent, 2019], 16 layers	-	85.568
GIN [Xu et al., 2019], 16 layers	-	85.387
RingGNN [Chen et al., 2019], 2 layers	-	86.245
RingGNN [Chen et al., 2019], 8 layers	-	diverged
PPGN [Maron et al., 2019a], 3 layers	-	85.661
PPGN [Maron et al., 2019a], 8 layers	-	diverged
ViT-GA, 1-sample	-	76.956 ± 0.033
ViT-GA, 10-sample	-	83.220 ± 0.057
ViT-GA, 1-sample	ImageNet-21k	81.933 ± 0.075
ViT-GA, 10-sample	ImageNet-21k	84.641 ± 0.020
ViT-FA	-	71.377
ViT-FA	ImageNet-21k	80.015
ViT-Canonical.	-	85.825
ViT-Canonical.	ImageNet-21k	86.534
ViT-PS (Ours), 1-sample	-	85.868 ± 0.017
ViT-PS (Ours), 10-sample	-	85.989 ± 0.011
ViT-PS (Ours), 1-sample	ImageNet-21k	86.573 ± 0.030
ViT-PS (Ours), 10-sample	ImageNet-21k	**86.650 ± 0.010**

such that combining it with an expressive base model f_θ (a sequence transformer) can adjust to $5! = 120$ equally possible permutations and their rotations in 3D. Our method outperforms group averaging and achieves a new state-of-the-art 0.00401 MSE, presumably as the distribution p_ω learns to further maximize task performance. On the other hand, the canonicalization approach, implemented by eliminating noise variable $\boldsymbol{\epsilon}$ from our method (Section 3.1.2), performs relatively poorly. We empirically observe that f_θ memorizes the per-input canonical orientations provided by $\rho(g) = q_\omega(\mathbf{x})$ that do not generalize to test inputs. This supports that probabilistic nature of $p_\omega(g|\mathbf{x})$ can be beneficial for performance.

Graph pattern recognition with vision transformer

One important goal of our approach, and symmetrization in general, is to *decouple* the symmetry of problem from the base function f_θ, such that we can leverage knowledge learned from other symmetries by transferring the parameters θ. We demonstrate an extreme case by transferring the parameters of a vision transformer [Dosovitskiy et al., 2021] trained on large-scale image classification (translation invariant) to solve node classification on graphs (S_n-equivariant) for the first time in literature. For this, we use the PATTERN dataset [Dwivedi et al., 2020] that contains 14,000 purely topological random SBM graphs with 44-188 nodes, whose task is finding certain subgraph pattern by binary node classification.

Based on pretrained ViT [Dosovitskiy et al., 2021, Steiner et al., 2021], we construct the base model f_θ by modifying only the input and output layers to take the flattened patches of 2D zero-padded adjacency matrices of size 188×188 and produce output as 1D per-node classification logits of length 188 with 2 channels. In addition to standard GNNs[4], we compare against group averaging, frame averaging, and canonicalization, and also test whether pretrained representations from ImageNet-21k is beneficial for

[4]We note that careful engineering such as graph positional encoding improves performance of GNNs, but we have chosen simple and representative GNNs in the benchmark [Dwivedi et al., 2020] to provide a controlled comparison.

Table 3.4: Results for real-world graph tasks. We report test performance at best validation performance.

	Peptides-func	Peptides-struct	PCQM-Contact			
	AP ↑	MAE ↓	Hits@1 ↑	Hits@3 ↑	Hits@10 ↑	MRR ↑
GCN [Kipf and Welling, 2017]	0.5930	0.3496	0.1321	0.3791	0.8256	0.3234
GCNII [Chen et al., 2020a]	0.5543	0.3471	0.1325	0.3607	0.8116	0.3161
GINE [Hu et al., 2020b]	0.5498	0.3547	0.1337	0.3642	0.8147	0.3180
GatedGCN [Bresson and Laurent, 2019]	0.5864	0.3420	0.1279	0.3783	0.8433	0.3218
GatedGCN+RWSE [Bresson and Laurent, 2019]	0.6069	0.3357	0.1288	0.3808	0.8517	0.3242
Transformer+LapPE [Dwivedi et al., 2022]	0.6326	0.2529	0.1221	0.3679	0.8517	0.3174
SAN+LapPE [Kreuzer et al., 2021]	0.6384	0.2683	0.1355	0.4004	0.8478	0.3350
SAN+RWSE [Kreuzer et al., 2021]	0.6439	0.2545	0.1312	0.4030	0.8550	0.3341
GraphGPS [Rampásek et al., 2022]	0.6535	0.2500	-	-	-	0.3337
Exphormer [Shirzad et al., 2023]	0.6527	**0.2481**	-	-	-	0.3637
ViT-PS (Ours)	**0.6575**	0.2559	**0.3287**	**0.6694**	**0.9526**	**0.5341**

the task. For the S_n-equivariant distribution p_ω in our method and canonicalization, we use a 3-layer GIN [Xu et al., 2019] with only 0.02% of the base model parameters.

The results are in Table 3.3. Transferring pretrained ViT parameters consistently improves node classification for all symmetrization methods. It indicates that some traits of the pretrained visual representation can benefit learning graph tasks which vastly differ in both the underlying symmetry (translation invariance $\to$ S_n-equivariance) and the data generating process (natural images $\to$ random process of SBM). In particular, vision pretraining allows group averaging to achieve 84.641% accuracy, on par with GNN baselines, which is surprising considering that memorizing all 188! equally possible permutations in this dataset is impossible. We conjecture that group averaged ViT can in some way learn meaningful graph representation internally to solve the task, and vision pretraining helps in acquiring the representation by providing a good initialization point or transferable computation motifs.

On the other hand, frame averaging shows a lower performance, 80.015% accuracy with vision pretraining, which is also surprising considering that frames vastly reduce the sample space of symmetrization in general; in fact, the size of frame of each graph in PATTERN is exactly 1. We observe that, unlike group averaging, ViT with frame averaging memorizes frames of training graphs rather than learning generalizable graph representations. In contrast, canonicalization that also uses a single sample per graph successfully learns the task with 86.534% accuracy. We conjecture that the learnable orderings provided by an equivariant network $\rho(g) = q_\omega(\mathbf{x})$ is more flexible and generalizable to unseen graphs compared to frames computed from fixed graph Laplacian eigenvectors. Lastly, our method achieves a better performance compared to other symmetrization methods, and the performance consistently improves with vision pretraining and more samples for testing. As a result, our model based on pretrained ViT and 10 samples for testing achieves 86.650% test accuracy, surpassing all baselines.

Real-world graph learning with vision transformer

Having observed that pretrained ViT can learn graph tasks well when symmetrized with our method, we now provide a preliminary test of it in real-world graph learning. We use three real-world graph datasets from Dwivedi et al. [2022] that involve chemical and biological graphs. PCQM-Contact dataset contains 529,434 molecular graphs with 53 nodes in maximum, and the task is contact map prediction framed as link prediction (S_n-equivariant), on whether two atoms would be proximal when the molecule is in 3D space. Peptides-func/-struct are based on the same set of 15,535 protein graphs with 444 nodes in maximum and the tasks are property prediction (S_n invariant); multi-label classification for Peptides-func and regression for Peptides-struct. The tasks require complex understanding of how the amino acids of

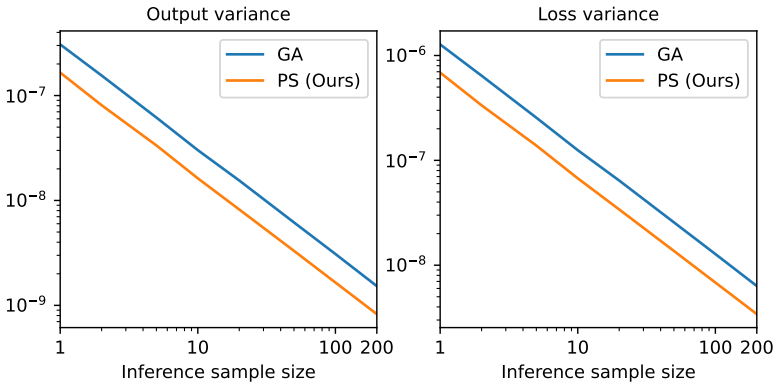


Figure 3.4: Sample variances of the output $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$ (left) and loss $\mathcal{L}(\mathbf{y}, g \cdot f_\theta(g^{-1} \cdot \mathbf{x}))$ (right) of an identical MLP f_θ symmetrized by equivariant distribution (PS) and uniform distribution (GA).

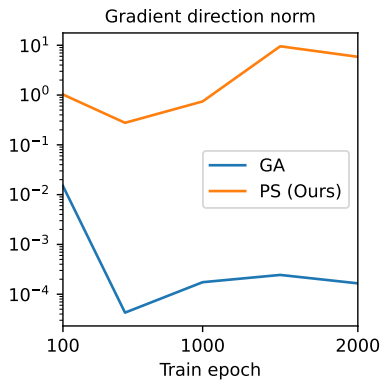


Figure 3.5: Norm of parameter gradients of an identically initialized MLP under PS and GA.

the proteins would interact in 3D space. We implement our method using a ViT-Base pretrained on ImageNet-21k as the base model f_θ and a 3-layer GIN as $p_\omega(g|\mathbf{x})$, following the previous experiment.

The results are in Table 3.4. In Peptides-func and PCQM-Contact, the pretrained ViT symmetrized with our method achieves better performances compared to both previous GNNs and graph transformers[5], largely improving the previous best for PCQM-Contact (0.1355 $\rightarrow$ 0.3287 Hits@1). This shows scalability of our method as Peptides-func involves 444 maximum nodes, and also its generality as it performs competitively for both S_n invariant (Peptides-func) and equivariant (PCQM-Contact) tasks. Unlike some baselines, our method does not require costly Laplacian eigenvectors to compute positional encoding. On Peptides-struct, our method achieves a slightly lower performance to state-of-the-art graph transformers while still better than GNNs. We conjecture that regression is harder for the model to learn due to its stochasticity in predictions, and leave improving regression performance as future work.

In-depth comparison to group averaging

We provide additional analysis of sample variance and convergence, comparing against group averaging. We use the setup for EXP-classify (Section 3.1.3; S_n-invariance) and compare MLP-PS and MLP-GA.

We first analyze whether the equivariant distribution $p_\omega(g|\mathbf{x})$ offers a lower-variance symmetrization, i.e., a better sample efficiency, compared to group averaging with Unif(G). We fix a randomly initialized MLP as f_θ and symmetrize it using both methods. Then, we measure: **(1)** the sample variance of the unbiased estimator $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$ in Equations (3.1) and (3.3), and **(2)** the sample variance of the task loss $\mathcal{L}(\mathbf{y}, g \cdot f_\theta(g^{-1} \cdot \mathbf{x}))$ where $\mathcal{L}$ is binary cross entropy. All measurements are repeated 100 times and averaged over the inputs and labels $(\mathbf{x}, \mathbf{y})$ of the validation dataset. The results are in Figure 3.4, showing that symmetrization with $p_\omega(g|\mathbf{x})$ consistently offers a lower variance than group averaging.

Then we analyze whether the equivariant distribution $p_\omega(g|\mathbf{x})$ for symmetrization offers more stable gradients for the base function f_θ during training compared to group averaging, as conjectured in Section 3.1.1. For this, we fix a randomly initialized MLP f_θ and symmetrize it using both methods and, for every few training epochs, we measure the loss gradient over the entire training dataset with respect to θ. This averages out the variance across datapoints and provides the net direction of the parameter gradient offered by $p_\omega(g|\mathbf{x})$ or Unif(G). The results are in Figure 3.5, showing that symmetrization with

[5]We note that the baseline architectures are constructed within 500k parameter budget as a convention [Dwivedi et al., 2022], while we use an identical architecture to ViT-Base to leverage pretrained representations.

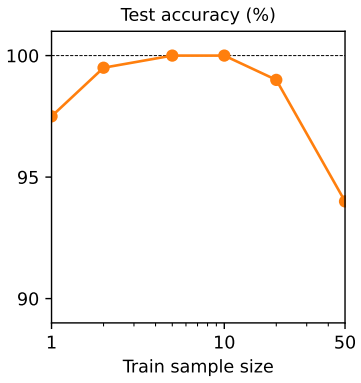


Figure 3.6: Test accuracy of an MLP symmetrized by equivariant distribution $p_\omega(g|\mathbf{x})$ trained on EXP-classify dataset across a range of training sample sizes. Inference sample size is set to 10.

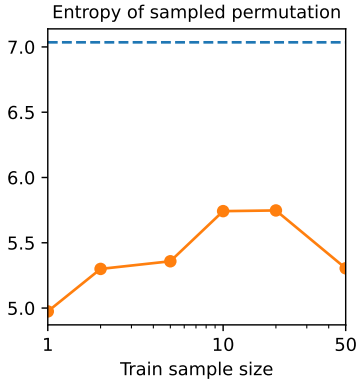


Figure 3.7: Row-/column-wise entropy of aggregated permutation matrices $\mathbf{P}_g \sim p_\omega(g|\mathbf{x})$ after trained on EXP-classify across training sample sizes. Dashed line indicates Unif(G).

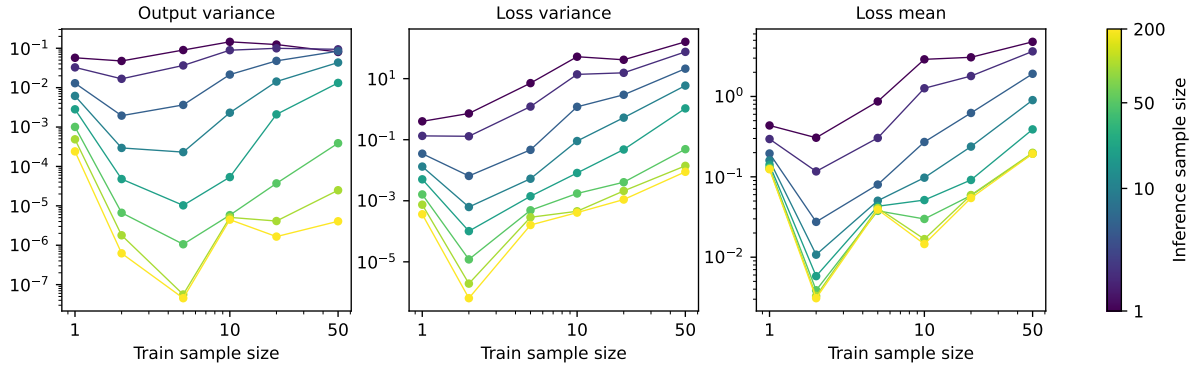


Figure 3.8: Variance of estimation of MLP f_θ symmetrized by equivariant distribution $p_\omega(g|\mathbf{x})$ and trained on EXP-classify for a range of training and inference sample sizes.

$p_\omega(g|\mathbf{x})$ offers a consistently larger gradient magnitude, while group averaging leads to near-zero gradients. This indicates, for Unif(G), the gradients across training data instances are oriented in a divergent manner and thus the training signal is collectively uninformative, while using $p_\omega(g|\mathbf{x})$ for symmetrization leads to more consistent gradient across training data instances, i.e., it offers a more stable training signal.

Effect of sample size on training and inference

We provide additional analysis on how the sample size for estimation of symmetrized function (Equation (3.3)) affects training and inference. We use the setup for EXP-classify (Section 3.1.3; S_n-invariance) and analyze the behavior of MLP-PS with identical initialization and hyperparameters, only controlling sample sizes $\in \{1, 2, 5, 10, 20, 50\}$ for training. We analyze **(1)** variance of permutation matrices $\mathbf{P}_g \sim p_\omega(g|\mathbf{x})$ measured indirectly by the entropy of their aggregation $\bar{\mathbf{P}} = \sum \mathbf{P}_g/N$ as in Section 3.1.3, **(2)** sample variance of the unbiased estimator $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$ of symmetrization (Equation (3.3)), and **(3)** sample mean and variance of the estimated task loss $\mathcal{L}(\mathbf{y}, g \cdot f_\theta(g^{-1} \cdot \mathbf{x}))$ where $\mathcal{L}$ is binary cross entropy. All measurements are repeated 100 times and averaged over the inputs and labels $(\mathbf{x}, \mathbf{y})$ in validation set.

Observations are as follows. First, models trained with smaller samples for symmetrization converge slower, but after sufficient training, all achieve $> 95\%$ test accuracy (Figure 3.6). Second, models trained with smaller samples for symmetrization tend to learn lower-variance symmetrization, not only in terms of

$p_\omega(g|\mathbf{x})$ (Figure 3.7) but also the output and loss (left and center panels of Figure 3.8). This indicates that small sample size serves as a variance regularizer. This is not always beneficial in terms of task loss (right panel of Figure 3.8), as training sample size 1 achieves a poor loss presumably due to over-regularization. That is, sample size for training induces a tradeoff; a small value requires more training iterations, but serves as a variance regularizer and thus a better inference-time efficiency. On the other hand, larger sample sizes for symmetrization during inference is always beneficial (Figure 3.8).

Interestingly, this is consistent with the theoretical claims in literature [Murphy et al., 2019a,b] on the sampling-based training of symmetrized models, which we reprise here. When training the symmetrized model $\phi_{\theta,\omega}(\mathbf{x})$ in Equation (3.3), we cannot directly observe $\phi_{\theta,\omega}(\mathbf{x})$, but observe samples of its unbiased estimator $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$. Thus, it can be questionable what objective we are actually optimizing during training. Based on Murphy et al. [2019a,b], minimizing a convex loss $\mathcal{L}$ on the estimator $g \cdot f_\theta(g^{-1} \cdot \mathbf{x})$ is equivalent to minimizing an upper bound to the true loss on the symmetrized output $\phi_{\theta,\omega}(\mathbf{x})$. This is since our estimation is no longer unbiased for the loss, as we have from Jensen's inequality:

$$\mathbb{E}_{p_\omega(g|\mathbf{x})}[\mathcal{L}(\mathbf{y}, g \cdot f_\theta(g^{-1} \cdot \mathbf{x}))] \geq \mathcal{L}(\mathbf{y}, \mathbb{E}_{p_\omega(g|\mathbf{x})}[g \cdot f_\theta(g^{-1} \cdot \mathbf{x})]) = \mathcal{L}(\mathbf{y}, \phi_{\theta,\omega}(\mathbf{x})).$$

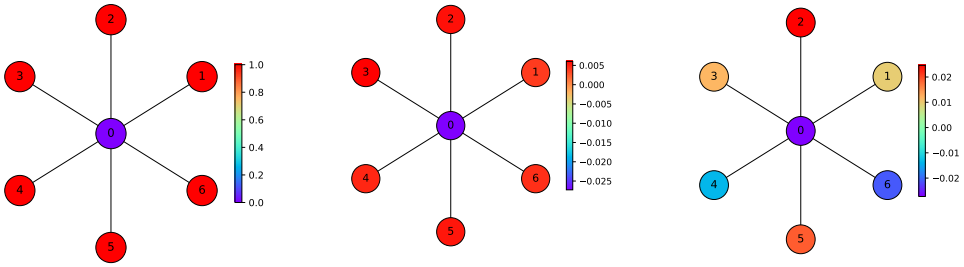
That is, the sampling-based loss is an upper-bound surrogate to the true objective. It has been claimed that optimizing this upper bound has an implicit low-variance regularization effect [Murphy et al., 2019a,b], which is consistent with our observations. This also explains why our distribution $p_\omega(g|\mathbf{x})$ does not collapse to uniform distribution although we do not impose any low-variance regularization explicitly; training to directly minimize the task loss with samples implicitly nudges towards low-variance solutions.

In-depth comparison to canonicalization

We provide additional analysis of our approach in comparison to canonicalization that uses a single group element g from an equivariant canonicalizer $C_\omega : \mathbf{x} \mapsto \rho(g)$. Our claim is that there always exist certain inputs that canonicalization fails to guarantee exact G-equivariance, while our approach guarantees equivariance for all inputs in expectation as in Theorem 9.

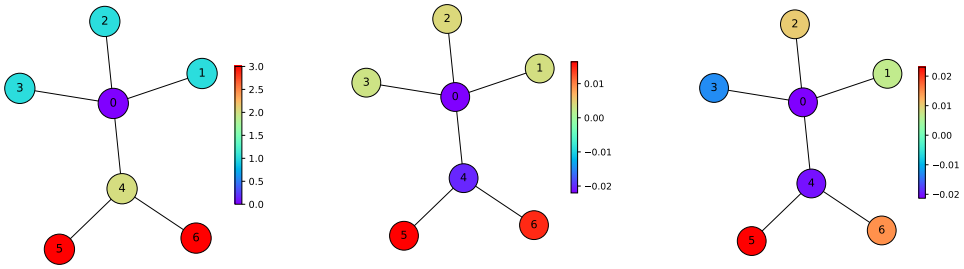
Recall that a canonicalizer C_ω is G-equivariant if $C_\omega(g \cdot \mathbf{x}) = \rho(g)C_\omega(\mathbf{x})$ for all $g \in G$ and $\mathbf{x} \in \mathcal{X}$. Consider an input $\mathbf{x}$ which has a non-trivial stabilizer $G_\mathbf{x}$, i.e., has inner symmetries. It can be shown that equivariant canonicalizers are ill-defined for these inputs. To see this, let $g_1 = gh_1$ and $g_2 = gh_2$ for some $g \in G$ and any $h_1, h_2 \in G_\mathbf{x}$ where $h_1 \neq h_2$. Then we have $C_\omega(g_1 \cdot \mathbf{x}) = C_\omega(gh_1 \cdot \mathbf{x}) = C_\omega(g \cdot \mathbf{x}) = C_\omega(gh_2 \cdot \mathbf{x}) = C_\omega(g_2 \cdot \mathbf{x})$, implying that $\rho(g_1)C_\omega(\mathbf{x}) = \rho(g_2)C_\omega(\mathbf{x})$. Since $g_1 \neq g_2$, this contradicts the group axiom, and thus an equivariant canonicalizer cannot exist for inputs with non-trivial inner-symmetries. To handle all inputs, Kaba et al. [2023] adopts relaxed equivariance: C_ω satisfies relaxed equivariance if $C_\omega(g \cdot \mathbf{x}) = \rho(gh)C_\omega(\mathbf{x})$ up to arbitrary action from the stabilizer $h \in G_\mathbf{x}$. The resulting symmetrization guarantees relaxed equivariance $\phi_{\theta,\omega}(g \cdot \mathbf{x}) = gh \cdot \phi_{\theta,\omega}(\mathbf{x})$ up to arbitrary action from the stabilizer $h \in G_\mathbf{x}$ [Kaba et al., 2023, Theorem A.2], which does not provide equivariance for inputs with inner symmetries.

To visually demonstrate this, we use several graphs $\mathbf{x}$ with non-trivial stabilizers $G_\mathbf{x}$, i.e., inner symmetries, taken from Thiede et al. [2021]. We fix a randomly initialized MLP $f_\theta : \mathbb{R}^{n \times n} \to \mathbb{R}^n$ and symmetrize it using our approach and canonicalization. When symmetrized, the MLP provides scalar node embeddings. In Figures 3.9, 3.10 and 3.11, for each graph we show three panels: the leftmost one shows the color-coding of the inner symmetry (automorphism), the middle one shows node embeddings from MLP-PS, and the rightmost one shows embeddings from MLP-Canonical. If a method is G-equivariant, it is expected to give identical embeddings for automorphic nodes, since an equivariant model cannot



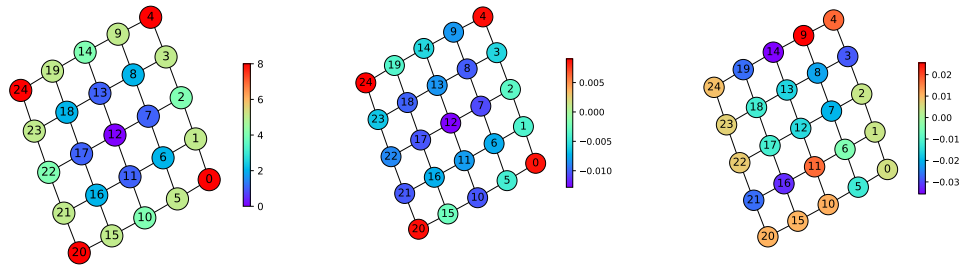
(a) Inner-symmetric nodes (b) MLP-PS (Ours) (c) MLP-Canonical.

Figure 3.9: A graph $\mathbf{x}$ with stabilizer $G_{\mathbf{x}} \cong \mathrm{S}_6$.



(a) Inner-symmetric nodes (b) MLP-PS (Ours) (c) MLP-Canonical.

Figure 3.10: A graph $\mathbf{x}$ with stabilizer $G_{\mathbf{x}} \cong \mathrm{S}_3 \times \mathrm{S}_2$.



(a) Inner-symmetric nodes (b) MLP-PS (Ours) (c) MLP-Canonical.

Figure 3.11: A graph $\mathbf{x}$ with stabilizer $G_{\mathbf{x}} \cong \mathrm{D}_4$.

distinguish them [Srinivasan and Ribeiro, 2020]. In the presence of inner symmetry (left panels), MLP with PS (middle panels) achieves G-equivariance and produces close embeddings for automorphic nodes. However, the same MLP with canonicalization produces relatively unstructured embeddings (right panels). This shows a potential advantage of PS when learning data with inner symmetries, which is often found in applications such as molecular graphs [McKay et al., 2022].

3.1.4 Discussion

We presented probabilistic symmetrization, a general framework that learns a distribution of group transformations conditioned on input data for symmetrization of an arbitrary function. By characterizing that the only condition for such distribution is equivariance to data symmetry, we instantiated models for a wide range of groups, including symmetric, orthogonal, Euclidean groups and their product combinations. Our experiments demonstrated that the proposed framework achieves consistent improvement over other symmetrization methods, and is competitive or outperforms equivariant networks on various datasets. We also showed that transferring pretrained parameters across data in different symmetries can sometimes be surprisingly beneficial.

3.2 Orbit distance minimization

Exploiting symmetries is a popular principle for developing an efficient learning system, which is typically realized by defining a hypothesis class of functions equivariant to a given group G of symmetries. While a dominant approach to define such a hypothesis class has been to design a specific G-equivariant neural-network architecture [Finzi et al., 2021, Villar et al., 2021], *architecture-agnostic* alternatives are explored recently [Puny et al., 2022, Basu et al., 2023, Kaba et al., 2023, Kim et al., 2023]. These alternatives are based on *symmetrization*, where any unconstrained function $\phi_\theta : \mathcal{X} \to \mathcal{Y}$ is made G-equivariant by averaging it over transformations of inputs and outputs induced by certain group elements $g \in G$. In this work, we improve one of the most powerful symmetrization methods from [Kim et al., 2023, Kaba et al., 2023], which symmeterizes ϕ_θ to the following function $\Phi_{\theta,\omega}$:

$$\Phi_{\theta,\omega}(\mathbf{x}) = \mathbb{E}_{\boldsymbol{\epsilon}}[g \cdot \phi_\theta(g^{-1} \cdot \mathbf{x})] \quad \text{with} \quad \rho(g) = r(q_\omega(\mathbf{x}, \boldsymbol{\epsilon})), \tag{3.5}$$

where $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{h} \in \mathbb{R}^{n \times n}$ is a G equivariant network, and $r : \mathbf{h} \mapsto \rho(g)$ is a G equivariant *contraction* operator producing the representation $\rho(g)$ of some element $g \in G$.

A major issue with Equation (3.5) is that designing the contraction r is often non-trivial; r should produce a valid group representation $\rho(g)$ from an unstructured feature $\mathbf{h}$ while being G equivariant itself. Prior works employed hand-designed algorithms, such as Gram-Schmidt process for $\mathrm{O}(n)$ [Kaba et al., 2023], but such algorithms are available only for certain groups [Kim et al., 2023]. Our goal is to overcome this bottleneck and making the symmetrization work for broader group symmetries, such as the Lorentz group $\mathrm{O}(1,3)$.

Our idea is to design a differentiable objective on q_ω of which gradient-based optimization makes $q_\omega(\mathbf{x}, \boldsymbol{\epsilon})$ directly output valid group representations $\mathbf{h} \approx \rho(g)$, thereby removing the need for contraction r. We design the objective in a principled manner as distance minimization on group orbit space. Compared to symmetrization algorithms of Kim et al. [2023], Kaba et al. [2023], this makes our approach applicable to a much broader range of matrix groups where orbit separating invariants are available. We implement our method for the special orthogonal group $\mathrm{SO}(2)$ and the Lorentz group $\mathrm{O}(1,3)$, and find that our objective can replace the known contraction r for $\mathrm{SO}(2)$ with a negligible performance drop, and successfully achieves symmetrization based equivariance on the Lorentz group $\mathrm{O}(1,3)$.

3.2.1 Method

Problem definition Let $\rho : G \to \mathrm{GL}(n)$ be a group representation that associates each group element $g \in G$ an invertible matrix $\rho(g) \in \mathbb{R}^{n \times n}$. For our G-equivariant neural network $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{h} \in \mathbb{R}^{n \times n}$, the group G acts on the output space through the matrix multiplication of the representation $\mathbf{h} \mapsto g \cdot \mathbf{h} = \rho(g)\mathbf{h}$. Our goal is to train q_ω such that its output is always a valid group representation $\mathbf{h} \in \rho(G)$, where $\rho(G)$ denotes the image of ρ.

Orbit distance minimization

We now present a training objective that contracts the output space of q_ω to valid group representations $\rho(G) \subset \mathbb{R}^{n \times n}$. Our key idea is that, instead of working on $\mathbb{R}^{n \times n}$ directly, working on the orbit space (quotient) $\mathbb{R}^{n \times n}/G$ greatly simplifies the problem. Let us write the orbit of an element $\mathbf{h} \in \mathbb{R}^{n \times n}$ under the action of G as $[\mathbf{h}] = \{g \cdot \mathbf{h} : g \in G\}$. The orbit space $\mathbb{R}^{n \times n}/G$ is defined accordingly as $\{[\mathbf{h}] : \mathbf{h} \in \mathbb{R}^{n \times n}\}$.

We now provide an observation that all valid group representations $\rho(G)$ precisely map onto a single point in the orbit space, which is the orbit of the identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$ since we have $\rho(G) = \{\rho(g) : g \in G\} = \{g \cdot \mathbf{I} : g \in G\} = [\mathbf{I}]$. This implies, on the orbit space, our objective is understood as contracting all orbits of neural network outputs $[\mathbf{h}]$ towards a fixed point target $[\mathbf{I}]$. Thus, if we can endow the orbit space with a distance metric $d : \mathbb{R}^{n\times n}/G \times \mathbb{R}^{n\times n}/G \to \mathbb{R}^+$, we can frame our training objective as distance minimization:

$$w^* = \arg\min_{\omega} d([\mathbf{h}], [\mathbf{I}]), \tag{3.6}$$

where we remark that $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{h} \in \mathbb{R}^{n\times n}$ is our G-equivariant neural network. We now show that the objective in Equation (3.6) indeed contracts the output of q_ω exactly to $\rho(G)$.

Theorem 12. *The training objective in Equation* (3.6) *achieves the global minimum with the value of* 0 *if and only if* q_ω *always outputs valid group representations* $\mathbf{h} \in \rho(G)$.

Our problem now reduces to defining a proper distance metric d on the orbit space $\mathbb{R}^{n\times n}/G$. The closest concept we could find in literature is the quotient metric [Burago et al., 2001], but it is intractable as it involves infimum over an infinite set. Instead, we propose a simple distance metric based on a class of functions called *orbit separating invariants*: G-invariant functions f that separate orbits $f(\mathbf{h}) \neq f(\mathbf{h}') \iff [\mathbf{h}] \neq [\mathbf{h}']$ [Dym and Gortler, 2022]. In detail, our distance metric can be defined as vector distance on outputs of f:

Theorem 13. *Let* $f : \mathbb{R}^{n\times n} \to \mathbb{R}^k$ *be an orbit separating invariant and* $\|\cdot\|$ *be vector norm. Then,* $d([\mathbf{h}], [\mathbf{h}']) = \|f(\mathbf{h}) - f(\mathbf{h}')\|$ *is a distance metric on the orbit space* $\mathbb{R}^{n\times n}/G$.

With Theorem 13, we can use the below objective for optimization problem in Equation (3.6):[6]

$$w^* = \arg\min_{\omega} \|f(\mathbf{h}) - f(\mathbf{I})\|. \tag{3.7}$$

In practice, it is desirable to have a differentiable objective such that we can perform a gradient-based optimization. Since $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{h}$ is already a neural network, the objective in Equation (3.7) would be differentiable almost everywhere with respect to ω if we choose f and $\|\cdot\|$ to be differentiable almost everywhere.

Learning symmetrization with orbit distance minimization

Our original goal is to symmetrize a function $\phi_\theta : \mathcal{X} \to \mathcal{Y}$ to be G-equivariant. We define our symmetrization as follows by removing contraction r from Equation (3.5):

$$\Phi_{\theta,\omega}(\mathbf{x}) = \mathbb{E}_{\mathbf{h}}[\mathbf{h} \cdot \phi_\theta(\mathbf{h}^{-1} \cdot \mathbf{x})] \quad \text{where} \quad \mathbf{h} = q_\omega(\mathbf{x}, \boldsymbol{\epsilon}).$$

Given training pairs $(\mathbf{x}, \mathbf{y})$ of input $\mathbf{x} \in \mathcal{X}$ and label $\mathbf{y} \in \mathcal{Y}$, we train for the joint objective of task loss $\mathcal{L}$ and the orbit distance loss (Equation (3.7)) weighted by a hyperparameter λ:

$$\theta^*, \omega^* = \arg\min_{\theta,\omega} \mathcal{L}(\mathbf{y}, \Phi_{\theta,\omega}(\mathbf{x})) + \lambda\, \mathbb{E}_{\mathbf{h}}\|f(\mathbf{h}) - f(\mathbf{I})\|. \tag{3.8}$$

[6] For some groups, orbit separation of f is guaranteed only for full-rank inputs [Dym and Gortler, 2022]. In this case, the optimality condition in Theorem 12 holds if we assume $\mathbf{h}$ to be full-rank.

Table 3.5: Orbit separating invariants for some group actions from Dym and Gortler [2022], along with the domain on which orbit separation is guaranteed. For the general linear group $\mathrm{GL}(n)$, the generating set consists solely of constant polynomial, and an alternative choice of non-trivial orbit separating invariant from Dym and Gortler [2022] is shown.

Group	Domain	Orbit separating invariant	Dimension
S_n	$\mathbb{R}^{n\times n}$	$[\phi_\alpha(\mathbf{h}) = \sum_{j=1}^n \mathbf{h}_j^\alpha], \quad \alpha \in \mathbb{Z}^d_{\geq 0}, \lvert\alpha\rvert \leq n$	$\binom{2n}{n}$
$\mathrm{O}(n)$	$\mathbb{R}^{n\times n}$	$[\mathrm{vec}(\mathbf{h}^\top\mathbf{h})]$	n^2
$\mathrm{SO}(n)$	$\mathbb{R}^{n\times n}$	$[\mathrm{vec}(\mathbf{h}^\top\mathbf{h}), \det\mathbf{h}]$	n^2+1
$\mathrm{O}(1,n-1)$	$\mathbb{R}^{n\times n}_{\text{full}}$	$[\mathrm{vec}(\mathbf{h}^\top\mathbf{\Lambda}\mathbf{h})], \quad \mathbf{\Lambda} = \mathrm{diag}([+1,-1,...,-1])$	n^2
$\mathrm{SL}(n)$	$\mathbb{R}^{n\times n}_{\text{full}}$	$[\det\mathbf{h}]$	1
$\mathrm{GL}(n)$	$\mathbb{R}^{n\times n}_{\text{full}}$	$[\det{}^2(\mathbf{h}\mathbf{W}_i)/\det{}^{-1}(\mathbf{h}\mathbf{h}^\top)], \quad \mathbf{W}_i \in \mathbb{R}^{n\times n}, i = 1,...2n^2+1$	$2n^2+1$

Intuitively, if the orbit loss is ≈ 0, we would have $\mathbf{h} \approx \rho(g)$ and the model would closely achieve the symmetrization in Equation (3.5) while not requiring contraction $r : \mathbf{h} \mapsto \rho(g)$.

Generality of orbit separating invariants

We now discuss the results from invariant theory implying orbit separating invariants $f : \mathbb{R}^{n\times n} \to \mathbb{R}^k$ of bounded dimension $k \leq 2n^2+1$ exist for a very general class of matrix groups and they are differentiable everywhere in general.

The existence of orbit separating invariants has been mainly shown for *linearly reductive groups* including $\mathrm{GL}(n)$, semi-simple groups $\mathrm{SL}(n)$, $\mathrm{O}(n)$, $\mathrm{SO}(n)$, finite group S_n, and also $\mathrm{O}(s, n-s)$. Most of the results are derived from the concept of *invariant polynomials*, which are polynomials on matrix entries that are G-invariant. To elaborate, consider a group G acting on $\mathbb{R}^{n\times n}$, and let $\mathcal{S}$ be the set of all invariant polynomials $f' : \mathbb{R}^{n\times n} \to \mathbb{R}$. We call a set of invariant polynomials $\{f_1, ..., f_k\}$ the *generating set* if every $f' \in \mathcal{S}$ can be written as $f'(\cdot) = h(f_1(\cdot), ..., f_k(\cdot))$ using some polynomial $h : \mathbb{R}^k \to \mathbb{R}$. For every linearly reductive group, Weyl's theorem [Weyl, 1946] guarantees the existence of a finite generating set. Furthermore, for many subclasses of these groups, it has been shown that this set separates orbits in $\mathbb{R}^{n\times n}$ whose closures do not intersect[7] [Dym and Gortler, 2022, Derksen and Kemper, 2015], allowing us to use their stack $f : \mathbb{R}^{n\times n} \to \mathbb{R}^k$ as our orbit separating invariant. Such f is differentiable everywhere as it is a stack of polynomials. For many groups, the generating set is known from the invariant theory, and so is f; some known examples are provided in Table 3.5.

Now consider the dimension k of the separating invariant $f : \mathbb{R}^{n\times n} \to \mathbb{R}^k$ which is the size of the generating set $\{f_1, ..., f_k\}$ in our context. While this can be large for some groups, Dym and Gortler [2022] has shown that random linear projection can almost always reduce it to a set of $2n^2+1$ polynomials that still separates orbits, which also reduces f while preserving differentiability.

3.2.2 Experiments

We evaluate our approach on two selected matrix groups: the special orthogonal group in two dimensions $\mathrm{SO}(2)$ and the Lorentz group $\mathrm{O}(1,3)$. For the $\mathrm{SO}(2)$ group, an equivariant contraction operator $r : \mathbf{h} \mapsto \rho(g)$ has been designed by Kim et al. [2023], and we test whether orbit distance

[7] For compact groups this guarantees orbit separation; for closed non-compact groups this guarantees orbit separation for full-rank inputs [Dym and Gortler, 2022] which relates to footnote 6.

Table 3.6: Experimental results on SO(2) and O(1, 3) group symmetries.

(a) Rotated MNIST, SO(2).

	Test Error % ↓
GCNN (p4)	2.36 ± 0.15
GCNN (p64)	2.28 ± 0.10
CNN	4.90 ± 0.20
CNN-Aug.	3.30 ± 0.20
CNN-Canonical.	2.32 ± 0.18
CNN-PS	2.21 ± 0.28
CNN-Canonical.-Orbit **(Ours)**	2.44 ± 0.12
CNN-PS-Orbit **(Ours)**	2.37 ± 0.35

(b) Particle Scattering, O(1, 3).

	Test MSE ↓
Scalar MLP	0.00171 ± 0.00004
MLP	0.65381 ± 0.23663
MLP-Aug.	0.09101 ± 0.03107
MLP-Canonical.	N/A
MLP-PS	N/A
MLP-Canonical.-Orbit **(Ours)**	0.01027 ± 0.00082
MLP-PS-Orbit **(Ours)**	0.00887 ± 0.00070

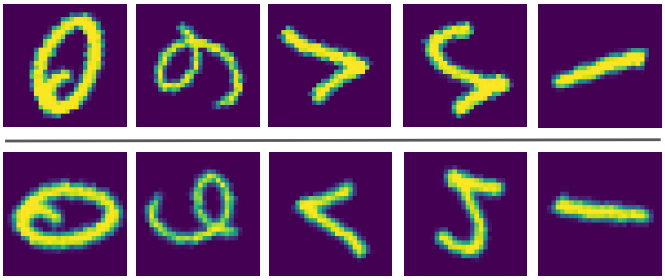


Figure 3.12: Example transformed images for Rotated MNIST. The first row shows images from the original dataset, while the second row shows them transformed by output of our learned symmetrizer $q_\omega(\mathbf{x}, \boldsymbol{\epsilon})$. It is clear from these figures that transformations associated with $q_\omega(\mathbf{x}, \boldsymbol{\epsilon})$ is purely rotation.

minimization can replace it. For the Lorentz group O(1, 3), the contraction operator is not known, so our experiments offer the first results on symmetrization based equivariance to our knowledge.

Rotation-invariant image classification

For SO(2), we use the Rotated MNIST [Larochelle et al., 2007], a common benchmark for equivariant models [Cohen and Welling, 2016, Finzi et al., 2020] with randomly rotated 62,000 digits (SO(2) invariance). We follow the setup of Kaba et al. [2023] and use the same 7-layer CNN as our base function ϕ_θ. For the symmetrizer q_ω, we use 2-layer EMLP [Finzi et al., 2021] of 64 hidden dimensions. For training (Equation (3.8)), we use cross entropy for task loss $\mathcal{L}$; for orbit distance loss $\lambda \|f(\mathbf{h}) - f(\mathbf{I})\|$ we use the orbit separating invariant $f(\mathbf{h}) = [\text{vec}(\mathbf{h}^\top \mathbf{h}), \det(\mathbf{h})]$ [Dym and Gortler, 2022], L1 norm, $\lambda = 1$.

We use the following baselines: CNN, CNN with SO(2) data augmentation, equivariant GCNN [Cohen and Welling, 2016], and CNN made SO(2) equivariant with symmetrization methods Canonicalization [Kaba et al., 2023] and Probabilistic Symmetrization (PS) [Kim et al., 2023] that follow Equation (3.5) except Canonicalization drops noise $\boldsymbol{\epsilon}$. We take the performances of CNN, data augmentation, and GCNN from Kaba et al. [2023], and train symmetrization baselines using SO(2) contraction of Kim et al. [2023]. The results are in Table 3.6a and Figure 3.12. Symmetrization improves CNN overall, as all symmetrized CNNs outperform data augmentation and perform on par with GCNN. Within symmetrization, replacing contraction $r : \mathbf{h} \mapsto \rho(g)$ with our orbit distance loss leads to a negligible drop in performance. This indicates orbit distance minimization can replace the role of contraction operator, with a slight tradeoff as q_ω takes an extra role of producing valid group representation $\mathbf{h} \approx \rho(g)$.

Lorentz-invariant particle scattering regression

For the Lorentz group $\mathrm{O}(1,3)$, we use Particle Scattering synthetic regression dataset from Finzi et al. [2021] for matrix element in electron muon scattering ($\mathrm{O}(1,3)$ invariance). We use 10,000 train data, and use 1,000 validation and 1,000 test data that are randomly $\mathrm{O}(1,3)$ transformed. We use 3-layer MLP of 128 hidden dimensions as our base function ϕ_θ, and our symmetrizer q_ω is based on 3-layer Scalar MLP [Villar et al., 2021] of 128 hidden dimensions. For training (Equation (3.8)), we use mean squared error for task loss $\mathcal{L}$; for orbit distance loss $\lambda \|f(\mathbf{h}) - f(\mathbf{I})\|$ we use the orbit separating invariant $f(\mathbf{h}) = [\mathrm{vec}(\mathbf{h}^\top \mathbf{\Lambda} \mathbf{h})]$, $\mathbf{\Lambda} = \mathrm{diag}([+1, -1, -1, -1])$ [Dym and Gortler, 2022], L1 norm, $\lambda = 1$.

We run the following baselines: MLP, MLP with $\mathrm{O}(1,3)$ data augmentation, and invariant Scalar MLP [Villar et al., 2021], with 3 layers of 128 hidden dimensions. Symmetrization baselines in Equation (3.5) cannot be built as $\mathrm{O}(1,3)$ contraction is not known. The results are in Table 3.6b. $\mathrm{O}(1,3)$ symmetrized MLPs based on our method significantly outperform MLP as well as data augmentation. To our knowledge, this is the first successful result in symmetrization based equivariance for $\mathrm{O}(1,3)$, implying symmetrization can be applied to groups where contraction $r : \mathbf{h} \mapsto \rho(g)$ is not available. Yet, our method has performance gap from invariant Scalar MLP. An explanation is that Scalar MLP gets Minkowsky inner product $\mathbf{x}^\top \mathbf{\Lambda} \mathbf{x}$ as input which is heavily correlated to label function of particle scattering, but our base MLP gets transformed input $\approx \rho(g)^{-1}\mathbf{x}$ which requires further processing. We leave closing this gap a future work.

3.2.3 Discussion

We proposed orbit distance minimization, a framework for symmetrization based equivariant learning. Our method is competitive on $\mathrm{SO}(2)$ invariant classification and successfully achieves symmetrization based equivariance on Lorentz group $\mathrm{O}(1,3)$.

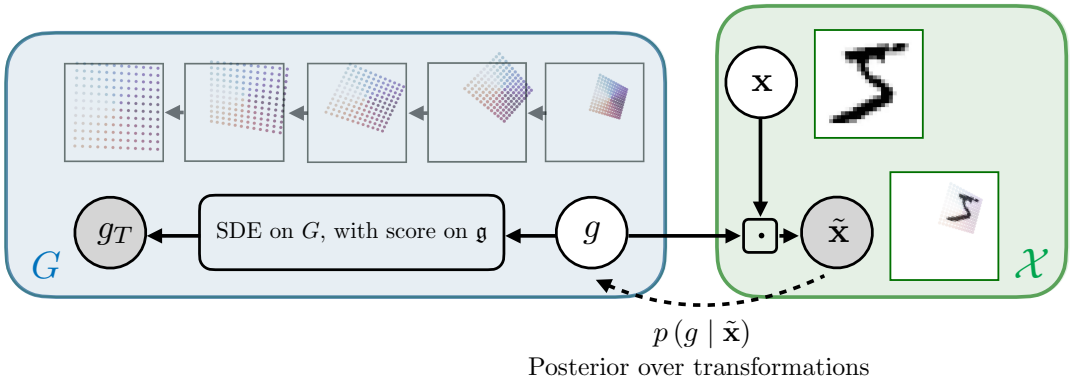


Figure 3.13: Graphical model describing our problem and method (with observed variables in gray and unobserved variables in white). $\mathcal{X}$ denotes the data space and G a group of transformations, here the group of image homographies $\mathrm{PGL}(3, \mathbb{R})$. We are provided a data sample $\tilde{\mathbf{x}}$ that is generated by transforming an unknown in-distribution sample $\mathbf{x}$ with an unknown transformation g. We wish to sample from the posterior over transformations. Inspired by diffusion models, we construct a fast sampler that reverts a diffusion process on the Lie group starting from a random group element. The scores of the stochastic differential equation (SDE) are computed in the Lie algebra $\mathfrak{g}$.

3.3 Transformation-inverting energy diffusion

Many data modalities are observed after unknown or unusual geometric transformations. In this work, we ask a general question: given data transformed by an unknown element of a Lie group, can we recover an inverse transformation that returns it to the original distribution? This problem appears in many scenarios, including viewpoint changes and projective warps in computer vision, sensor motion and registration in medical imaging, and changes of reference frame in scientific modeling [Olver, 1993, Hartley and Zisserman, 2003, Celledoni et al., 2021]. It is an instance of a blind inverse problem as the nuisance transform is unknown; Probabilistic methods are therefore a natural fit [Kaipio and Somersalo, 2005]. Unlike existing approaches that sample in data space [Chung et al., 2022] or are specialized to particular transformation families [Zitova and Flusser, 2003, Beg et al., 2005, Hartley et al., 2013], we tackle the problem for general Lie group actions, with all inference and sampling done on the group.

We model inverse transformations using energy-based models that quantify the likelihood of different transformations of the data. This allows us to leverage pretrained energy models or energies designed using domain expertise. Sampling from these models can however be slow and challenging, especially when the energy landscape is rugged and multimodal. To address this, we present a new sampling method on Lie groups based on diffusion [Song et al., 2020], which follows scores along a noise schedule. We show that the sampling can be performed on the group by estimating scores in the Lie algebra (Figure 3.13), extending trivializations for handling curved manifolds [Lezcano Casado, 2019, Zhu et al., 2025]. Our method covers a very general class of groups without assuming compactness, a bi-invariant metric, or linear actions.

As a key application, we focus on the case where the transformed data is the input to a neural network. It is known that pretrained networks can exhibit a marked lack of robustness to geometric transformations [Hendrycks and Dietterich, 2019, Ollikka et al., 2024]. Equivariant neural networks address this for some groups and modalities [Cohen and Welling, 2016, Bronstein et al., 2021], but require highly specialized architectures that can be difficult to scale. Instead of using equivariant models, we wish to improve the robustness of generic, pretrained networks. The ability to invert data transformation ensures that we can

bring samples back in-distribution and that the model does not perform inference on out-of-distribution transformed data. Prior methods based on deterministic canonicalization [Jaderberg et al., 2015, Esteves et al., 2017, Kaba et al., 2023] pursue this idea but either rely on training additional equivariant networks [Mondal et al., 2023] or use optimization methods [Schmidt and Stober, 2024, Shumaylov et al., 2024, Singhal et al., 2025] which can be brittle and do not readily extend to general Lie groups.

Our method allows to make any pretrained model equivariant to Lie groups at test-time with access to an energy-based model. The energy can be approximated via the pretrained model itself [Grathwohl et al., 2019, Singhal et al., 2025], which results in a training-free pipeline. This can be interpreted as a form of inference-time scaling for equivariance. The inference-time cost comes from sampling in-distribution transformations, which is made efficient and fast using our novel diffusion sampler.

Contributions Our contributions are as follows:

1. We introduce Transformation-Inverting Energy Diffusion (TIED), a novel diffusion sampler for inverting data transformations. It is guaranteed to stay on manifold and handles general Lie groups and nonlinear actions, only requiring knowledge of Lie algebra.

2. We prove that inversion of data transformations can be used for test-time equivariance of pretrained models only requiring an approximate energy model.

3. We show experimentally that our method can invert transformations on challenging groups such as image homographies and Lie point symmetries for partial differential equations (PDEs), where our method consistently outperforms baselines in improving the performance of pretrained networks.

3.3.1 Related work

Inverse problems on groups Recovering unknown transformations has a long history in problems that are naturally formulated on a group. In alignment and rotation synchronization, the aim is to recover poses on orthogonal or Euclidean groups from relative measurements [Singer, 2011, Hartley et al., 2013]. Related are also image registration problems in which one seeks to recover the affine or homography transform between pairs of images [Zitova and Flusser, 2003, Beg et al., 2005, Lorenzi and Pennec, 2013]. By contrast to our setting, in these problems one is often provided pairs of samples, as opposed to a single one. In addition, these methods typically rely on group-specific knowledge. A recent line of work proposes to use generative models to recover inverse samples [Venkatakrishnan et al., 2013, Kadkhodaie and Simoncelli, 2020, Chung et al., 2022]. Since they operate in data space instead of the group, they require large diffusion models and for non-linear actions do not recover transformations on the manifold.

Test-time equivariance Canonicalization methods improve robustness of neural networks by mapping inputs to a reference pose before prediction [Jaderberg et al., 2015, Esteves et al., 2017, Kaba et al., 2023, Mondal et al., 2023]. These methods offer an alternative to equivariant networks [Cohen and Welling, 2016, Bronstein et al., 2021] and data augmentation [Shorten and Khoshgoftaar, 2019, Cubuk et al., 2019] that does not require specialized architectures or more expensive training procedures. Canonicalization yields equivariance for any predictor if the canonicalization function is itself equivariant. Previous works have proposed probabilistic variants [Kim et al., 2023, Dym et al., 2024, Cornish, 2024, Lawrence et al., 2025], but they rely on equivariant networks and are therefore limited in applications. Most closely related to our method are optimization-based canonicalization methods [Kaba et al., 2023, Schmidt and Stober,

2024, Shumaylov et al., 2024, Singhal et al., 2025], which obtain an equivariant canonicalizer without requiring equivariant primitives. They are, however, purely deterministic, often specialize in specific groups and necessitate optimization of highly rugged energy functions. Test-time data augmentation methods are an alternative [Krizhevsky et al., 2012, Szegedy et al., 2015, Wang et al., 2019a, Shanmugam et al., 2020, Kim et al., 2020b], and improve prediction through an ensembling effect [Hansen and Salamon, 2002]. They, however, lack equivariance guarantees and use heuristic augmentation distributions. Group averaging [Yarotsky, 2022] and its frame-based variant [Puny et al., 2022] guarantee equivariance, yet are often intractable for Lie groups and may average over out-of-distribution transformations.

Diffusion sampling Diffusion models [Sohl-Dickstein et al., 2015, Ho et al., 2020, Song et al., 2020] are a class of generative models that produce samples by first sampling from a simple base distribution $p_1(\mathbf{x})$ and integrating along the stochastic differential equation (SDE)

$$\mathrm{d}\mathbf{x}_t = -\gamma(t)^2 \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\,\mathrm{d}t + \gamma(t)\,\mathrm{d}\bar{\mathbf{w}}_t,$$

where $t \in [0,1]$, $\gamma(t)$ is a scalar diffusion coefficient, $\bar{\mathbf{w}}_t$ is Brownian motion run backwards in time. $p_t(\mathbf{x}_t)$ is a marginal 'noisy' distribution under the forward process

$$\mathrm{d}\mathbf{x}_t = \gamma(t)\,\mathrm{d}\mathbf{w}_t, \tag{3.9}$$

starting from the target distribution $p_0(\mathbf{x}) \equiv p(\mathbf{x})$. We consider here the variance-exploding (VE) SDE. The scores $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ can be estimated from data [Vincent, 2011]. A key advantage of diffusion sampling is that the scores of the noisy densities are smoother than those of the original density and non-vanishing in regions of low energy [Song and Ermon, 2019]. The success of diffusion models has inspired a variety of methods for sampling from Boltzmann distributions using reverse SDEs; see, e.g., Richter and Berner [2023], Vargas et al. [2023], Akhound-Sadegh et al. [2024], De Bortoli et al. [2024]. These methods enable the estimation of scores along the probability path, assuming access to the energy and its gradients, rather than samples. We discuss later extensions to Lie groups, which are relevant to our problem. Diffusion models over transformations have also been proposed [Bansal et al., 2023], but in contrast to us, use deterministic non-invertible transformations.

3.3.2 Preliminaries

Lie groups We consider a set of transformations G forming a connected Lie group. While a Lie group is a curved manifold, a lot of its properties derive from its tangent space at the identity, the Lie algebra, denoted by $\mathfrak{g} \equiv T_e G$. We denote by $\exp : \mathfrak{g} \to G$ the exponential map and by μ the (unnormalized) left-invariant Haar measure on G. The left multiplication map $L_g : G \to G$ is defined as $L_g : h \mapsto gh$, and its tangent map $\mathrm{d}(L_g)_h : T_h G \to T_{gh} G$ is one-to-one. We highlight two cases:

$$\mathrm{d}(L_g)_e : \mathfrak{g} \to T_g G, \quad \mathrm{d}(L_{g^{-1}})_g : T_g G \to \mathfrak{g}.$$

These tangent maps are useful for working in the Lie algebra instead of the tangent spaces of arbitrary group elements, a technique called *(left-)trivialization* in literature [Lezcano Casado, 2019, Kong and Tao, 2024]. A particular example we frequently use is the trivialized gradient operator $\nabla_{\mathfrak{g}} \equiv \mathrm{d}(L_{g^{-1}})_g \nabla$, which takes a function on G and evaluates its gradient in $\mathfrak{g}$. Notably, it can be computed with standard automatic differentiation tools without the need to explicitly handle tangent maps. We refer the reader

to Appendix A.6.1 for additional background on Lie groups and trivialization.

Group actions We consider data samples $\mathbf{x} \in \mathcal{X}$ and outputs $\mathbf{y} \in \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y} \subseteq \mathbb{R}^d$. The data density is denoted $p_{\mathbf{x}}(\mathbf{x})$. We assume a diffeomorphic but not necessarily linear action of $g \in G$ on data samples $\phi_g(\mathbf{x})$ also denoted by $g \cdot \mathbf{x}$. The Jacobian of the group action ϕ_g evaluated at $\mathbf{x}$ is denoted by $J_g(\mathbf{x})$. For linear actions, J_g is a group representation independent of $\mathbf{x}$. The orbit of $\mathbf{x}$ is the set of samples that can be obtained through transformations and is denoted by $G \cdot \mathbf{x}$. We denote by $\mu_{G\cdot\mathbf{x}}$ the pushforward of the Haar measure using $\phi_g(\mathbf{x})$. A function $f : \mathcal{X} \to \mathcal{Y}$ is equivariant if $f(g \cdot \mathbf{x}) = g \cdot f(\mathbf{x})$ for all $g \in G$, $\mathbf{x} \in \mathcal{X}$ and invariant if $f(g \cdot \mathbf{x}) = f(\mathbf{x})$. Similarly, a conditional density is equivariant if $p(g \cdot \mathbf{y} \mid g \cdot \mathbf{x}) = p(\mathbf{y} \mid \mathbf{x}) |\det J_g(\mathbf{y})|^{-1}$ for all $g \in G$, $\mathbf{x} \in \mathcal{X}$, $\mathbf{y} \in \mathcal{Y}$.

3.3.3 Method

Probabilistic transformation inversion

We now formally introduce the problem of transformation inversion. We assume that we are given an out-of-distribution sample $\tilde{\mathbf{x}} = g \cdot \mathbf{x}$, generated by applying an unknown transformation g on an in-distribution sample $\mathbf{x}$. Our aim is to solve the blind inverse problem and recover the pair $(g, \mathbf{x})$. Since this problem does not admit a unique solution, a probabilistic approach is a natural choice. Assuming a uniform prior over the unknown transformation, a classical solution to inverse problems [e.g., Stuart, 2010] is to consider the Bayesian posterior

$$p(\mathbf{x} \mid \tilde{\mathbf{x}}) = \frac{p_{\mathbf{x}}(\mathbf{x})\, p(\tilde{\mathbf{x}} \mid \mathbf{x})}{p(\tilde{\mathbf{x}})} \propto p_{\mathbf{x}}(\mathbf{x})\, \mathbb{1}_{G\cdot\mathbf{x}}(\tilde{\mathbf{x}}),$$

where the density is with respect to $\mu_{G\cdot\mathbf{x}}$ and $\mathbb{1}_{G\cdot\mathbf{x}}$ is the indicator function over the orbit of $\mathbf{x}$.

Rather than modeling the posterior in data space, we can alternatively model the posterior $p(g \mid \tilde{\mathbf{x}})$ directly on the group. This has the advantage that the group is (often significantly) lower dimensional than the data space, and that this allow to recover the inverse transformation g. We can show that the posterior is given by the following.

Proposition 7 (Transformation inversion posterior)**.** *Let G act freely on $\mathcal{X}$. Then,*

1. *The posterior distribution of g has density $p(g \mid \tilde{\mathbf{x}}) \propto p_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) |\det J_{g^{-1}}(\tilde{\mathbf{x}})|$.*
2. *The random variable $\mathbf{x}' = h^{-1} \cdot \tilde{\mathbf{x}}$, with $h \sim p(g \mid \tilde{\mathbf{x}})$ has density $p(\mathbf{x}' \mid \tilde{\mathbf{x}}) \propto p_{\mathbf{x}}(\mathbf{x}')\, \mathbb{1}_{G\cdot\mathbf{x}}(\tilde{\mathbf{x}})$ with respect to $\mu_{G\cdot\mathbf{x}}$.*

All the proofs appear in Appendix A.6.2. This result shows that when a prior model of data distribution $p_{\mathbf{x}}$ is available, the posterior over transformations can be directly recovered. Additionally, sampling $h \sim p(g \mid \tilde{\mathbf{x}})$ and canonicalizing the out-of-distribution sample via $h^{-1} \cdot \tilde{\mathbf{x}}$ yields samples from the prior data distribution, which conforms to intuition.

Writing the posterior distribution over g as a Boltzmann density, we have

$$p(g \mid \tilde{\mathbf{x}}) = \frac{e^{-E_{\mathbf{x}}\left(g^{-1}\cdot\tilde{\mathbf{x}}\right)} |\det J_{g^{-1}}(\tilde{\mathbf{x}})|}{Z(\tilde{\mathbf{x}})}, \quad Z(\tilde{\mathbf{x}}) = \int_G e^{-E_{\mathbf{x}}\left(g^{-1}\cdot\tilde{\mathbf{x}}\right)} |\det J_{g^{-1}}(\tilde{\mathbf{x}})| \mathrm{d}\mu(g), \tag{3.10}$$

where $E_{\mathbf{x}}(\mathbf{x}) = -\log p_{\mathbf{x}}(\mathbf{x})$ is the energy associated with the data prior. Under mild conditions on the energy, the normalization constant $Z(\tilde{\mathbf{x}})$ is finite even for non-compact groups and the density is

well-defined. Note that any function of $\tilde{\mathbf{x}}$ can be added to the energy without changing the density, since it will cancel in the normalization. Therefore, the energy does not need to be accurate across orbits, but only within each orbit. This is significantly simpler than modeling the data space and is closer to a generative model over transformations [Allingham et al., 2024].

Application to test-time equivariance

We now consider an application of our framework in improving the robustness of a pretrained neural network $f_\theta : \mathcal{X} \to \mathcal{Y}$. A common failure mode of neural networks, including those trained on large datasets and with data augmentation, is their poor generalization to transformed samples [Hendrycks and Dietterich, 2019]. By investing a small amount of compute at test-time to sample an inverse transformation, we aim to improve the pretrained model.

Following the canonicalization method [Kaba et al., 2023], we define the test-time predictor $\tilde{f}_\theta$ as

$$\tilde{f}_\theta(\tilde{\mathbf{x}}) = g \cdot f\left(g^{-1} \cdot \tilde{\mathbf{x}}\right), \quad g \sim p(g \mid \tilde{\mathbf{x}}),$$

where input data to the pretrained model $g^{-1} \cdot \tilde{\mathbf{x}}$ is guaranteed to lie in distribution by Proposition 7. Results by Bloem-Reddy and Teh [2020], Lawrence et al. [2025] show that this randomized predictor is equivariant if $p(g \mid \tilde{\mathbf{x}})$ is an equivariant conditional distribution. The natural question is therefore, does the transformation inversion posterior satisfy conditional equivariance? This is indeed the case, which ensures the soundness of our approach.

Proposition 8 (Equivariance of the posterior)**.** *For any $E_{\mathbf{x}} : \mathcal{X} \to \mathbb{R}$, the posterior density $p(g \mid \tilde{\mathbf{x}})$ is G-equivariant.*

The intuition for this is that if the sample $\tilde{\mathbf{x}}$ is transformed, then the posterior must change in an opposite way to ensure that $g^{-1} \cdot \tilde{\mathbf{x}}$ would still be distributed according to the data prior.

Following Kim et al. [2023], we can also sample an ensemble of diverse in-distribution samples and average predictions over them when $p(g \mid \tilde{\mathbf{x}})$ is equivariant:

$$\tilde{f}(\tilde{\mathbf{x}}) = \mathbb{E}_{g \sim p(g \mid \tilde{\mathbf{x}})}\left[g \cdot f\left(g^{-1} \cdot \tilde{\mathbf{x}}\right)\right].$$

It has been shown that ensembling over augmentations can further improve the performance of pretrained neural networks [Shanmugam et al., 2020]. Our method provides a principled form for the augmentation distribution $p(g \mid \tilde{\mathbf{x}})$, whereas most methods use distributions motivated by heuristics.

There is significant flexibility in the choice of the energy function $E_{\mathbf{x}}$. Following optimization-based canonicalization methods [Schmidt and Stober, 2024, Singhal et al., 2025], we consider defining the energy directly from the pretrained model f_θ using its prediction confidence if it is a classifier [Grathwohl et al., 2019]. We also consider the evidence lower-bound (ELBO) approximation of the energy using variational autoencoders (VAEs) [Kingma and Welling, 2014, Shumaylov et al., 2024].

Challenges in sampling from general Lie groups

The next question is how to sample from the transformation-inversion distribution Equation (3.10), assuming we have access to an energy-based model and can take its gradient. This is challenging for two reasons. First, the energy landscape can be highly rugged and multi-modal, especially when the energy is derived from a neural network [Montúfar et al., 2014]. This leads to slow mixing for MCMC, even

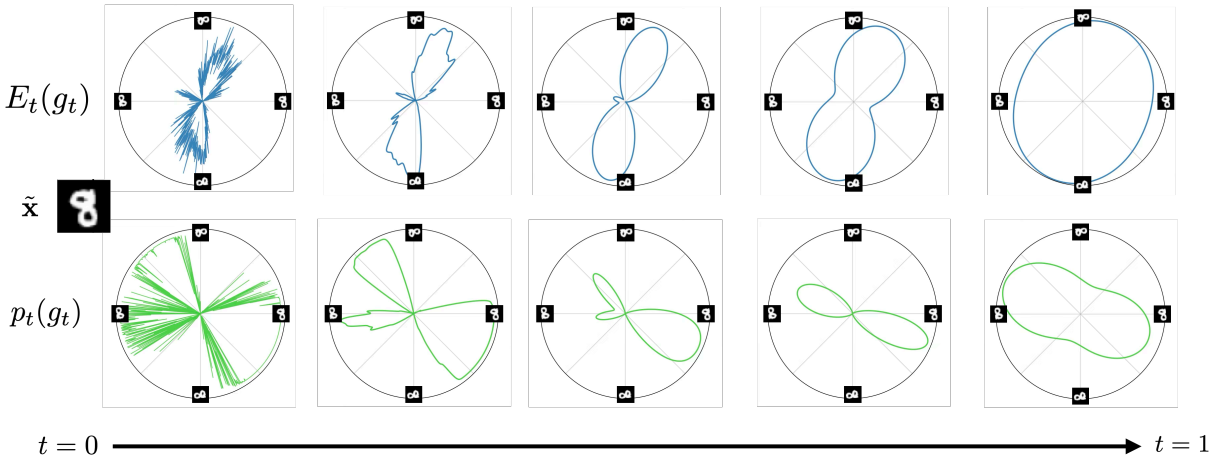


Figure 3.14: Energy (top) and density (bottom) along the forward process Equation (3.11) for the group of rotations $G = \mathrm{SO}(2)$. The energy of the prior $E_0(g) \equiv E_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right)$ is defined using the LogSumExp of classifier logits from a ResNet18 trained on MNIST. The energy at small timesteps (top left) is low for likely orientations of the MNIST digit $\tilde{\mathbf{x}}$. Note the multimodal nature of the posterior $p_0(g)$ (bottom left), with modes centered around the most likely transformations $g = -20°, 135°, 180°$. Since the energy is obtained from a neural network, its landscape is highly rugged, resulting in exploding and vanishing scores. Going to the right, the densities along the forward process are plotted. The landscapes are increasingly smooth and address the ruggedness issue.

for gradient-based methods such as Langevin sampling [Roberts and Tweedie, 1996, Welling and Teh, 2011]. Second, we must sample on the Lie group and respect manifold constraints, while the energy is defined in data space rather than directly on the group. The first issue motivates the use of diffusion sampling methods, which have shown significant promise in sampling efficiently from Boltzmann densities. Figure 3.14 shows an example of the benefits of diffusion for our problem. However, as we will see, existing methods do not address the second issue for general Lie groups.

Diffusion sampling with trivialized target score

We now present Transformation-Inverting Energy Diffusion (TIED), and its associated diffusion sampler on Lie groups, overcoming the aforementioned challenges. To simplify the notations, we fix a data $\tilde{\mathbf{x}}$ and write the transformation inversion posterior defined in Equation (3.10) as $p(g) \equiv p(g \mid \tilde{\mathbf{x}})$. We also consider the energy $E : G \to \mathbb{R}$ associated with the posterior $p(g)$:

$$E(g) \equiv E_{\mathbf{x}}(g^{-1} \cdot \tilde{\mathbf{x}}) - \log |\det J_{g^{-1}}(\tilde{\mathbf{x}})|.$$

To sample from the posterior, we adopt *diffusion sampling*. The idea is to construct a *forward* noising process $p_t(g_t)$ on the group that gradually transforms the posterior $p_0 \equiv p$ into a simple noise distribution p_1. We then run this process *backwards in time* to turn noise samples into samples from the posterior.

For the forward process, we propose to use a direct Lie-group analogue of the Euclidean variance-exploding SDE $\mathrm{d}\mathbf{x}_t = \gamma(t)\,\mathrm{d}\mathbf{w}_t$ in Equation (3.9). The idea is to draw infinitesimal perturbations in the Lie algebra $\mathfrak{g}$ and then transport them to the current point on the group via right multiplication (i.e., trivialization):

$$\mathrm{d}g_t = \mathrm{d}(L_{g_t})_e\left[\gamma(t)\,\mathrm{d}\mathbf{w}_t^{\mathfrak{g}}\right], \quad g_0 \sim p, \tag{3.11}$$

where $\mathrm{d}\mathbf{w}_t^{\mathfrak{g}}$ is Brownian motion in the Lie algebra. Intuitively, we first draw a small random step in $\mathfrak{g}$, then use the tangent map $\mathrm{d}(L_g)_e : \mathfrak{g} \to T_g G$ to translate that step from the identity to the current location g_t on the group. This plays the same role as adding Gaussian noise in the Euclidean setting.

A convenient property of this SDE is that its solution can be written in a simple *multiplicative* form: $g_t = g_0\, w_t$, where $w_t \sim k_t$ is a noise random variable on the group that is independent of g_0. This mirrors the Euclidean relationship $\mathbf{x}_t = \mathbf{x}_0 + \mathbf{w}_t$ for Gaussian noise $\mathbf{w}_t$, except that on a Lie group we combine the signal and noise via group multiplication rather than vector addition. While the distribution k_t does not, in general, admit a closed-form expression because of the group geometry, we can still sample from it efficiently using the exponential map, which is sufficient for our purposes (see Appendix A.6.3).

For diffusion sampling, we reverse this construction: we start from noise samples $w_1 \sim k_1$ and run the SDE backwards in time to obtain samples from $p_0 \equiv p$. This is valid when the diffusion strengths $\gamma(t)$ are large enough that k_1 approximates the marginal p_1 of the forward process. We prove that this time-reversed dynamics is again a trivialized diffusion: it evolves on the group via Lie-algebra noise, but now with an additional drift term given by a *trivialized score* (score evaluated in the Lie algebra).

Proposition 9 (Reverse trivialized SDE)**.** *If each $p_t(g_t)$ is smooth and strictly positive with respect to the Haar measure, then the time-reversal of Equation* (3.11) *is*

$$\mathrm{d}g_t = \mathrm{d}(L_{g_t})_e\big[-\gamma(t)^2\, \nabla_{\mathfrak{g}} \log p_t(g_t)\, \mathrm{d}t + \gamma(t)\, \mathrm{d}\bar{\mathbf{w}}_t^{\mathfrak{g}}\big], \tag{3.12}$$

where $\bar{\mathbf{w}}_t^{\mathfrak{g}}$ is Brownian motion in $\mathfrak{g}$ run backwards in time, and $\nabla_{\mathfrak{g}} \log p_t(g_t)$ is the trivialized score, i.e., the score expressed as an element of the Lie algebra.

Provided we can evaluate or estimate the trivialized score, diffusion sampling becomes straightforward: we discretize the reverse-time SDE and update group elements using the exponential map at each step. We present the resulting sampling scheme in Appendix A.6.3 and focus next on how to estimate the score.

A natural first idea for estimating the score is to learn it with a neural network via score matching [Huang et al., 2022, De Bortoli et al., 2022, Zhu et al., 2025]. However, these methods typically assume access to *clean* samples from $p_0 \equiv p$. In our setting, the posterior p is only available through its associated energy $E \equiv -\log p$ rather than through samples. To address this problem, we introduce a new estimator of the trivialized score that uses the energy instead of requiring clean samples.

We build on the work of Akhound-Sadegh et al. [2024], De Bortoli et al. [2024], who use the *target score identity* to express scores of noisy distributions in terms of gradients of a clean energy. We present a new generalization of this identity to Lie groups that is compatible with trivialization:

Proposition 10 (Trivialized target score identity)**.** *For the forward SDE in Equation* (3.11)*, we have*

$$\nabla_{\mathfrak{g}} \log p_t(g_t) = \int_G \nabla_{\mathfrak{g}} \log p_0(g_0)\, p_{0|t}(g_0 \mid g_t)\, \mathrm{d}\mu(g_0) \tag{3.13}$$

where the argument of $\log p_0$ is interpreted as $g_t b$ for $b \equiv g_t^{-1} g_0$, and $\nabla_{\mathfrak{g}}$ is taken with respect to g_t.

In words, the identity says that the trivialized score at time t can be written as an *average* of the initial score $\nabla_{\mathfrak{g}} \log p_0$ over all possible starting points g_0 that could have led to the current state g_t, weighted by the conditional density $p_{0|t}(g_0 \mid g_t)$. Since $\nabla_{\mathfrak{g}} \log p_0 = -\nabla_{\mathfrak{g}} E$ and we have access to E, this gives us a way to express the desired score entirely in terms of clean energy gradients.

A key feature of our result is that, unlike the Lie-group extension of the target score identity in De Bortoli et al. [2024, Section 3.2], it applies to *general* Lie groups and does not require a bi-invariant

Algorithm 1: Sampling with TIED

Input: Lie group G, energy E, noise schedule γ, step size Δt, Monte Carlo sample size N
Output: Sample $\hat{g}_0 \sim p \propto e^{-E(\cdot)}$

$M \leftarrow 1/\Delta t$
$\hat{g}_M \sim k_1$ using Equation (A.52)
for $m \leftarrow M$ **to** 1 **do**
  $t \leftarrow m\Delta t$
  for $i \leftarrow 1$ **to** N **do**
    $w^{(i)} \sim k_t$ using Equation (A.52)
    Compute $\log \lambda(w^{(i)})$ using Proposition 27
  end
  $f(\cdot) \leftarrow \log \sum_i e^{-E\left((\cdot)\, w^{(i)-1}\right) - \log \lambda(w^{(i)})}$
  $\hat{s}_m \leftarrow \nabla_{\mathfrak{g}} f(\hat{g}_m)$ using Proposition 24
  $\bar{z}_m \sim \mathcal{N}(0, \Delta t I)$
  $\hat{g}_{m-1} \leftarrow \hat{g}_m \exp(\gamma(t)^2 \hat{s}_m \Delta t + \gamma(t)\bar{z}_m)$
end

metric. The reason lies in the use of trivialization. By always expressing gradients as elements of the Lie algebra, we can meaningfully average energy gradients coming from different points on the group without explicitly transporting vectors between different tangent spaces. This removes the need for the right-invariance assumption used in De Bortoli et al. [2024, Appendix A.4] to handle such transports.

Based on the identity, we now derive a form of the trivialized score that is suitable for Monte Carlo estimation. The idea is to rewrite Equation (3.13), currently weighted by the conditional density $p_{0|t}$, into an average weighted by the noise density k_t which can be sampled easily. We achieve this by exploiting the multiplicative form $g_t = g_0 w$ for $w \sim k_t$, and performing a corresponding change of variables on the group. The derivation is in Appendix A.6.2, which is based on a nontrivial extension of the Euclidean case in Akhound-Sadegh et al. [2024, Section 3.1] to Lie groups.

Proposition 11 (Monte Carlo score estimator)**.** *For the forward SDE in Equation* (3.11)*, where p_0 is a Boltzmann density specified by a smooth energy E, we have*

$$\nabla_{\mathfrak{g}} \log p_t(g_t) = \nabla_{\mathfrak{g}} \log \int_G k_t(w) \exp\big(-E(g_t w^{-1}) - \log \lambda(w)\big) \mathrm{d}\mu(w) \tag{3.14}$$

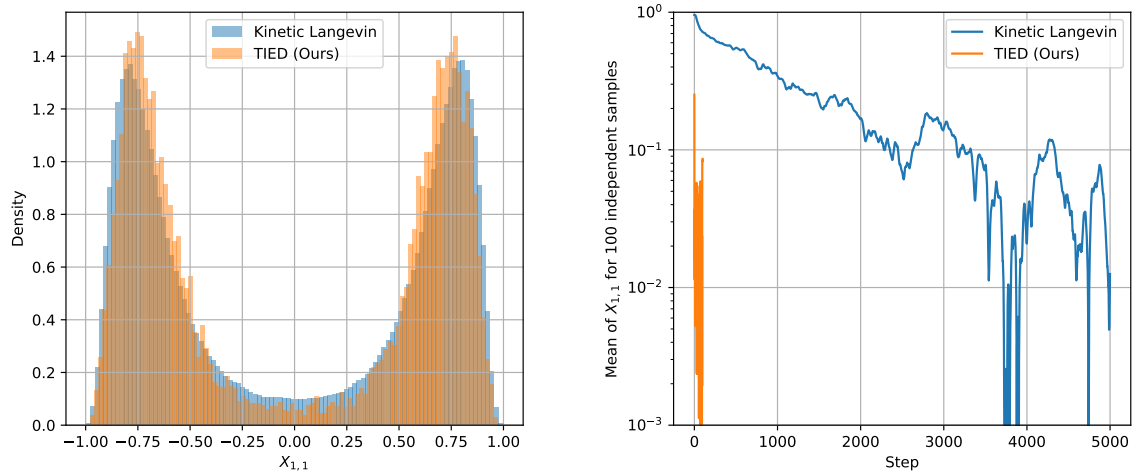
where λ accounts for the change of the Haar measure under inversion, $\mathrm{d}\mu(w^{-1}) = \lambda(w)^{-1}\mathrm{d}\mu(w)$.

This expression is well-suited for practical estimation. We can efficiently draw samples $w \sim k_t$, and the integral in Equation (3.14) can then be approximated via Monte Carlo. We describe this procedure in detail and prove its consistency in Appendix A.6.2.

Practical implementation

We now present a practical implementation of energy-based diffusion sampling with TIED (Algorithm 1). It runs time-discretized reverse SDE on Lie group using energy-based MC estimation of the trivialized score. Here, the number of samples N is a hyperparameter which trades off computational cost and the quality of estimation. In our problem context of inverting transformations, we find that $N \lessapprox 10$ usually yields a satisfactory performance and can be parallelized.

This implementation relies on a nice property of the score estimator that all its required components, including the volume correction λ and the trivialized gradient $\nabla_{\mathfrak{g}}$, can be computed for general Lie groups using standard automatic differentiation tools. We explain these computations in detail in Appendix A.6.3.



(a) Distributions of sampled $\mathbf{X}_{1,1}$.

(b) Mean of $\mathbf{X}_{1,1}$ over timesteps (lower the better).

Figure 3.15: Sampling on SO(10) under energy $E : \mathbf{X} \mapsto -10\mathbf{X}_{1,1}^2$ using a kinetic Langevin sampler [Kong and Tao, 2024] and TIED (Ours). The distribution of $\mathbf{X}_{1,1}$ induced by the energy has two symmetric modes around zero, and thus the mean of $\mathbf{X}_{1,1}$ approaches zero as the sampling converges.

3.3.4 Experiments

We evaluate TIED on (i) a synthetic sampling problem on a high-dimensional Lie group; (ii) two image classification problems using a trained convolutional neural network under unknown affine and homography (perspective) transformations; and (iii) two partial differential equations (PDEs) solving problems using trained neural operators under unknown Lie point symmetry transformations.

Synthetic sampling task on SO(10)

The first setting we consider is a synthetic energy-based sampling problem on a high-dimensional Lie group. We adopt the setup of Kong and Tao [2024] that considers the special orthogonal group SO(10) represented as a set of 10×10 matrices $\mathbf{X}$ with determinant 1. This group is chosen since its large intrinsic dimension $\dim \mathrm{SO}(10) = 45$ offers a challenge for sampling methods (by contrast to e.g. $\dim \mathrm{SO}(3) = 3$). The energy function we consider is $E : \mathbf{X} \mapsto -10\mathbf{X}_{1,1}^2$ where $\mathbf{X}_{1,1}$ is the value of the top-left matrix element. While the definition is simple, this energy induces a multimodal density $p \propto e^{-E(\cdot)}$ on the group, thereby offering a testbed aligning with our motivations in Section 3.3.3.

We compare TIED against the trivialized kinetic Langevin sampler proposed in Kong and Tao [2024], which only uses the energy gradient $\nabla_{\mathfrak{g}} E$. We run the Langevin sampler for 100,000 steps to ensure convergence. For TIED, we run 100 steps of diffusion, with sample size 100 for MC estimation of the score $\nabla_{\mathfrak{g}} \log p_t$. We provide the results in Figure 3.15, focusing on the top-left matrix element $\mathbf{X}_{1,1}$ for visualization. The results show that TIED samples from the correct multimodal distribution, and does so in a much fewer sampling steps, thanks to the diffusion formulation.

Affine and homography invariant image classification

Next, we demonstrate TIED on image classification problems using a trained convolutional neural network under unknown affine and homography (perspective) transformations, closely following the experimental setup of Shumaylov et al. [2024]. As our pretrained network f_θ, we use a ResNet18 [He

Table 3.7: MNIST classification test accuracy and FID. * from the original paper's table.

	MNIST			
test transformations	none			
	Acc ↑	FID ↓		
ResNet18	99.35%	-		
test transformations	$\mathrm{Aff}(2,\mathbb{R})$		$\mathrm{PGL}(3,\mathbb{R})$	
	Acc ↑	FID ↓	Acc ↑	FID ↓
affConv/homConv* [MacDonald et al., 2022]	95.08%	-	95.71%	-
ResNet18	55.48%	-	87.95%	-
Energy: VAE evidence lower bound (+ adv. reg. [Shumaylov et al., 2024])				
ResNet18 + ITS [Schmidt and Stober, 2024]	45.79%	11.00	n/a	n/a
ResNet18 + FoCal [Singhal et al., 2025]	86.35%	3.38	89.69%	2.18
ResNet18 + Kinetic Langevin [Kong and Tao, 2024]	74.55%	7.15	93.72%	0.77
ResNet18 + LieLAC [Shumaylov et al., 2024]	94.36%	0.93	97.42%	**0.58**
ResNet18 + TIED (Ours)	**96.84%**	**0.71**	**97.45%**	**0.58**
Energy: Classifier logit confidence				
ResNet18 + ITS [Schmidt and Stober, 2024]	69.03%	9.88	n/a	n/a
ResNet18 + FoCal [Singhal et al., 2025]	66.73%	10.88	86.24%	3.92
ResNet18 + Kinetic Langevin [Kong and Tao, 2024]	53.83%	15.25	88.26%	2.48
ResNet18 + LieLAC [Shumaylov et al., 2024]	73.58%	6.97	85.91%	2.63
ResNet18 + TIED (Ours)	**85.21%**	**5.07**	**89.81%**	**1.83**

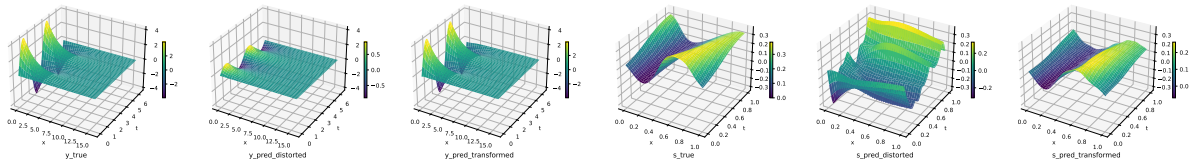
et al., 2016] trained to classify 40×40 padded MNIST images. To test its robustness and generalization, we consider two challenging transformation groups. The first is the group of affine transformations $\mathrm{Aff}(2,\mathbb{R})$, and the second is the group of homography (perspective) transformations, isomorphic to the projective general linear group $\mathrm{PGL}(3,\mathbb{R})$. Despite their widespread use in computer vision, these groups have high mathematical complexities: both are noncompact, non-Abelian, and lack a bi-invariant metric. We construct the respective test sets as 10,000-sized random subsets of affNIST and homNIST test images from MacDonald et al. [2022]. In addition to classification accuracy of f_θ, as a supplementary metric we measure the Fréchet inception distance (FID) [Heusel et al., 2017, Fatir, 2018] between the inverse-transformed test images the training images of f_θ.

The baselines include general optimization and sampling methods on Lie groups, as well as specialized methods for inverting transformations for robust neural perception. For the former, we test trivialized kinetic Langevin [Kong and Tao, 2024] and Lie algebra canonicalization (LieLAC) [Shumaylov et al., 2024], which respectively do trivialized gradient-based sampling and optimization [Lezcano Casado, 2019]. For the latter, we test inverse transformation search (ITS) [Schmidt and Stober, 2024], which performs an iterative search specifically designed for affine transforms, and FoCal [Singhal et al., 2025], which performs Bayesian optimization on transformation spaces (we use the Lie algebra).

Notably, all the baselines use an energy function (sometimes referred to as a data prior) to identify an inverse transformation, making it natural to compare them under the same choices of energy. For image classification, we experiment with two choices of energy functions, coarsely representative of probabilistic and predictive ones. For the probabilistic energy, we use ELBO of a VAE trained on clean MNIST images augmented with adversarial regularization, adopted from Shumaylov et al. [2024]. For the predictive energy, we adopt the confidence-based energy measured by `-logsumexp` of output logits [Grathwohl et al., 2019], similarly to Schmidt and Stober [2024], Singhal et al. [2025]. For this, we use the same ResNet18 classifier f_θ for measuring the classification performance.

Table 3.8: PDE solving relative test L2 error.

	1D Heat Eq.	1D Heat Eq. + Data Aug.	1D Burgers Eq.
test transformations	none	none	none
DeepONet	0.011	0.031	0.017
test transformations	$\mathrm{SL}(2,\mathbb{R}) \ltimes \mathrm{H}(1,\mathbb{R})$	$\mathrm{SL}(2,\mathbb{R}) \ltimes \mathrm{H}(1,\mathbb{R})$	$\mathrm{SL}(2,\mathbb{R}) \ltimes (\mathbb{R}^2, +)$
DeepONet	0.690	0.081	0.867
Energy: Distance to training domain			
DeepONet + FoCal [Singhal et al., 2025]	3.044 ± 2.133	1.618 ± 1.081	0.215 ± 0.031
DeepONet + Kinetic Langevin [Kong and Tao, 2024]	0.587 ± 0.049	0.170 ± 0.151	0.745 ± 0.035
DeepONet + LieLAC [Shumaylov et al., 2024]	0.078 ± 0.014	0.088 ± 0.013	0.190 ± 0.015
DeepONet + TIED (Ours)	**0.042 ± 0.002**	**0.052 ± 0.001**	**0.167 ± 0.028**



(a) 1D heat equation.

(b) 1D Burgers equation.

Figure 3.16: For each PDE, we show an out-of-domain case for DeepONet f_θ. From left: true solution, f_θ prediction, f_θ prediction under test-time equivariance via TIED (Ours).

The results are in Table 3.7. While the trained ResNet18 achieves high accuracy on clean images, affine or homography transformations substantially degrade it. Kinetic Langevin and LieLAC are based on (trivialized) gradients of the energy. Both restore the accuracy reasonably well, but relatively underperforms on affine using classifier energy. This is possibly due to the large magnitudes of transformations (as evidenced by the ResNet18 accuracies) and anomalies in the classifier's energy landscape affecting the gradients. In contrast, ITS and FoCal, which are iterative search methods, attain decent results on affine with classifier energy while often outperformed by gradient-based methods on others. We conjecture that their nonlocal iterative exploration helps handle the anomalies in the energy landscape. Finally, our method, TIED, combines precise local optimization via gradients with exploration by diffusion at high noise levels. It outperforms all baselines in all settings, in particular significantly improving the accuracy under affine transforms and classifier energy (55.48% $\rightarrow$ 82.51%). With a proper choice of energy, TIED also outperforms specialized equivariant architectures affConv/homConv [MacDonald et al., 2022].

Point symmetry equivariant PDE solving

Next, we test TIED on a more challenging problem of solving PDEs with a trained neural operator under unknown point symmetry transforms. Here, the model f_θ performs function-valued regression. At a high-level, the input to f_θ is a function evaluation $\mathbf{u}_0$ at a set of space and time points $(\mathbf{x}_0, t_0)$ specifying an initial condition. The task is evaluating the function at another set of points $(\mathbf{x}_f, t_f)$, after $\mathbf{u}_0(\mathbf{x}_0, t_0)$ has evolved under the PDE of interest. More details can be found in Li et al. [2020], Lu et al. [2021].

Many PDEs possess a symmetry G that transform a solution $\mathbf{u}(\mathbf{x}, t)$, to create other valid solutions [Olver, 1993, Brandstetter et al., 2022, Akhound-Sadegh et al., 2023]. Since a neural operator f_θ is trained for a specific range of initial conditions $\mathbf{u}_0(\mathbf{x}_0, t_0)$ and prediction points $(\mathbf{x}_f, t_f)$, one expects it to fail outside the training domain. Akin to our problem setup, this can be overcome by finding a symmetry transformation g which acts on input as $g^{-1} \cdot (\mathbf{u}_0(\mathbf{x}_0, t_0), \mathbf{x}_f, t_f)$ to push it into the training domain of f_θ.

Table 3.9: MNIST classification test wall-clock runtime measured on a single NVIDIA A6000 GPU.

	MNIST	
test transformations	$\mathrm{Aff}(2,\mathbb{R})$	$\mathrm{PGL}(3,\mathbb{R})$
affConv/homConv* [MacDonald et al., 2022] (training time)	36 hours	36 hours
Energy: VAE evidence lower bound (+ adv. reg. [Shumaylov et al., 2024])		
ResNet18 + ITS [Schmidt and Stober, 2024]	13 mins	n/a
ResNet18 + FoCal [Singhal et al., 2025]	8 hours	12 hours
ResNet18 + Kinetic Langevin [Kong and Tao, 2024]	2 hours	2 hours
ResNet18 + LieLAC [Shumaylov et al., 2024]	70 mins	70 mins
ResNet18 + TIED (Ours)	53 mins	53 mins
Energy: Classifier logit confidence		
ResNet18 + ITS [Schmidt and Stober, 2024]	13 mins	n/a
ResNet18 + FoCal [Singhal et al., 2025]	2 hours	2 hours
ResNet18 + Kinetic Langevin [Kong and Tao, 2024]	1 hour	73 mins
ResNet18 + LieLAC [Shumaylov et al., 2024]	30 mins	30 mins
ResNet18 + TIED (Ours)	30 mins	30 mins

Then, f_θ can make out-of-domain prediction as $g \cdot f_\theta(g^{-1} \cdot (\mathbf{u}_0(\mathbf{x}_0, t_0), \mathbf{x}_f, t_f))$. The main challenge is the complexity of Lie point symmetries: in many cases, they are noncompact and non-Abelian. This makes TIED an appealing candidate as it handles these cases.

We closely follow the setup of Shumaylov et al. [2024], using DeepONet neural operators [Lu et al., 2021] as the pretrained network f_θ. To test the robustness and generalization, we use two PDEs with challenging symmetry groups. The first is 1D heat equation $u_t - \nu u_{xx} = 0$ with the symmetry group $\mathrm{SL}(2,\mathbb{R}) \ltimes \mathrm{H}(1,\mathbb{R})$, the semidirect product of special linear group and rank-one polarized Heisenberg group. The second is 1D Burgers' equation $u_t + uu_x - \nu u_x x = 0$ with the symmetry group $\mathrm{SL}(2,\mathbb{R}) \ltimes (\mathbb{R}^2, +)$. The test sets are constructed by applying these transformations to send $\mathbf{u}_0$ out of the training domain of f_θ. For the choice of energy, we follow Shumaylov et al. [2024] and use a distance measure between the input $(\mathbf{u}_0(\mathbf{x}_0, t_0), \mathbf{x}_f, t_f)$ and the training domain of f_θ.

The results are in Table 3.8 and Figure 3.16. While all DeepONets excel in their training domain, their performances degrade when presented with out-of-domain transformation, with data augmentation reducing the degradation but not perfectly. While FoCal performs poorly for heat equations, we find that it has a strong performance on Burgers equation. We conjecture that this is partially due to the Euclidean component $(\mathbb{R}^2, +)$ in the semidirect product, which may have created a more amenable environment for Bayesian optimization. While both based on gradients, LieLAC outperforms kinetic Langevin and attains good performances. This could be due to the Brownian motion term slowing down convergence. Our method TIED can leverage smoothened energy landscape at high noise levels effectively to speed up convergence, and achieves the best performance in all cases.

Runtime measurements

In Tables 3.9 and 3.10, we provide the wall-clock runtime of MNIST classification experiment and PDE solving experiment, respectively. TIED is often faster than baseline methods, supporting the scalability and efficiency of the proposed diffusion sampling.

Table 3.10: PDE solving wall-clock runtime measured on a single NVIDIA 6000 GPU.

	1D Heat Eq.	1D Heat Eq. + Data Aug.	1D Burgers Eq.
Test transformations	$\mathrm{SL}(2,\mathbb{R}) \ltimes \mathrm{H}(1,\mathbb{R})$	$\mathrm{SL}(2,\mathbb{R}) \ltimes \mathrm{H}(1,\mathbb{R})$	$\mathrm{SL}(2,\mathbb{R}) \ltimes (\mathbb{R}^2,+)$
Energy: Distance to training domain			
DeepONet + FoCal [Singhal et al., 2025]	17 mins	17 mins	7.5 mins
DeepONet + Kinetic Langevin [Kong and Tao, 2024]	2 hours	2 hours	88 mins
DeepONet + LieLAC [Shumaylov et al., 2024]	38 secs	38 secs	47 secs
DeepONet + TIED (Ours)	43 secs	43 secs	47 secs

3.3.5 Discussion

We studied the problem of inverting unknown data transformations from general Lie groups in a setting where we have access to a *data-space prior* specified by an energy function. Given such an energy in data space and a fixed pretrained model, our goal is to *transform* a test-time input so that it becomes well aligned with the model's training distribution and thereby achieves strong performance. This setting is increasingly relevant in the era of large foundation models, where robustness to out-of-distribution transformations remains a practical concern.

To address this, we proposed TIED, a purely test-time method that requires no training. TIED operates by reverting a diffusion process over transformations in the Lie algebra. Our approach naturally handles the rugged and multimodal energy landscapes that arise in practice, applies to a broad family of (possibly noncompact, non-Abelian) groups and nonlinear actions, and remains computationally attractive by working in the low-dimensional space of transformations rather than the high-dimensional data space.

On challenging problems, such as affine and homography invariant computer vision and neural PDE solving under unseen initial conditions, we showed that sampling from the transformation posterior and canonicalizing the inputs at test time can reliably bring data back in-distribution and significantly improve the robustness of pretrained models without any retraining.

Multiple avenues remain for future work. One limitation of our method is that the score estimator requires evaluating the energy and its gradients over multiple MC samples. We could also consider accelerating the sampling, for example, using consistency distillation methods [Song et al., 2023]. Applications to other inverse problems, such as image registration, would also prove interesting. Another potential extension is to problems where the group action is not known a priori and has to be learned [Koyama et al., 2023, Yang et al., 2023, Mitchel et al., 2024]. Finally, a potentially exciting application of our method is to transformations that do not form a group. In practice, many relevant applications require dealing with transformations that are not invertible or inherently noisy.

Chapter 4. Random Walk Neural Networks

The previous chapters addressed architecture-agnostic invariance through tokenization schemes and stochastic group transformations of the input. In this chapter, we pursue a complementary approach of restructuring the input representation itself. We investigate random walk neural networks, which convert graphs to sequences via stochastic traversals. This transformation allows sequence models, including pre-trained language models, to be applied directly to graph learning tasks with provable invariance guarantees. Despite their conceptual and empirical promise, random walk neural networks have previously lacked a unified theoretical foundation. We address this gap by formalizing these models through probabilistic invariance and Markov chain theory.

This chapter is adapted from the following papers:

- "Revisiting Random Walks for Learning on Graphs", which originally appeared at ICLR 2025 and is joint work with Olga Zaghen, Ayhan Suleymanzade, Youngmin Ryou, and Seunghoon Hong,
- "Flock: A Knowledge Graph Foundation Model via Learning on Random Walks", which is currently under review and is joint work with Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and İsmail İlkan Ceylan.

The first work provides a theoretical and empirical foundation for random walk neural networks on general graphs, and the second applies this framework to build a foundation model for knowledge graphs. In Section 4.1, we analyze random walk neural networks by decomposing them into three components: the walk algorithm, the recording protocol, and the neural processor. We show that these models can achieve universal approximation in probability while being more robust to over-smoothing and over-squashing than message-passing networks, and demonstrate that pre-trained language models can be applied in an invariant manner to graph reasoning, classification on scientific and social networks, and protein property prediction. In Section 4.2, we introduce Flock, a knowledge graph foundation model based on random walk neural networks. By leveraging probabilistic rather than deterministic equivariance, Flock overcomes the limited expressivity of previous methods, which fail to distinguish structurally similar but semantically distinct relations, achieving state-of-the-art performance in zero-shot link prediction on knowledge graphs with entirely unseen entities and relations.

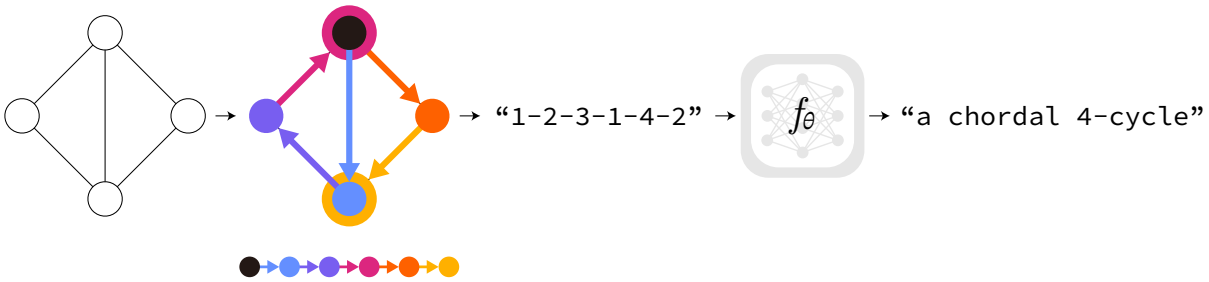


Figure 4.1: An RWNN that reads text record using a language model.

4.1 Random walk neural networks

Message passing neural networks (MPNNs) are a popular class of neural networks on graphs where each vertex keeps a feature vector and updates it by propagating messages over neighbors [Battaglia et al., 2018, Gilmer et al., 2017]. MPNNs have achieved success, with one reason being their respect for the natural symmetries of graph problems (i.e., invariance to graph isomorphism) [Chen et al., 2019]. On the other hand, MPNNs in their basic form can be viewed as implementing color updates of the Weisfeiler-Lehman (1-WL) graph isomorphism test, and thus their expressive power is not stronger [Xu et al., 2019]. Also, their inner workings are often tied to the topology of the input graph, which is related to over-smoothing, over-squashing, and under-reaching of features under mixing [Giraldo et al., 2023].

In this work, we revisit an alternative direction for learning on graphs, where a random walk on a graph produces a machine-readable record, and this record is processed by a deep neural network that directly makes graph-level or vertex-level predictions. We refer to these stochastic machines as **random walk neural networks (RWNNs)**. Pioneering works were done in this direction, often motivated by the compatibility of random walks with sequence learning methods. Examples include DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016], which use skip-gram on walks, AWE [Ivanov and Burnaev, 2018], which uses embeddings of anonymized walks [Micali and Zhu, 2016], and CRaWl [Tönshoff et al., 2023], which uses 1D CNNs on sliding window descriptions of walks. These methods were proven powerful in various contexts, such as graph isomorphism learning that requires expressive powers surpassing 1-WL test [Tönshoff et al., 2023, Martinkus et al., 2023].

Despite the potential, unlike MPNNs, we have not yet reached a good principled understanding of RWNNs. First, from the viewpoint of geometric deep learning, it is unclear how we can systematically incorporate the symmetries of graph problems into RWNNs, as their neural networks are not necessarily isomorphism invariant. Second, the upper bound of their expressive power, along with the requirements on each component to reach it, is not clearly known. Third, whether and how the over-smoothing and over-squashing problems may occur in RWNNs is not well understood. Addressing these questions would improve our understanding of the existing methods and potentially allow us to design enhanced RWNNs by short-circuiting the search in their large design space.

Contributions We aim to establish a principled understanding of RWNNs with a focus on the previously stated questions. Our starting point is a new formalization of RWNNs which decouples random walks, their records, and neural networks that process them. This abstracts a wide range of design choices including those of prior methods, as in Table 4.1. Our key idea for the analysis is then to understand RWNNs through a combination of two recent perspectives in geometric deep learning: probabilistic notions of invariance and expressive power [Bloem-Reddy and Teh, 2020, Abboud et al., 2021], and

invariant projection of non-invariant neural networks [Puny et al., 2022, Dym et al., 2024].

This idea allows us to impose probabilistic invariance on RWNNs even if their neural networks lack symmetry, by instead requiring (probabilistic) invariance conditions on random walks and their records. This provides a justification for the anonymized recording of walks [Micali and Zhu, 2016] and our novel extension of it using named neighbors. As long as probabilistic invariance holds, this also enables recording walks in plain text and adopting a language model to process them (Figure 4.1).

We then upper-bound the expressive power of RWNNs as universal approximation in probability which surpasses the WL hierarchy of MPNNs, under the condition that random walk records the graph of interest with a high probability. This establishes a useful link to cover times of Markov chains, providing guidance on designing random walks to minimize or bound the cover times. From this we motivate a novel adaptation of minimum degree local rule (MDLR) walks [David and Feige, 2018] for RWNNs, and introduce restarts when working on a large and possibly infinite graph. We also provide a justification for the widespread use of non-backtracking [Tönshoff et al., 2023].

Continuing the link to Markov chain theory, we further analyze RWNNs and establish a parallelism to a linearized model of MPNNs. From this, we show that over-smoothing in MPNNs is inherently avoided in RWNNs, while over-squashing manifests as probabilistic under-reaching. This eliminates the typical trade-off between over-smoothing and over-squashing in MPNNs [Giraldo et al., 2023, Nguyen et al., 2023a], and allows an RWNN to focus on overcoming under-reaching by scaling the walk length or using rapidly-mixing random walks such as non-backtracking walks.

We empirically demonstrate RWNNs on several graph problems. On synthetic setups, optionally with a small 1-layer transformer, we verify our claims on cover times, over-smoothing, and over-squashing. Then, we demonstrate adapting a language model [He et al., 2021b, DeBERTa] to solve the challenging task of isomorphism learning of strongly regular graphs with perfect accuracy whereas the 3-WL test fails, improving over the previously known best result of RWNNs. We further suggest that our approach can turn transductive classification problem on a large graph into an in-context learning problem by recording labeled vertices during random walks. To demonstrate this, we apply Llama 3 [Dubey et al., 2024] model on transductive classification on arXiv citation network with 170k vertices, and show that it can outperform a range of MPNNs as well as zero- and few-shot baselines. Our experiments show the utility of our approach in the analysis and development of RWNNs.

4.1.1 Method

We start our discussion by formalizing random walk neural networks (RWNNs). We define an RWNN as a randomized function $X_\theta(\cdot)$ that takes a graph G as input and outputs a random variable $X_\theta(G)$ on the output space $\mathbb{R}^d$.[1] In the case of vertex-level tasks, it is additionally queried with an input vertex v which gives a random variable $X_\theta(G, v)$. An RWNN consists of the following components:

1. **Random walk algorithm** that produces l steps of vertex transitions $v_0 \to \cdots \to v_l$ on the input graph G. If an input vertex v is given, we fix the starting vertex by $v_0 = v$.

2. **Recording function** $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ that produces a machine-readable record $\mathbf{z}$ of the random walk. It may access the graph G to record auxiliary information such as attributes.

3. **Reader neural network** $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}}$ that processes record $\mathbf{z}$ and outputs a prediction $\hat{\mathbf{y}}$ in $\mathbb{R}^d$. It is the only trainable component and is not restricted to specific architectures.

[1] While any output type such as text is possible (Figure 4.1), we explain with vector output for simplicity.

Table 4.1: An overview of prior methods in the context of RWNNs and new components originating from our analysis. NB is non-backtracking; MDLR is minimum degree local rule (Section 4.1.1).

Method	Random walk	Recording function q	Reader NN f_θ
DeepWalk [Perozzi et al., 2014]	Uniform	Identity	Skip-gram
node2vec [Grover and Leskovec, 2016]	Second-order pq-walk	Identity	Skip-gram
AWE [Ivanov and Burnaev, 2018]	Uniform	Anonymization	Embedding table
CRaWl [Tönshoff et al., 2023]	Uniform + NB	Sliding window	1D CNN
RW-AgentNet [Martinkus et al., 2023]	Uniform	Neighborhoods	RNN
WalkLM [Tan et al., 2023a]	Uniform	Text attributes	Language model
Ours (§4.1.1, §4.1.2)	MDLR (+ NB) (+ restarts)	Anonymization (+ named neighbors)	Any universal

For graph-level prediction, we sample $\hat{\mathbf{y}} \sim X_\theta(G)$ by running a random walk on G and processing its record using the reader NN f_θ. For vertex-level prediction $\hat{\mathbf{y}} \sim X_\theta(G, v)$, we query the input vertex v by simply starting the walk from it. In practice, we ensemble several sampled predictions e.g. by averaging, which can be understood as Monte Carlo estimation of the mean predictor $(\cdot) \mapsto \mathbb{E}[X_\theta(\cdot)]$.

We now analyze each component for graph-level tasks on finite graphs, and then vertex-level tasks on possibly infinite graphs. The latter simulates the problem of scaling to large graphs such as in transductive classification. As our definition captures many known designs (Table 4.1), we discuss them as well. We leave the pseudocode in Appendix A.7.2 and notations and proofs in Appendix A.7.4.

Graph-level tasks

Let $\mathbb{G}$ be the class of undirected, connected, and simple graphs[2]. Let $n \geq 1$ and $\mathbb{G}_n$ be the collection of graphs in $\mathbb{G}$ with at most n vertices. Our goal is to model a graph-level function $\phi : \mathbb{G}_n \to \mathbb{R}^d$ using an RWNN $X_\theta(\cdot)$. Since ϕ is a graph function, it is reasonable to assume isomorphism invariance:

$$\phi(G) = \phi(H), \qquad \forall G \simeq H.$$

Incorporating the invariance structure to our model class $X_\theta(\cdot)$ would offer generalization benefit. As $X_\theta(\cdot)$ is randomized, we accept the probabilistic notion of invariance [Bloem-Reddy and Teh, 2020]:

$$X_\theta(G) \overset{d}{=} X_\theta(H), \qquad \forall G \simeq H. \tag{4.1}$$

An intuitive justification is that, if $X_\theta(\cdot)$ is invariant in probability, its mean predictor $(\cdot) \mapsto \mathbb{E}[X_\theta(\cdot)]$ would be an invariant function (we leave further discussion in Section 4.1.3). We now claim that we can achieve probabilistic invariance of $X_\theta(\cdot)$ by properly choosing the random walk algorithm and recording function while not imposing any constraint on the reader NN f_θ.

Proposition 12. *$X_\theta(\cdot)$ is invariant in probability, if its random walk algorithm is invariant in probability and its recording function is invariant.*

In other words, even if f_θ lacks symmetry, invariant random walk and recording function provably converts it into an invariant random variable $X_\theta(\cdot)$. This is an extension of invariant projection operators on functions [Puny et al., 2022, Dym et al., 2024] to a more general and probabilistic setup.

[2]We assume this for simplicity but extending to directed or attributed graphs is possible.

Random walk algorithm A random walk algorithm is invariant in probability if it satisfies:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \qquad \forall G \overset{\pi}{\simeq} H, \tag{4.2}$$

where $v_{[\cdot]}$ is a random walk on G, $u_{[\cdot]}$ is a random walk on H, and $\pi : V(G) \to V(H)$ specifies the isomorphism from G to H. It turns out that many random walk algorithms in literature are already invariant. To see this, let us write the probability of walking from a vertex u to its neighbor $x \in N(u)$:

$$\text{Prob}[v_t = x | v_{t-1} = u] \coloneqq \frac{c_G(u,x)}{\sum_{y \in N(u)} c_G(u,y)}, \tag{4.3}$$

where the function $c_G : E(G) \to \mathbb{R}_+$ assigns positive weights (conductances) to edges. If we set $c_G(\cdot) = 1$, we recover uniform random walk used in DeepWalk [Perozzi et al., 2014]. We show:

Proposition 13. *The random walk in Equation* (4.3) *is invariant in probability if its conductance* $c_{[\cdot]}(\cdot)$ *is invariant:*

$$c_G(u,v) = c_H(\pi(u), \pi(v)), \qquad \forall G \overset{\pi}{\simeq} H. \tag{4.4}$$

It includes constant conductance, and any choice that only uses degrees of endpoints $\deg(u)$, $\deg(v)$.

We favor using degrees since it is cheap while potentially improving the behaviors of walks. Among many possible choices, we find that the following conductance function called the minimum degree local rule (MDLR) [Abdullah et al., 2015, David and Feige, 2018] is particularly useful:

$$c_G(u,v) \coloneqq \frac{1}{\min[\deg(u), \deg(v)]}. \tag{4.5}$$

MDLR is special as its vertex cover time, i.e., the expected time of visiting all n vertices of a graph, is $O(n^2)$, optimal among first-order random walks (Equation (4.3)) that use degrees of endpoints.

A common practice is to add non-backtracking property that enforces $v_{t+1} \neq v_{t-1}$ [Tönshoff et al., 2023], and we also find this beneficial. In Appendix A.7.1 we extend Equation (4.2) and Proposition 13 to second-order walks that include non-backtracking and node2vec walks [Grover and Leskovec, 2016].

Recording function A recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ takes a random walk and produces a machine-readable record $\mathbf{z}$. We let $q(\cdot, G)$ have access to the graph G the walk is taking place. This allows recording auxiliary information such as vertex or edge attributes. A recording function is invariant if it satisfies the following for any given random walk $v_{[\cdot]}$ on G:

$$q(v_0 \to \cdots \to v_l, G) = q(\pi(v_0) \to \cdots \to \pi(v_l), H), \qquad \forall G \overset{\pi}{\simeq} H. \tag{4.6}$$

Invariance requires that $q(\cdot, G)$ produces the same record $\mathbf{z}$ regardless of re-indexing of vertices of G into H. For this, we have to be careful in how we represent each vertex in a walk $v_0 \to \cdots \to v_l$ as a machine-readable value, and which auxiliary information we record from G. For example, identity recording used in DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016] are not invariant as the result is sensitive to vertex re-indexing. We highlight two choices which are invariant:

- **Anonymization.** We name each vertex in a walk with a unique integer, starting from $v_0 \mapsto 1$ and incrementing it based on their order of discovery. For instance, a random walk $a \to b \to c \to a$

translates to a sequence $1 \to 2 \to 3 \to 1$.

- **Anonymization + named neighbors.** While applying anonymization for each vertex v in a walk, we record its neighbors $u \in N(v)$ if they are already named but the edge $v \to u$ has not been recorded yet. For instance, a walk $a \to b \to c \to d$ on a fully-connected graph translates to a sequence $1 \to 2 \to 3\langle 1\rangle \to 4\langle 1, 2\rangle$, where $\langle\cdot\rangle$ represents the named neighbors.

The pseudocode of both algorithms can be found in Appendix A.7.2. We now show the following:

Proposition 14. *A recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ that uses anonymization, optionally with named neighbors, is invariant.*

While anonymization was originally motivated by privacy concerns in Micali and Zhu [2016], Proposition 14 offers a new justification based on invariance. Our design of recording named neighbors is novel, and is inspired by sublinear algorithms that probe previously discovered neighbors in a walk [Dasgupta et al., 2014]. In our context, it is useful since whenever a walk visits a set of vertices $S \subseteq V(G)$ it automatically records the entire induced subgraph $G[S]$. As a result, to record all edges of a graph, a walk only has to visit all vertices. While traversing all edges, i.e. edge cover time, is $O(n^3)$ [Zuckerman, 1991] in general, it is possible to choose a walk algorithm that takes only $O(n^2)$ time to visit all n vertices. MDLR in Equation (4.5) exactly achieves this.

Reader neural network A reader neural network $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}}$ processes the record $\mathbf{z}$ of the random walk and outputs a prediction $\hat{\mathbf{y}}$ in $\mathbb{R}^d$. As in Proposition 12, there is no invariance constraint imposed on f_θ, and any neural network that accepts the recorded walks and has a sufficient expressive power can be used (we will make this precise). This is in contrast to MPNNs where invariance is hard-coded in feature mixing operations. Also, our record $\mathbf{z}$ can take any format, such as a matrix, byte sequence, or plain text, as long as f_θ accepts it. Thus, it is possible (while not required) to choose the record to be plain text, such as `"1-2-3-1"`, and choose f_θ to be a pre-trained language model. This offers expressive power [Yun et al., 2020] and has a potential benefit of knowledge transfer from the language domain [Lu et al., 2022, Rothermel et al., 2021].

Vertex-level tasks

We now consider vertex-level tasks. In case of finite (small) graphs, we may simply frame a vertex-level task $(G, v) \mapsto \mathbf{y}$ as a graph-level task $G' \mapsto \mathbf{y}$ where G' is G with its vertex v marked. Then, we can solve $G' \mapsto \mathbf{y}$ by querying an RWNN $X_\theta(G, v)$ to start its walk at $v_0 = v$.

A more interesting case is when G is an infinite graph that is only locally finite, i.e. has finite degrees. This simulates problems on a large graph such as transductive classification. In this case, we may assume that our target function depends on finite local structures [Tahmasebi et al., 2023].

Let $r \geq 1$, and $\mathbb{B}_r \coloneqq \{B_r(v)\}$ be the collection of local balls in G of radius r centered at $v \in V(G)$. We would like to model a vertex-level function $\phi : \mathbb{B}_r \to \mathbb{R}^d$ on G using an RWNN $X_\theta(\cdot)$ by querying the vertex of interest, $X_\theta(G, v)$. We assume that ϕ is isomorphism invariant:

$$\phi(B_r(v)) = \phi(B_r(u)), \qquad \forall B_r(v) \simeq B_r(u).$$

The probabilistic invariance of $X_\theta(\cdot)$ is defined as follows:

$$X_\theta(G, v) \overset{d}{=} X_\theta(G, u), \qquad \forall B_r(v) \simeq B_r(u).$$

We can achieve probabilistic invariance by choosing the walk algorithm and recording function similar to the graph-level case[3]. As a modification, we query $X_\theta(G, v)$ with the starting vertex $v_0 = v$ of the random walk. Anonymization informs v to the reader NN by always naming it as 1.

Then, we make a key choice of localizing random walks with restarts. That is, we reset a walk to its starting vertex v_0 either with a probability $\alpha \in (0, 1)$ at each step or periodically every k steps. Restarting walks tend to stay more around starting vertex v_0, and were used to implement locality bias in personalized PageRank algorithm for search [Page et al., 1999]. This localizing effect is crucial in our context since a walk can drift away from $B_r(v_0)$ before recording all necessary information, which may take an infinite time to return as G is infinite [Janson and Peres, 2012]. Restarts make the return to v_0 mandatory, ensuring that $B_r(v_0)$ can be recorded in a finite expected time. We show:

Theorem 14. *For a uniform random walk on an infinite graph G starting at v, the vertex and edge cover times of the finite local ball $B_r(v)$ are not always finitely bounded.*

Theorem 15. *In Theorem 14, if the random walk restarts at v with any nonzero probability α or any period $k \geq r + 1$, the vertex and edge cover times of $B_r(v)$ are always finite.*

4.1.2 Theoretical analysis

In Section 4.1.1, we have described the design of RWNNs, primarily relying on the principle of (probabilistic) invariance. In this section, we provide in-depth analysis on their expressive power and relations to the issues in MPNNs such as over-smoothing, over-squashing, and under-reaching.

Expressive power

Intuitively, if the records of random walks contain enough information such that the structures of interest, e.g., graph G or local ball $B_r(v)$, can be fully recovered, a powerful reader NN such as an MLP [Hornik et al., 1989, Cybenko, 1989] or a transformer [Yun et al., 2020] on these records would be able to approximate any function of interest. Our analysis formalizes this intuition.

We first consider using an RWNN $X_\theta(\cdot)$ to universally approximate graph-level functions $\phi(\cdot)$ in probability, as defined in Abboud et al. [2021].

Definition 6. *$X_\theta(\cdot)$ is a universal approximator of graph-level functions in probability if, for all invariant functions $\phi : \mathbb{G}_n \to \mathbb{R}$ for a given $n \geq 1$, and $\forall \epsilon, \delta > 0$, there exist choices of length l of the random walk and network parameters θ such that the following holds:*

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] > 1 - \delta, \qquad \forall G \in \mathbb{G}_n.$$

We show that, if the random walk is long enough and the reader NN f_θ is universal, an RWNN $X_\theta(\cdot)$ is capable of graph-level universal approximation. The length of the walk l controls the confidence $> 1 - \delta$, with edge cover time $C_E(G)$ or vertex cover time $C_V(G)$ playing a central role.

Theorem 16. *An RWNN $X_\theta(\cdot)$ with a sufficiently powerful f_θ is a universal approximator of graph-level functions in probability (Definition 6) if it satisfies either of the below:*

- *It uses anonymization to record random walks of lengths $l > C_E(G)/\delta$.*
- *It uses anonymization + named neighbors to record walks of lengths $l > C_V(G)/\delta$.*

[3]This is assuming that the random walks are localized in $B_r(v)$ and $B_r(u)$, e.g. with restarts.

While both cover times are $O(n^3)$ for uniform random walks [Aleliunas et al., 1979, Zuckerman, 1991], we can use MDLR in Equation (4.5) to achieve an $O(n^2)$ vertex cover time, in conjunction with named neighbors recording. While universality can be in principle achieved with uniform random walks, our design reduces the worst-case length l required for the desired reliability $> 1 - \delta$.

We now show an analogous result for the universal approximation of vertex-level functions.

Definition 7. *$X_\theta(\cdot)$ is a universal approximator of vertex-level functions in probability if, for all invariant functions $\phi : \mathbb{B}_r \to \mathbb{R}$ for a given $r \geq 1$, and $\forall \epsilon, \delta > 0$, there exist choices of length l and restart probability α or period k of the random walk and network parameters θ such that:*

$$\text{Prob}[|\phi(B_r(v)) - X_\theta(G, v)| < \epsilon] > 1 - \delta, \qquad \forall B_r(v) \in \mathbb{B}_r.$$

Accordingly, using local vertex cover time $C_V(B_r(v))$ and edge cover time $C_E(B_r(v))$, we show:

Theorem 17. *An RWNN $X_\theta(\cdot)$ with a sufficiently powerful f_θ and any nonzero restart probability α or restart period $k \geq r + 1$ is a universal approximator of vertex-level functions in probability (Definition 7) if it satisfies either of the below for all $B_r(v) \in \mathbb{B}_r$:*

- *It uses anonymization to record random walks of lengths $l > C_E(B_r(v))/\delta$.*
- *It uses anonymization + named neighbors to record walks of lengths $l > C_V(B_r(v))/\delta$.*

Since restarts are required to finitely bound the local cover times on an infinite graph (Theorem 15), non-restarting walks cannot support vertex-level universality in general, and our design is obligatory.

Over-smoothing, over-squashing, and under-reaching

Often, in MPNNs, each layer operates by passing features over edges and mixing them using weights deduced from e.g., adjacency matrix. This ties them to the topology of the input graph, and a range of prior work has shown how this relates to the well-known issues of over-smoothing, over-squashing, and under-reaching. We connect RWNNs to these results to verify if similar issues may take place.

Let G be a connected non-bipartite graph with row-normalized adjacency matrix P. We consider a linearized MPNN, where the vertex features $\mathbf{h}^{(0)}$ are initialized as some probability vector $\mathbf{x}$, and updated by $\mathbf{h}^{(t+1)} = \mathbf{h}^{(t)} P$. This simplification is often useful in understanding the aforementioned issues [Giraldo et al., 2023, Zhao and Akoglu, 2019]. Specifically, in this model:

- Over-smoothing happens as the features exponentially converge to a stationary vector $\mathbf{h}^{(l)} \to \boldsymbol{\pi}$ as $l \to \infty$, smoothing out the input $\mathbf{x}$ [Giraldo et al., 2023].
- Over-squashing and under-reaching occur when a feature $\mathbf{h}_u^{(l)}$ becomes insensitive to distant input $\mathbf{x}_v$. While under-reaching refers to insufficient depth $l < \text{diam}(G)$ [Barceló et al., 2020], over-squashing refers to features getting overly compressed at bottlenecks of G, even with sufficient depth l. The latter is described by the Jacobian $|\partial \mathbf{h}_u^{(l)} / \partial \mathbf{x}_v| \leq [\sum_{t=0}^{l} P^t]_{uv}$,[4] as the bound often decays exponentially with l [Topping et al., 2022, Black et al., 2023].

What do these results tell us about RWNNs? We can see that, while P drives feature mixing in the message passing schema, it can be also interpreted as the transition probability matrix of uniform random walk where P_{uv} is the probability of walking from u to v. This parallelism motivates us to design an analogous, simplified RWNN and study its behavior.

[4] This bound is obtained by applying Lemma 3.2 of Black et al. [2023] to our linearized MPNN.

We consider a simple RWNN that runs a uniform random walk $v_0 \to \cdots \to v_l$, reads the record $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by averaging, and outputs it as $\mathbf{h}^{(l)}$. Like linear MPNN, the model involves l steps of time evolution through P. However, while MPNN uses P to process features, this model uses P only to obtain a record of input, with feature processing decoupled. We show that in this model, over-smoothing does not occur as in simple MPNNs[5]:

Theorem 18. *The simple RWNN outputs* $\mathbf{h}^{(l)} \to \mathbf{x}^\top \boldsymbol{\pi}$ *as* $l \to \infty$.

Even if the time evolution through P happens fast (i.e. P is rapidly mixing) or with many steps l, the model is resistant to over-smoothing as the input $\mathbf{x}$ always affects the output. In fact, if P is rapidly mixing, we may expect improved behaviors based on Theorems 16 and 17 as the cover times could reduce.

On the other hand, we show that over-squashing manifests as probabilistic under-reaching:

Theorem 19. *Let* $\mathbf{h}_u^{(l)}$ *be output of the simple RWNN queried with* u. *Then:*

$$\mathbb{E}\left[\left|\frac{\partial \mathbf{h}_u^{(l)}}{\partial \mathbf{x}_v}\right|\right] = \frac{1}{l+1}\left[\sum_{t=0}^{l} P^t\right]_{uv} \to \boldsymbol{\pi}_v \qquad as \qquad l \to \infty. \tag{4.7}$$

The equation shows that the feature Jacobians are bounded by the sum of powers of P, same as in simple MPNN. Both models are subject to over-squashing phenomenon that is similarly formalized, but manifests through different mechanisms. While in message passing the term is related to over-compression of features at bottlenecks [Topping et al., 2022], in RWNNs it is related to exponentially decaying probability of reaching a distant vertex v, i.e., probabilistic under-reaching.

In many MPNNs, it is understood that the topology of the input graph inevitably induces a trade-off between over-smoothing and over-squashing [Nguyen et al., 2023a, Giraldo et al., 2023]. Our results suggest that RWNNs avoid the trade-off, and we can focus on overcoming under-reaching e.g., with long or rapidly-mixing walks, while not worrying much about over-smoothing. Design choices such as MDLR (Equation (4.5)) and non-backtracking [Alon et al., 2007] can be understood as achieving this.

4.1.3 Related work

Random walks for learning on graphs In graph learning, random walks have received interest due to their compatibility with sequence learning methods (Table 4.1). DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016] used skip-gram models on walks. CRaWl [Tönshoff et al., 2023] used 1D CNN on sliding window-based walk records, with expressive power bounded by the window size. AgentNet [Martinkus et al., 2023] learns agents that walk on a graph while recurrently updating their features. While optimizing the walk strategy, this may trade off speed as training requires recurrent roll-out of the network. Our method allows pairing simple and fast walkers, such as MDLR, with parallelizable NNs such as transformers. WalkLM [Tan et al., 2023a] proposed a fine-tuning method for language models on walks on text-attributed graphs. By utilizing anonymization, our approach is able to process graphs even if no text attribute is given. Lastly, a concurrent work [Wang and Cho, 2024] has arrived at a similar use of anonymization combined with RNNs.

Probabilistic invariant neural networks Whenever a learning problem is compatible with symmetry, incorporating the associated invariance structure to the hypothesis class often leads to generalization benefit [Bronstein et al., 2021, Elesedy, 2022, 2023]. This is also the case for probabilistic invariant

[5]While we show not forgetting $\mathbf{x}$ for brevity, we may extend to initial vertex v_0 using its anonymization as 1.

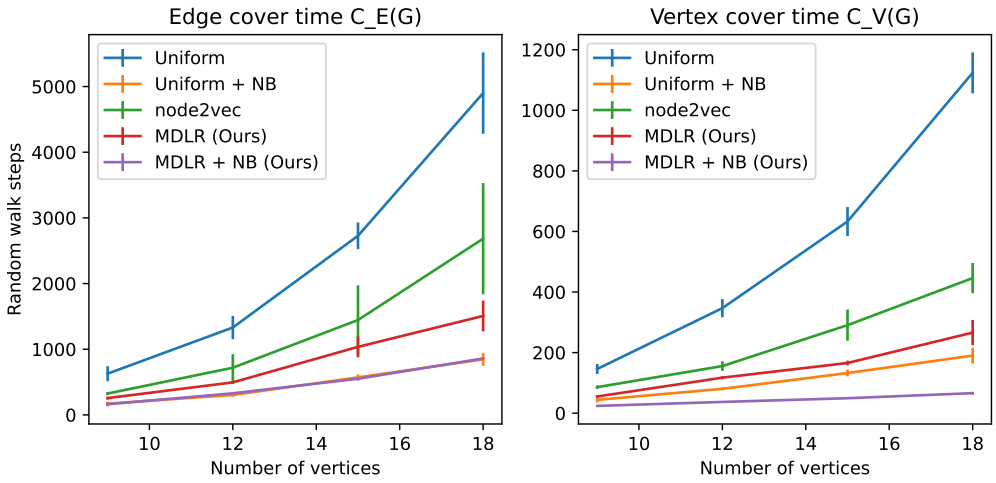


Figure 4.2: Cover times of random walks on varying sizes of lollipop graphs. NB is non-backtracking.

NNs [Lyle et al., 2020, Bloem-Reddy and Teh, 2020], which includes our approach. Probabilistic invariant NNs have recently gained interest due to their potential of achieving higher expressive powers compared to deterministic counterparts [Cotta et al., 2023, Cornish, 2024]. In graph learning, this is often achieved with stochastic symmetry breaking between vertices using randomized features [Loukas, 2020, Puny et al., 2020, Abboud et al., 2021, Kim et al., 2022a], vertex orderings [Murphy et al., 2019b, Kim et al., 2023], or dropout [Papp et al., 2021]. Our approach can be understood as using random walk as a symmetry-breaking mechanism for probabilistic invariance, which provides an additional benefit of natural compatibility with sequence learning methods.

4.1.4 Experiments

We perform a set of experiments to demonstrate RWNNs. We implement random walks in C++ based on Chenebaux [2020], which produces good throughput (<0.1 seconds for 10k steps) without using GPUs as in Tönshoff et al. [2023]; pseudocode is in Appendix A.7.2. We implement training and inference pipelines in PyTorch with NVIDIA GPUs and Intel Gaudi 2 compatibility.

Synthetic experiments

On synthetic setups, we first verify our claims on cover times in Section 4.1.1, focusing on the utility of MDLR (Equation (4.5)), non-backtracking, and named neighbor recording. We measure the edge cover times $C_E(G)$ and vertex cover times $C_V(G)$ on varying sizes of lollipop graphs G [Feige, 1995] which are commonly used to establish the upper bounds of cover times. While the strict definition of edge cover requires traversing each edge in both directions, our measurements only require traversing one direction, which suffices for the universality of RWNNs. The results are in Figure 4.2. We find that **(1)** MDLR achieves a significant speed-up compared to uniform and node2vec walks, **(2)** adding non-backtracking significantly improves all first-order walks, and **(3)** edge cover times are often much larger than vertex cover times, strongly justifying the use of named neighbor recording (Theorem 16).

Then, we verify our claims in Theorems 18 and 19 using Clique and Barbell datasets of Bamberger et al. [2024] that explicitly test for over-smoothing and over-squashing, respectively. Our RWNN uses MDLR walks with non-backtracking, and the reader NN is a 1-layer transformer encoder with width 128,

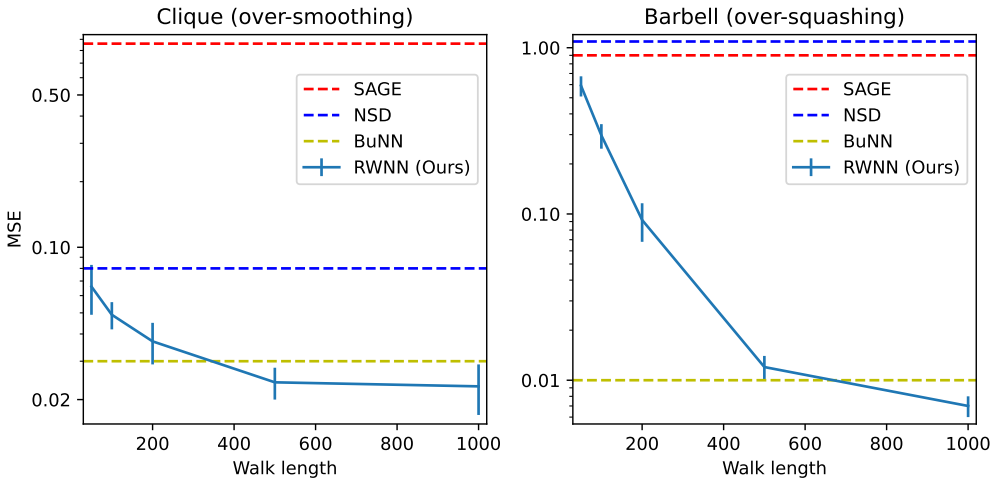


Figure 4.3: Over-smoothing and over-squashing (MSE ↓) for various walk lengths l.

Table 4.2: Over-smoothing and over-squashing results. We report test mean squared error (MSE ↓) aggregated for 4 randomized runs, except for CRaWl which took >3 days per run. Since CRaWl uses vertex-level average pooling, while RWNN-transformer uses walk-level, we also test a variant that uses walk-level pooling and denote it by CRaWl*.

	Clique (over-smoothing)	Barbell (over-squashing)
Baseline 1	30.94 ± 0.42	30.97 ± 0.42
Baseline 2	0.99 ± 0.08	1.00 ± 0.07
MLP	1.10 ± 0.08	1.08 ± 0.07
GCN	29.65 ± 0.34	1.05 ± 0.08
SAGE	0.86 ± 0.10	0.90 ± 0.29
GAT	20.97 ± 0.40	1.07 ± 0.09
$l = 100$		
WalkLM [Tan et al., 2023a]	0.096 ± 0.018	0.388 ± 0.048
RWNN-MLP	0.082 ± 0.018	0.326 ± 0.035
RWNN-transformer (Ours)	0.049 ± 0.007	0.297 ± 0.050
$l = 1000$		
CRaWl [Tönshoff et al., 2023]	0.103 ± N/A	0.289 ± N/A
CRaWl* [Tönshoff et al., 2023]	0.121 ± N/A	0.103 ± N/A
WalkLM [Tan et al., 2023a]	**0.023 ± 0.003**	0.037 ± 0.009
RWNN-MLP	0.028 ± 0.026	0.015 ± 0.005
RWNN-transformer (Ours)	**0.023 ± 0.006**	**0.007 ± 0.001**

matching the baselines. Figure 4.3 shows the results for varying walk lengths l. The model performs well on Clique overall, while Barbell requires scaling the walk length. This agrees with our claims that RWNNs in general avoid over-smoothing, but over-squashing manifests as probabilistic under-reaching (and is therefore mitigated by sufficiently long walks). With $l = 1000$, our model in Table 4.2 achieves the best performance among the considered methods. This supports our hypothesis that RWNNs can be resistant to over-smoothing and over-squashing (for the latter, if the walk is long), but also shows that proper design choices are important to obtain the best performance. For example, CRaWl and CRaWl* are limited by their window size of information processing, and RWNN-MLP is bottlenecked by the fixed-sized hidden feature. While WalkLM performs on par with RWNN-transformer on Clique, its error is higher than RWNN-MLP on Barbell, showing the benefits of our design of walks and their records.

Table 4.3: Isomorphism learning results. We report accuracy of classifying training data rounded to the first decimal point. *, †, ‡, §, and ◇ are from Papp et al. [2021], Murphy et al. [2019b], Zhao et al. [2022b], Martinkus et al. [2023], and Alvarez-Gonzalez et al. [2024], respectively.

	CSL 1, 2-WL fails	SR16 3-WL fails	SR25 3-WL fails
GIN [Xu et al., 2019]	10.0%†	50.0%§	6.7%‡
PPGN [Maron et al., 2019a]	100.0%§	50.0%§	6.7%‡
GIN-AK+ [Zhao et al., 2022b]	-	-	6.7%‡
PPGN-AK+ [Zhao et al., 2022b]	-	-	100.0%‡
GIN+ELENE [Alvarez-Gonzalez et al., 2024]	-	-	6.7%◇
GIN+ELENE-L (ED) [Alvarez-Gonzalez et al., 2024]	-	-	100.0%◇
GIN-RNI [Abboud et al., 2021]	16.0%*	-	-
GIN-RP [Murphy et al., 2019b]	37.6%†	-	-
GIN-Dropout [Papp et al., 2021]	82.0%*	-	-
RW-AgentNet [Martinkus et al., 2023]	100.0%§	50.5%§	-
AgentNet [Martinkus et al., 2023]	100.0%§	100.0%§	6.7%
CRaWl [Tönshoff et al., 2023]	100.0%§	100.0%§	46.6%
RWNN-DeBERTa (Ours)	100.0%	100.0%	100.0%

Graph isomorphism learning

We now verify the claims on expressive power in Theorem 16 using three challenging datasets where the task is recognizing the isomorphism type of input graph where certain WL test fails.

The Circular Skip Links (CSL) graphs dataset [Murphy et al., 2019b] contains 10 non-isomorphic regular graphs with 41 vertices of degree 4. Distinguishing these graphs requires computing lengths of skip links, and 1-WL and 2-WL tests fail. The 4×4 rook's and Shrikhande graphs [Alvarez-Gonzalez et al., 2024], which we call SR16, are a pair of strongly regular graphs with 16 vertices of degree 6. Distinguishing the pair requires detecting 4-cliques, and 3-WL test fails. The SR25 dataset [Balcilar et al., 2021] contains 15 strongly regular graphs with 25 vertices of degree 12, on which 3-WL test fails. Examples of the graphs and random walks on them can be found in Appendix A.7.2.

Our RWNN uses MDLR walks with non-backtracking, and recording function with anonymization and named neighbors that produces plain text (Algorithm 4). We use pre-trained DeBERTa-base language model as the reader NN to leverage its capacity, and fine-tune it with cross-entropy loss for $\leq$100k steps using AdamW optimizer with 2e-5 learning rate and 0.01 weight decay following Kim et al. [2023]. We truncate the input to 512 tokens due to memory constraints. We use batch size 256, and accumulate gradients for 8 steps for SR25, which we found sufficient to stabilize the training. At test time, we ensemble 4 predictions by averaging logits.

The results are in Table 4.3. MPNNs that align with certain WL tests fail when asked to solve harder problems, e.g., GIN aligns with 1-WL and fails in CSL. A limited set of the state-of-the art NNs solve SR25, but at the cost of introducing specialized structural features [Alvarez-Gonzalez et al., 2024]. Alternative approaches based on stochastic symmetry-breaking, e.g., random node identification, often fail on CSL although universal approximation is possible in theory [Abboud et al., 2021], possibly due to learning difficulties. For algorithms based on walks, AgentNet and CRaWl solve CSL and SR16, while failing on SR25. This is because learning a policy to walk in AgentNet can be challenging in complex tasks especially at the early stage of training, and the expressiveness of CRaWl is limited by the receptive field of the 1D CNN. Our approach based on DeBERTa language model overcomes the problems, demonstrating as the first RWNN that solves SR25.

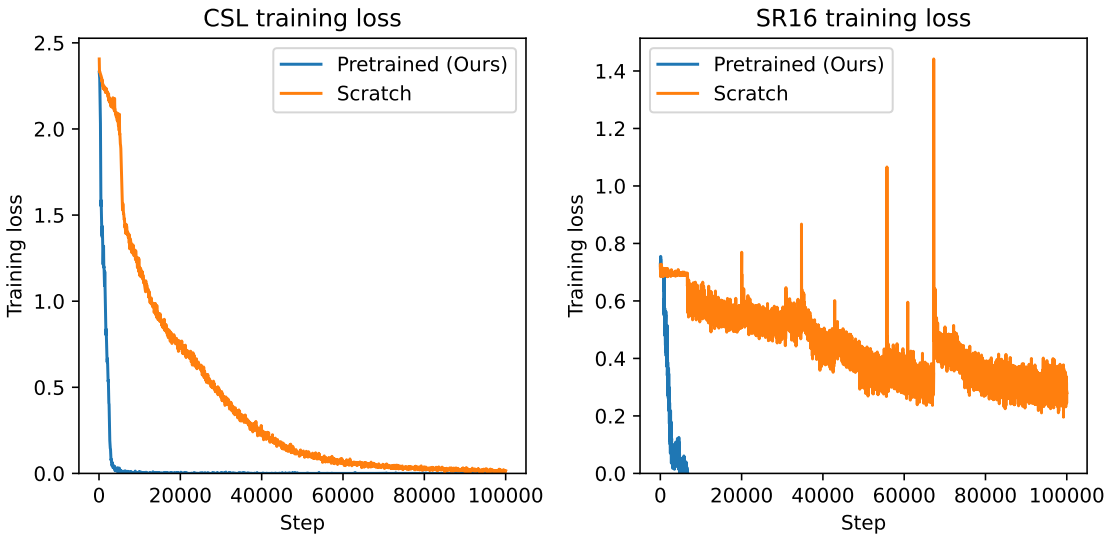


Figure 4.4: The impact of language pre-training on graph isomorphism learning.

In Appendix A.7.3, we further provide visualizations of learned attentions by mapping attention weights on text records of walks to input graphs. We find that the models often focus on sparse, connected substructures, which presumably provide discriminative information on the isomorphism types.

The impact of language pre-training We have used language pre-trained DeBERTa for graph isomorphism learning, even though the records of random walks do not resemble natural language (Appendix A.7.2 and A.7.3). To test if language pre-training is useful, we additionally trained RWNN-DeBERTa on CSL and SR16 datasets using the same configurations to our reported models but from random initialization. The training curves are shown in Figure 4.4. We find that language pre-training has significant benefits over random initialization for isomorphism learning tasks, as randomly initialized models converge very slowly for CSL, and fails to converge for SR16.

We conjecture that certain representations or computational circuits acquired in language pre-training are useful for processing text records of walks to recognize graphs. For example, some circuits specialized in recognizing certain syntactic structures in natural language could be repurposed during fine-tuning to detect skip links of CSL graphs from text records of walks. Our results are also consistent with prior observations on cross-model transfer of pre-trained language models to different domains such as image recognition [Lu et al., 2022, Rothermel et al., 2021].

Real-world transductive classification

Previous results considered undirected, unattributed, and relatively small graphs. We now show that RWNN based on Llama 3 [Dubey et al., 2024] language models can solve real-world problem on a large graph with directed edges and textual attributes. We use ogbn-arxiv [Hu et al., 2020a], a citation network of 169,343 arXiv papers with title and abstract. The task is transductive classification into 40 areas such as `"cs.AI"` using a set of labeled vertices.

We consider two representative types of baselines. The first reads title and abstract of each vertex using a language model and solves vertex-wise classification problem, ignoring graph structure. The second initializes vertex features as language model embeddings and trains an MPNN, at the risk of

Table 4.4: Test accuracy on ogbn-arxiv. † denotes using validation labels for label propagation or in-context learning following Huang et al. [2020]. For Llama 3, we ensemble 5 predictions by voting.

	Accuracy ↑
MLP [Hu et al., 2020a]	55.50%
node2vec [Hu et al., 2020a]	70.07%
GCN [Hu et al., 2020a]	71.74%
GraphSAGE [Hu et al., 2020a]	71.49%
GAT [Hu et al., 2020a]	73.91%
RevGAT [Hu et al., 2020a]	74.26%
Label propagation [Huang et al., 2020]	68.50%
C&S [Huang et al., 2020]	72.62%
C&S† [Huang et al., 2020]	74.02%
Llama2-13b zero-shot [He et al., 2023]	44.23%
GPT-3.5 zero-shot [He et al., 2023]	73.50%
DeBERTa, fine-tuned [Zhao et al., 2022a]	74.13%
Llama3-8b zero-shot	51.31%
Llama3-8b one-shot	52.81%
Llama3-8b one-shot†	52.82%
Llama3-70b zero-shot	65.30%
Llama3-70b one-shot†	67.65%
RWNN-Llama3-8b (Ours)	71.06%
RWNN-Llama3-8b (Ours)†	73.17%
RWNN-Llama3-70b (Ours)†	**74.85%**

over-compressing the text. To take the best of both worlds, we design the recording function so that, not only it does basic operations such as anonymization, it records a complete information of the local subgraph including title and abstract, edge directions, and notably, **labels in case of labeled vertices** (Appendix A.7.2) [Sato, 2024]. The resulting record naturally includes a number of input-label pairs of the classification task at hand, implying that we can frame transductive classification problem as a simple in-context learning problem [Brown et al., 2020]. This allows training-free application of Llama 3 [Dubey et al., 2024] language model for transductive classification. We choose restart rate $\alpha = 0.7$ (Section 4.1.1) by hyperparameter search on a 100-sample subset of the validation data.

The results are in Table 4.4. Our models based on frozen Llama 3 perform competitively against a range of previous MPNNs on text embeddings, as well as outperforming language models that perform vertex-wise predictions ignoring graph structures such as GPT-3.5 and fine-tuned DeBERTa. Especially, our model largely outperforms one-shot baselines, which are given 40 randomly chosen labeled examples (one per class). This is surprising as our model observes fewer vertices, 31.13 in average, due to other recorded information. This is presumably since the record produced by our algorithm informs useful graph structure to the language model to quickly learn the task in-context compared to randomly chosen shots. In Appendix A.7.3, we visualize the attention weights, verifying that the model makes use of the graph structure recorded by random walks to make predictions.

As a final note, our approach is related to label propagation algorithms [Zhu and Ghahramani, 2002, Zhu, 2005, Grady, 2006] for transductive classification, which makes predictions by running random walks and probing the distribution of visited labeled vertices. The difference is that, in our approach, the reader NN i.e. the language model can appropriately use other information such as attributes, as well as do meaningful non-linear processing rather than simply probing the input. As shown in Table 4.4, our approach outperforms label propagation, verifying our intuition.

Table 4.5: Fine-tuning for arXiv, extending Table 4.4. † uses validation labels as in Table 4.4.

	Training-free?	Accuracy ↑
Llama3-8b zero-shot	◯	51.31%
Llama3-8b one-shot	◯	52.81%
Llama3-8b one-shot†	◯	52.82%
Llama3-70b zero-shot	◯	65.30%
Llama3-70b one-shot	◯	67.65%
RWNN-Llama3-8b (Ours)	◯	71.06%
RWNN-Llama3-8b (Ours)†	◯	73.17%
RWNN-Llama3-70b (Ours)†	◯	74.85%
RWNN-Llama3-8b (Ours), fine-tuned	×	**74.95%**

Table 4.6: Transductive classification test accuracy on homophilic (Cora, Citeseer) and heterophilic (Amazon Ratings) datasets. The baselines scores are from Sato [2024], Zhao et al. [2023], Liu et al. [2024], Chen et al. [2024a,b], Platonov et al. [2023b]. † denotes using validation labels as in Table 4.4.

	Training-free?	Cora 20-shot	Cora	Citeseer	Amazon Ratings
Training-free GNN	◯	60.00%	-	-	-
GCN	×	81.40%	89.13%	74.92%	48.70%
GAT	×	80.80%	89.68%	75.39%	49.09%
GraphText [Zhao et al., 2023]	◯	68.30%	67.77%	68.98%	-
LLaGA [Chen et al., 2024a]	◯	-	59.59%	-	-
GraphText [Zhao et al., 2023]	×	-	87.11%	74.77%	-
LLaGA [Chen et al., 2024a]	×	-	88.86%	-	28.20%
OFA [Liu et al., 2024]	×	75.90%	-	-	51.44%
RWNN-Llama3-8b (Ours)	◯	72.29%	85.42%	79.62%	42.71%
RWNN-Llama3-8b (Ours)†	◯	79.84%	87.08%	80.41%	38.66%
RWNN-Llama3-8b (Ours), fine-tuned	×	79.45%	86.72%	80.72%	55.76%

Fine-tuning In Table 4.4, our RWNNs are training-free. We demonstrate fine-tuning the Llama 3 8b reader network using the training labels with quantized low-rank adaptation (QLoRA) [Hu et al., 2022, Dettmers et al., 2023]. The result is in Table 4.5, showing a promising performance.

Additional datasets We provide additional experiments on real-world transductive classification datasets Cora, Citeseer, and Amazon Ratings. Our RWNNs are designed similarly to the ones used for arXiv. We compile the baseline scores from Sato [2024], Zhao et al. [2023], Liu et al. [2024], Chen et al. [2024a,b], Platonov et al. [2023b], which includes MPNNs and language models on graphs. The results are in Table 4.6. Llama 3 based RWNN is competitive among training-free methods, and performs reasonably well when fine-tuned with QLoRA. Especially, on Amazon Ratings which is heterophilic [Platonov et al., 2023b], our fine-tuned RWNN-Llama3-8b achieves the highest accuracy (55.76%). This supports our theoretical results that RWNNs avoid over-smoothing, as avoiding it is known to be important for handling heterophily [Yan et al., 2022].

Real-world graph classification

We conduct a preliminary demonstration of RWNN on real-world graph classification, using the Peptides-func dataset [Dwivedi et al., 2022] used in Tönshoff et al. [2023]. Our model is a pre-trained DeBERTa-base (identical to Table 4.3) fine-tuned on text records of non-backtracking MDLR walks with anonymization and named neighbor recording. The recording function is designed to properly incorporate the vertex and edge attributes provided in the dataset, including the atom and bond types. An example

Table 4.7: Peptides-func graph classification. The baseline scores are from Tönshoff et al. [2024].

	Test AP ↑
GCN	0.6860
GINE	0.6621
GatedGCN	0.6765
Transformer	0.6326
SAN	0.6439
CRaWl	0.7074
RWNN-DeBERTa (Ours)	**0.7123 ± 0.0016**

text is in Figure A.12.

In Table 4.7, we report the test average precision (AP) at best validation accuracy, with 40 random predicted logits averaged at test time. We report the mean and standard deviation for five repeated tests. The baseline scores, including CRaWl, are from Tönshoff et al. [2023], Tönshoff et al. [2024]. We see that RWNN-DeBERTa, which has been successful in graph isomorphism learning (Table 4.3), also shows a promising result in real-world protein graph classification, despite its simplicity of recording random walks in plain text and processing them with a fine-tuned language model.

Role of cover times

In Section 4.1.1, we use cover times as a key tool for the analysis and comparison of different random walk algorithms. In the analysis, cover times provide the worst-case upper bounds of walk length required to achieve universality, which is in line with the literature on the cover times of Markov chains [Feige, 1995]. To demonstrate how the differences in cover times may translate to different model behaviors, we provide two examples where performing interventions that change cover times leads to substantial differences in task performances.

First, in SR16 graph separation, we test the use of named neighbor recording, which changes the required cover time from vertex cover time $C_V(G)$ (when used) to edge cover time $C_E(G)$ (when not used). The result is in Table 4.8. We can see that not using named neighbor recording has a drastic effect on the required cover time (vertex cover time 48.25 → edge cover time[6] 490.00), and the test performance deteriorates from perfect to random accuracy (100% → 50%).

Second, in arXiv transductive classification, due to the high homophily [Platonov et al., 2023a] we can assume that the cover time of the local neighborhoods, i.e., $B_r(v)$ for small r, has an important influence on performance. Thus, we set $r = 1$ and check if the walk strategy that lowers the cover time of $B_r(v)$ leads to a better performance. To test different local cover times, we use two choices of the restart probability $\alpha = 0.3$ and 0.7. Since we use named neighbor recording, we measure vertex cover times. Due to the large size of G, we estimate the cover times by fixing a random set of 100 test vertices and running 100 walks from each of them. The result is in Table 4.9. Increasing the restart probability (0.3 → 0.7) simultaneously results in a lower cover time (161.2 → 96.84) and a higher test accuracy (74.40% → 74.85%), even though the average number of observed vertices decreases (45.28 → 31.13). This implies that lower cover time is correlated with better performance, while the number of observed vertices itself is not directly correlated with better performance.

While the above shows cases where interventions in cover times lead to different task performances, in general, there are factors that make it not always trivial to equate task performances and cover times.

[6]As in Figure 4.2, while the strict definition of edge cover requires traversing each edge in both directions, our measurements only require traversing one direction, which suffices for the universality of RWNNs.

Table 4.8: The performance of RWNN-DeBERTa on SR16 graph separation (Table 4.3) for different required cover times controlled by the use of named neighbor recording.

	Test accuracy	Cover time
w/o named neighbor recording	50%	$C_E(G) = 490.00$
w/ named neighbor recording	**100%**	$C_V(G) = \mathbf{48.25}$

Table 4.9: The performance of RWNN-Llama3-70b† on arXiv transductive classification (Table 4.4) for different local cover times controlled by the restart probability α of the random walk.

	Test accuracy	Cover time $C_V(B_{r=1}(v))$	Unique vertices observed per walk
$\alpha = 0.3$	74.40%	161.2	**45.28**
$\alpha = 0.7$	**74.85%**	**96.84**	31.13

Table 4.10: Test performances of RWNN-Llama3-70b† (Table 4.4) and RWNN-DeBERTa (Table 4.7) for different numbers of randomized predictions for ensembling by voting and averaging, respectively. For *, we were unable to run repeated experiments due to sampling costs.

# samples	arXiv test accuracy
1	74.45 ± 0.049%
3	74.75 ± 0.035%
5	74.83 ± 0.028%
7	74.86% ± N/A*
10	**74.87% ± N/A***

# samples	Peptides-func test AP
1	0.5300 ± 0.0092
10	0.6937 ± 0.0039
20	0.7056 ± 0.0026
40	0.7123 ± 0.0016
60	0.7145 ± 0.0015
80	0.7155 ± 0.0014
100	0.7161 ± 0.0009
160	0.7176 ± 0.0013
320	**0.7177 ± N/A***

This is because a substantial amount of useful information in practice could be recovered from possibly non-covering walks.

Role of test-time ensembling

For each test input to an RWNN X_θ, we often sample several randomized predictions $\hat{\mathbf{y}} \sim X_\theta(\cdot)$ and ensemble them by averaging (or voting, for Llama 3 models where we cannot directly obtain continuous logits). As discussed in Section 4.1.1, this can be viewed as MC estimation of the underlying invariant mean predictor $(\cdot) \mapsto \mathbb{E}[X_\theta(\cdot)]$ (or the invariant limiting classifier in the case of voting [Cotta et al., 2023]). From another perspective, we can also interpret this as a type of test-time scaling with repeated sampling, which has been shown beneficial in language models [Brown et al., 2024] and randomized algorithms (called amplification [Cotta et al., 2023, Sieradzki et al., 2022]). To see if similar perspectives can be held for RWNNs, we measure the model performances with increased numbers of samples for averaging or voting. The results are in Table 4.10. We see that increasing sample size consistently leads to better performances and reduced variances overall. The point of plateau seems to differ per application and model architecture, and we leave obtaining a better understanding of the scaling behavior for future work.

Effective walk lengths

In our experiments, the effective lengths of walks are affected by the language model tokenizers since text records exceeding the maximum number of tokens are truncated. The maximum number of tokens

Table 4.11: Effective random walk lengths.

Dataset	Length
CSL (Table 4.3)	214.03 ± 2.48
SR16 (Table 4.3)	222.00 ± 0.00
SR25 (Table 4.3)	129.54 ± 2.26
arXiv (Table 4.4)	194.52 ± 10.59

is a hyperparameter which we determine from memory constraints; for the DeBERTa tokenizer used in Table 4.3, we set it to 512, and for the Llama 3 tokenizer used in Table 4.4, we set it to 4,096. The effective lengths of random walks for each dataset are measured and provided in Table 4.11. The effective length is shorter (129.54 on average) for SR25 graphs compared to CSL and SR16 (214.03 and 222.00 on average, respectively) although using the same tokenizer configurations, which is because the higher average degrees of SR25 graphs (Figure A.8) compared to CSL and SR16 graphs (Figures A.6 and A.7, respectively) has led to a higher number of text characters invested in recording named neighbors ("#" symbols in the figures).

4.1.5 Discussion

We contributed a principled understanding of RWNNs, where random walks on graphs are recorded and processed by NNs to make predictions. We analyzed invariance, expressive power, and information propagation, showing that we can design RWNNs to be invariant and universal without constraining its neural network. Experiments support the utility of our approach.

Our current main limitation is the cost of performing training and inference if using language models, especially on long walks. While random walk sampling and text recording can be done efficiently, the main computational bottleneck comes from the use of language models as the reader NN. For example, in Table 4.4, each test prediction of RWNN-Llama-3-70b takes around 0.410 ± 0.058 seconds on a 4× H100 machine (RWNN-Llama-3-8b is faster, 0.116 ± 0.064 seconds per prediction). The inference time of standard MPNNs are much shorter, e.g., a 3-layer GAT with 16 hidden dimensions takes around one second for all test vertices of ogbn-arxiv on a V100 GPU [Wang et al., 2019b]. Overcoming this issue is an important future direction. We remark that, if language model embeddings are used and fine-tuned, the training cost of MPNNs can also be substantial, e.g., recent work has reported that prior methods for training an MPNN jointly with language model embeddings on arXiv take 25-46 hours on a 6× RTX 3090 machine [Zhu et al., 2024, Table 3].

On the theory side, we focus on cover times of random walks in our analysis, which corresponds to the worst-case length bounds required for universal approximation. Yet, in practice, there is a substantial amount of useful information that can be already recovered from possibly non-covering walks. Examples include spectral properties [Benjamini et al., 2006], number of edges and mixing time [Ben-Hamou et al., 2018], and triangle counts [Bera and Seshadhri, 2020], among others studied in the field of sublinear algorithms. These results suggest that shorter walks may suffice in practice for RWNNs, if the task at hand requires properties that can be estimated without seeing the whole graph, and the reader NN can learn to recover them (e.g., leveraging universality). Understanding this aspect better is an important direction for future work.

Another question for future work is whether we can train a language model on a large pseudocorpus of random walks [Klubicka et al., 2019, 2020] to build a foundation model for graphs which is language-compatible. We plan to investigate this direction in the future.

4.2 Flock: A knowledge graph foundation model

Knowledge graph foundation models (KGFMs) [Lee et al., 2023, Geng et al., 2023, Galkin et al., 2024, Zhang et al., 2024, Cui et al., 2024, Huang et al., 2025] aim to infer missing links over novel knowledge graphs (KGs) that are not part of the training graphs or domains. This task requires generalization to *both* unseen nodes and unseen relation types. To achieve this, KGFMs rely on learning *node and relation invariants*: structural properties of nodes and relations that are transferable across KGs even when their relational vocabularies differ. This inductive bias is formalized by Gao et al. [2023] as double-equivariance — equivariance under permutations of both entities and relations — and used as a core principle in the design of KGFMs in the literature.

Problem statement In this work, we challenge the fundamental assumption of existing KGFMs dictated by strict equivariance: *structural isomorphism of relations implies semantic equivalence*. Consider, for example the KG from Figure 4.5, where the relations *like* and *dislike* are structurally isomorphic, and yet they represent semantically opposite relations. In this motivating example, any KGFM that computes relation invariants is forced to assign the same representation to both *like* and *dislike* — losing the ability to distinguish between two entities with opposite relationships. This expressiveness limitation is an architectural one and *cannot* be resolved through finetuning, which further limits the downstream use of existing KGFMs. This raises a central question: How can we design KGFMs that are both *expressive* and have the right *inductive bias* for generalization?

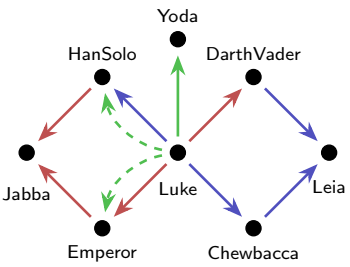


Figure 4.5: A KG representing characters' relationships in Star Wars movies. Blue arrows indicate like, red arrows – dislike, and green arrows indicate relation (friendWith).

Our approach We propose a new approach for KGFMs, which relies on *probabilistic* node-relation equivariance as inductive bias. Instead of enforcing *deterministic* equivariance over nodes and relations, these KGFMs respect *probabilistic* node-relation equivariance. This relaxes the hard constraint that "structurally isomorphic relations *must have* identical representations", and requires only that "the representations of structurally isomorphic relations need to be equivalent in distribution" over a model's stochastic processes. This way, the model retains crucial inductive bias needed for generalizing across different KGs, while the stochasticity of each forward pass ensures that structurally identical but semantically distinct relations are assigned different representations, allowing the model to distinguish between them [Srinivasan and Ribeiro, 2020, Abboud et al., 2021].

Inspired by the success of models that learn probabilistic invariants via random walks [Perozzi et al., 2014, Grover and Leskovec, 2016, Nikolentzos and Vazirgiannis, 2020, Kim et al., 2025c], we introduce Flock, a KGFM that inherently computes probabilistic node-relation invariants. Given a (potentially unseen) KG, and a query, in each iteration, Flock first samples a set of random walks over KG based on the query, noting down both encountered nodes and relations with a recording protocol. To ensure the model can generalize to unseen entities and relation types, we *anonymize* all nodes and relations, enforcing that Flock only learn from their structural roles. These anonymized sequences are then fed into a sequence processor, and the representations for each node and relation are aggregated via a consensus protocol. Finally, we construct per-query (triple) features from the aggregated entity and relation embeddings and input them into a binary classifier for link prediction.

Contributions The design of Flock offers several key advantages over existing KGFMs. First, it entirely abandons the conventional two-stage process of encoding relations and node representations via two separate networks, and does not rely on message-passing at all, thereby avoiding the well-known expressivity limitations of MPNNs on KGs [Barceló et al., 2022, Huang et al., 2023, 2025]. Second, Flock is a universal approximator (see Proposition 15), and thus can approximate every link-level function defined on KGs of any bounded size. Finally, Flock architecture inherently respects the principle of probabilistic node-relation equivariance, which enables a strong generalization capacity. Our experimental results over both entity prediction and relation prediction validate the strength of this approach, demonstrating that Flock consistently outperforms state-of-the-art KGFMs on existing benchmarks. Our contributions can be summarized as follows:

- We highlight a limitation in existing KGFMs: their over-reliance on deterministic node–relation equivariance prevents them from distinguishing between structurally similar but semantically different relations, limiting their expressivity.
- We propose to leverage *probabilistic node-relation equivariance*, a property for KGFMs that ensures invariance only in distribution, as an effective solution balancing expressivity and generalization.
- We propose Flock, a KGFM that respects probabilistic node-relation equivariance. Flock replaces the conventional two-stage, message-passing paradigm with a direct sequence encoding approach based on random walks, and acts as a universal approximator of link-level functions.
- We validate our approach on both entity and relation prediction tasks across 54 diverse KGs, where Flock consistently achieves state-of-the-art performance over existing KGFMs. We further design a synthetic dataset Petals to confirm our theoretical results empirically.

4.2.1 Related work

Link prediction and KGFMs Early methods for inferring missing links in KGs [Bordes et al., 2013, Sun et al., 2019, Balazevic et al., 2019, Abboud et al., 2020, Schlichtkrull et al., 2018, Vashishth et al., 2020] rely on learned embeddings, hence operating in the *transductive* setting, incapable of generalizing to unseen entities or relation types. Later GNN-based approaches based on the labeling trick [Teru et al., 2020, Zhang et al., 2021] or conditional message passing [Zhu et al., 2021, 2023, Zhang and Yao, 2022, Zhang et al., 2023c, Huang et al., 2023], unlocked the *node inductive* scenario, while remaining restricted to a fixed relational vocabulary. KGFMs eliminate this restriction and enable *node-relation inductive* link prediction over both unseen nodes and relation types through the use of a two-stage process by first encoding relations and then nodes. The first examples of this paradigm are InGram [Lee et al., 2023] and ULTRA [Galkin et al., 2024]. Their ideas were extended by TRIX [Zhang et al., 2024] to build a more expressive framework. KG-ICL [Cui et al., 2024] achieved full inductivity by combining in-context learning with node-relation tokenization. ISDEA [Gao et al., 2023] and MTDEA [Zhou et al., 2023] highlighted the benefits of equivariance over both nodes and relations. MOTIF [Huang et al., 2025] was proposed as a general KGFM framework, supported by a theoretical analysis of the expressive power of KGFMs. Our work further advances the field with a stochastic KGFM, which is invariant in probability and provably more expressive than all the existing methods. Notably, Flock achieves universality without any form of message passing, instead relying on random walks and sequence models to encode both nodes and relations anonymously to ensure generalization. This is distinct from previous stochastic KGFMs that rely on random initialization to message passing [Lee et al., 2023, Gao et al., 2023].

Random walks for graph representations Random walks have attracted a lot of attention in graph learning, due to their simplicity and ability to gather context from neighborhoods. DeepWalk [Perozzi et al., 2014] and node2vec [Grover and Leskovec, 2016] were among the first to explore the potential of random walks for producing graph embeddings, treating walks as analogues of sentences in natural language and processing them using skip-gram models. Nikolentzos and Vazirgiannis [2020] generated graph-level task predictions by executing joint random walks on direct products of graphs with their extracted subgraphs. CRaWL [Tönshoff et al., 2023] represents the input graph as a collection of random walks and processes them with a 1-dimensional convolutional NN. WalkLM [Tan et al., 2023b] samples random walks from graphs with textual features, passing them to a language model for embedding generation. RWNN [Kim et al., 2025c] and RUM [Wang and Cho, 2024] anonymize the extracted walks and process them with sequence models and RNNs, respectively. NeuralWalker [Chen et al., 2025] aggregates embeddings derived by encoding random walks into message passing layers.

Probabilistic invariance Neural architectures that enforce invariance to specific transformations often exhibit more stable training and improved performance [Bronstein et al., 2021], but this inductive bias can reduce their expressivity by preventing the model from distinguishing non-equivalent inputs. In graph learning, this trade-off is exemplified by MPNNs, whose power is limited by the 1-WL test [Xu et al., 2019, Morris et al., 2019a]. Randomization has emerged as a solution, enhancing expressivity through techniques such as noise injection [Abboud et al., 2021], vertex dropping [Papp et al., 2021], subgraph sampling [Bevilacqua et al., 2022b, Zhang et al., 2023a], dynamic rewiring [Finkelshtein et al., 2024], and random walks [Kim et al., 2025c, Wang and Cho, 2024]. Despite their stochasticity, such methods can remain probabilistically invariant, ensuring that equivalent inputs yield identical expected outputs, or even identical output distributions. In line with Gao et al. [2023], we extend the notion of probabilistic invariance to KGs, prove that Flock satisfies it, and further clarify its usefulness in KG learning.

4.2.2 Preliminary

Knowledge graphs A *knowledge graph* (KG) is a tuple $G = (V, E, R)$, where V denotes the set of entities (nodes), R the set of relation types, and $E \subseteq V \times R \times V$ the set of labeled edges (*facts*). A fact is written as (h, r, t) (or $h \xrightarrow{r} t$ interchangeably) with $r \in R$ and $h, t \in V$. A (potential) *link* in G is any triple (h, r, t) in $V \times R \times V$, regardless of whether it is present in E. We denote by R^{-1} the set of inverses of relations R, defined as $\{r^{-1} \mid r \in R\}$, and mean r when writing $(r^{-1})^{-1}$. Further, let $\mathbb{K}_{n,m}$ be the space of knowledge graphs with n vertices and m relation types.

Isomorphism An *isomorphism* between two knowledge graphs $G = (V, E, R)$ and $G' = (V', E', R')$ is a pair of bijections $\mu = (\pi, \phi)$, where $\pi : V \to V'$ and $\phi : R \to R'$, such that a fact (h, r, t) belongs to E if and only if the fact $\mu((h, r, t)) = (\pi(h), \phi(r), \pi(t))$ belongs to E'. Two KGs are *isomorphic* if such a mapping exists, in which case we write $G \simeq G'$.

Link invariance In this work, we focus on link-invariant functions. Let ω be a function assigning to each KG $G = (V, E, R) \in \mathbb{K}_{n,m}$ a map $\omega(G) : V \times R \times V \to \mathbb{R}^d$. We say that ω is *link invariant* if for every pair of isomorphic KGs $G, G' \in \mathbb{K}_{n,m}$, every isomorphism (π, ϕ) from G to G', and every link (h, r, t) in G, we have $\omega(G)((h, r, t)) = \omega(G')((\pi(h), \phi(r), \pi(t)))$.

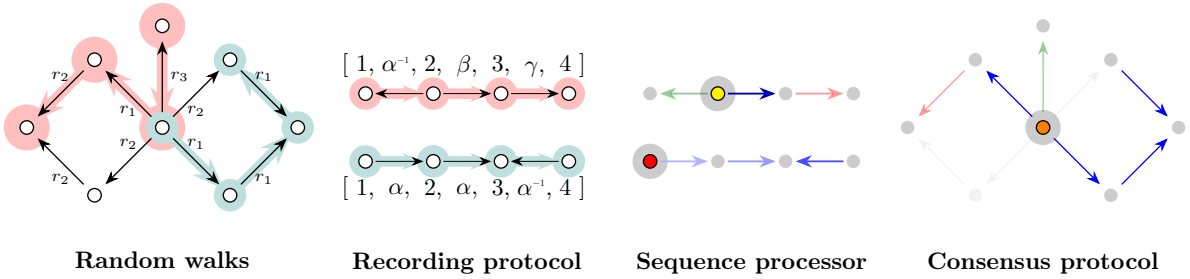


Figure 4.6: Overall pipeline of Flock. In each updating step, Flock 1) samples random walks on the KG (two walks indicated by red and teal, respectively), 2) anonymizes the encountered nodes and relations via a recording protocol (for each walk, nodes are anonymized as $1, 2, ...$ and relations as $\alpha, \beta, ...$), and 3) feeds the sequences in a sequence processor to compute node and relation representations. 4) A consensus protocol then pools them back to the original KG's nodes and relations.

Probabilistic invariance A stochastic KG model φ can be viewed as a function that takes a KG and returns a random variable $\varphi(G)$. Following Kim et al. [2025c], we call φ *invariant in probability* if

$$\forall G, G' \in \mathbb{K}_{n,m} : \qquad G \simeq G' \implies \varphi(G) \overset{d}{=} \varphi(G')$$

i.e. the distributions of $\varphi(G)$ and $\varphi(G')$ are equal. In particular, this implies $\mathbb{E}[\varphi(G)] = \mathbb{E}[\varphi(G')]$.

4.2.3 Method

We present Flock, a KGFM respecting probabilistic node-relation invariance. Flock is a randomized function $X_\theta(\cdot)$ which takes as input a KG $G = (V, E, R)$ and a link prediction query q. We consider two types of queries: *entity prediction* $q = (h, r, ?)$ and *relation prediction* $q = (h, ?, t)$. Flock outputs a random variable $\hat{\mathbf{y}} \sim X_\theta(G, q)$ which is suited for the task at hand. For entity prediction, it outputs $\hat{\mathbf{y}} : V \to [0, 1]$ such that a potential link (h, r, t) can be evaluated by $\hat{\mathbf{y}}(t) \in [0, 1]$. For relation prediction, it outputs $\hat{\mathbf{y}} : R \to [0, 1]$ such that a link (h, r, t) can be evaluated by $\hat{\mathbf{y}}(r)$.

At test time, we average multiple (P) independent stochastic predictions to produce the final output. This improves performance and reduces variance through an ensembling effect.

We now describe the architecture of Flock, focused on four main components, and then analyze its theoretical properties in Section 4.2.4, showing universality and probabilistic equivariance. An expansion on the model details can be found in Section A.8.2.

Architecture of Flock

Internally, Flock has two lookup tables of hidden states, $\mathbf{v} : V \to \mathbb{R}^d$ for entities and $\mathbf{r} : R \to \mathbb{R}^d$ for relations, respectively. At each forward pass, it starts from trained initializations of these states $\mathbf{v}^{(0)}(\cdot) \coloneqq \mathbf{v}_0$ and $\mathbf{r}^{(0)}(\cdot) \coloneqq \mathbf{r}_0$, and updates them iteratively $\mathbf{v}^{(i)}, \mathbf{r}^{(i)}$ for $i \leq I$. Each update is done residually using a randomized function U_{θ_i}:

$$\mathbf{v}^{(i+1)} \coloneqq \mathbf{v}^{(i)} + \Delta\mathbf{v}, \qquad \mathbf{r}^{(i+1)} \coloneqq \mathbf{r}^{(i)} + \Delta\mathbf{r}, \qquad (\Delta\mathbf{v}, \Delta\mathbf{r}) \sim \text{update}_{\theta_i}(\mathbf{v}^{(i)}, \mathbf{r}^{(i)}).$$

The final hidden states $\mathbf{v}^{(I)} : V \to \mathbb{R}^d$ and $\mathbf{r}^{(I)} : R \to \mathbb{R}^d$ are then processed by a binary classifier head $: \mathbb{R}^d \to [0, 1]$ to produce the output $\hat{\mathbf{y}}$ which is $V \to [0, 1]$ or $R \to [0, 1]$ depending on task.

We now describe the randomized update_θ. We drop i for brevity. It consists of four components:

1. **Random walk algorithm** produces n random walks $\eta_1, ..., \eta_n$ of length ℓ on the input KG.
2. **Recording protocol** $w : \eta_j \mapsto \mathbf{z}_j$ transforms each walk into a graph-agnostic sequence.
3. **Sequence processor** $f_\theta : \mathbf{z}_j \mapsto \mathbf{h}_j$ processes each sequence independently, outputting features.
4. **Consensus protocol** $c : (\mathbf{h}_{1:N}, \eta_{1:N}) \mapsto (\Delta\mathbf{v}, \Delta\mathbf{r})$ collects features of all walks and decides state updates for each entity and relation type.

An overview is presented in Figure 4.6. We note that w, f_θ, and c are all deterministic, and the random walk is the only source of stochasticity. We now discuss the design choice for each. For the ease of exposition, we explain for entity prediction tasks $q = (h, r, ?)$, but relation prediction is similar.

Random walks In Flock, random walks are central in two ways: they rewrite the connectivity of nodes and relations as sequences, and support generalization via probabilistic equivariance.

Formally, the random walk algorithm produces n random walks $\eta_1, ..., \eta_n$ of length ℓ on KG G. Each random walk η is a chain of random variables, written as:n as:

$$\eta = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \cdots \xrightarrow{r_\ell} v_\ell, \qquad v_s \in V, r_s \in R, (v_{s-1}, r_s, v_s) \in E,$$

where the underlying transition mechanism and ℓ are hyperparameters.

To support probabilistic equivariance, we ask the walk algorithm to be invariant in probability. We say η is invariant in probability if for any $G \simeq H$ in $\mathbb{K}_{n,m}$ with isomorphism (π, ϕ) from G to H:

$$\pi(v_0) \xrightarrow{\phi(r_1)} \pi(v_1) \xrightarrow{\phi(r_2)} \cdots \xrightarrow{\phi(r_\ell)} \pi(v_\ell) \overset{d}{=} u_0 \xrightarrow{s_1} u_1 \xrightarrow{s_2} \cdots \xrightarrow{s_\ell} u_\ell$$

where $v_0 \xrightarrow{r_1} \cdots \xrightarrow{r_\ell} v_\ell$ and $u_0 \xrightarrow{s_1} \cdots \xrightarrow{s_\ell} u_\ell$ follow the distributions of $\eta(G, \ell)$ and $\eta(H, \ell)$, respectively. In such case, the isomorphism (π, ϕ) yields a natural translation from walks in G to H.

In Flock, we use a simple random walk algorithm which we show to be invariant in probability. Specifically, we use uniform walks with non-backtracking, with minor modifications to handle directed multi-edges of KGs. Despite the simplicity, we find that this choice works well in practice, consistent with findings of prior works [Tönshoff et al., 2023, Kim et al., 2025c].

Under this choice, we diversify the starting locations of walks such that local context around the query q and broad regions of the nodes and relations in a KG are both well-captured. Our *diversification strategy* is as follows: given a base walk count n, for entity prediction queries $(h, r, ?)$, we use $3n$ walks with three types of start locations. The first n walks start at query node h, capturing local context around the query; the second n walks start by traversing a random edge (v, s, u) where s is a uniformly chosen relation, broadly capturing the relations of the KG including r; the last n walks start at random nodes, broadly capturing various regions of the KG. For relation prediction queries $(h, ?, t)$, we additionally start n walks at the tail node t, sampling a total of $4n$ walks.

We lastly discuss how to choose the base walk count n. While this is a fixed hyperparameter n_{train} at pretraining, we find that scaling it adaptively to input KG at test-time benefits size generalization. We thus propose *test-time adaptation of walk counts*, and use:

$$n = n_{\text{train}} \times \text{harmonic mean}\left(\frac{|V|}{|V|_{\text{train}}}, \frac{|E|}{|E|_{\text{train}}}\right) \tag{4.8}$$

where $|V|_{\text{train}}, |E|_{\text{train}}$ are average numbers of nodes and edges in pretraining KGs, respectively. Intuitively, this scales n proportionally to the sizes of test KGs relative to pretraining. In practice, we clamp n to the nearest power of 2 and limit its value in an interval to avoid out-of-memory errors.

Recording protocol While random walks provide a basis for invariant sequence representations of KGs, two issues remain: (1) They reveal nodes v_s and relations r_s specific to each KG which obstructs transferability to unseen KGs. (2) They do not offer a way to condition on current states of entities $\mathbf{v}$, relations $\mathbf{r}$, and the query $q = (h, r, ?)$ as often done in KGFMs via the labeling trick.

The recording protocol $w : \eta_j \mapsto \mathbf{z}_j$ resolves this by transforming each walk into a *graph-agnostic* sequence that only leaves structural information. Motivated by prior works on node anonymization for invariance [Kim et al., 2025c, Wang and Cho, 2024], we propose an extension called node-relation anonymization: reserve separate namespaces for nodes and relations, respectively, and assign unique names in the order of discovery. For example, with $1, 2, 3, ...$ for nodes and $\alpha, \beta, ...$ for relations:

$$\eta = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} v_2 \xrightarrow{r_1^{-1}} v_0 \qquad \mapsto \qquad 1 \xrightarrow{\alpha} 2 \xrightarrow{\beta} 3 \xrightarrow{\alpha^{-1}} 1,$$

where $(\cdot)^{-1}$ marks direction of a relation. The protocol additionally employs a simple conditioning on current states $(\mathbf{v}, \mathbf{r})$ and query $q = (h, r, ?)$, completing the record $\mathbf{z}$ as follows:

$$w : \eta \mapsto \mathbf{z} = (1, \mathbf{v}(v_0), \mathbf{1}_h(v_0)) \xrightarrow{\alpha, \mathbf{r}(r_1), \mathbf{1}_r(r_1)} (2, \mathbf{v}(v_1), \mathbf{1}_h(v_1)) \xrightarrow{\beta, \mathbf{r}(r_2), \mathbf{1}_r(r_2)} \cdots, \tag{4.9}$$

with indicator functions $\mathbf{1}_h(\cdot), \mathbf{1}_r(\cdot)$ at h and r, respectively. As we will show, the recording protocol keeps node-relation invariance by hiding nodes and relations while leaving their structural roles.

Sequence processor Now that the recordings $\mathbf{z}$ only encode structural information of KG, we can safely process them with an arbitrary neural network $f_\theta : \mathbf{z} \mapsto \mathbf{h}$ without the risk of losing invariance. Since $\mathbf{z}$ are sequences, we choose sequence networks to leverage their inductive bias. Specifically, we use bidirectional GRU [Cho et al., 2014] equipped with RMSNorm [Zhang and Sennrich, 2019] and SwiGLU feedforward network [Shazeer, 2020], which provided robust results. To convert anonymizations into input feature vectors to the GRU, we use trainable embedding tables.

Given that f_θ is a sequence network, it is convenient to interpret its output $\mathbf{h}$ as positionally aligned with each step of the walk η or record $\mathbf{z}$. Specifically, for the example in Equation (4.9), we obtain:

$$f_\theta : \mathbf{z} \mapsto \mathbf{h} = (\Delta\mathbf{v}_0, a_0) \xrightarrow{\Delta\mathbf{r}_1, b_1} (\Delta\mathbf{v}_1, a_1) \xrightarrow{\Delta\mathbf{r}_2, b_2} \cdots.$$

where $\Delta\mathbf{v}_s, \Delta\mathbf{r}_s \in \mathbb{R}^{h \times d_h}$ and $a_s, b_s \in \mathbb{R}^h$ are the decoded outputs at each position using linear projections. Intuitively, $\Delta\mathbf{v}_s, \Delta\mathbf{r}_s$ encode proposals of state updates for entities and relations by f_θ, and a_s, b_s encode respective confidences of f_θ for the proposed updates. This separation is useful due to the localized, pure-structure nature of the recordings $\mathbf{z}$. If a random walk η densely visited a cycle-like region and then terminated in a dangling manner, it is natural to assign more confidence to the cycle-like region of the structural encodings $\mathbf{h}$, and less confidence to the dangling region.

Consensus protocol After sequence processing, we are left with a handful of state update proposals $\mathbf{h}_{1:N}$ from f_θ, that are positionally aligned with random walks $\eta_{1:N}$ on KG. The consensus protocol c uses the information to decide final state updates $\Delta\mathbf{v} : V \to \mathbb{R}^d$ and $\Delta\mathbf{r} : R \to \mathbb{R}^d$.

Since c can access how each $\Delta\mathbf{v}_s$ within $\mathbf{h}_j$ is associated to a node $v_s \in V$ (and how each $\Delta\mathbf{r}_s$ is associated to a relation $r_s \in R$) through the random walk η_j, a simple way to form a consensus is by finding all proposals $\{\Delta\mathbf{v}_s\}$ associated to each node v, and all $\{\Delta\mathbf{r}_s\}$ associated to each relation r, and take averages of these proposals. The drawback is that uninformative proposals from e.g., dangling regions of walks are not directly suppressed, and can affect the state updates.

We can leverage the confidences a_s, b_s from f_θ to alleviate this issue. For each node $v \in V$ or relation $r \in R$, we first find all respective associated pairs $\{(\Delta\mathbf{v}_s, a_s)\}$ or $\{(\Delta\mathbf{r}_s, b_s)\}$ of proposals and confidences, and compute a multi-head softmax-normalized weighted average:

$$\Delta\mathbf{v}(v) \coloneqq \left[\sum \exp(a_s) \odot \Delta\mathbf{v}_s\right] \oslash \sum \exp(a_s) \qquad \Delta\mathbf{r}(r) \coloneqq \left[\sum \exp(b_s) \odot \Delta\mathbf{r}_s\right] \oslash \sum \exp(b_s),$$

where $\odot$ and $\oslash$ are row-wise multiplication and division, respectively. Intuitively, this normalization induces competition between state update proposals, naturally leading to uninformative proposals being suppressed. Similar ideas are presented by Locatello et al. [2020].

Again, we can show that the consensus protocol does not operate in a way specific to particular KGs, and hence retains node-relation equivariance.

4.2.4 Theoretical analysis

Expressivity Following the notion of probabilistic expressivity introduced by Abboud et al. [2021], we say that a Flock model X_θ is a universal approximator of link invariant functions over $\mathbb{K}_{n,m}$ if for any link invariant $\varphi : \mathbb{K}_{n,m} \to (V \times R \times V \to [0,1])$ and any $\epsilon, \delta > 0$, there exists a choice of the network parameters θ and the length of the sampled random walks ℓ, such that:

$$\mathbb{P}(|\varphi(G)((h,r,t)) - X_\theta(G,(h,r,?))(t)| < \epsilon) > 1 - \delta$$

for all graphs $G = (V, E, R) \in \mathbb{K}_{n,m}$ and all links $(h, r, t) \in V \times R \times V$.

Proposition 15. *With a powerful enough sequence processor f_θ, the Flock framework described above is a universal approximator of link invariant functions over $\mathbb{K}_{n,m}$ for all pairs (n, m).*

All proofs are in Section A.8.1. To offer an intuition behind the result, we provide a proof sketch.

Proof sketch. A sufficiently long random walk will cover all edges of the graph with high probability. Then, from its anonymized version, assigning unique positional identifiers to every node and relation, one can reconstruct the input graph, up to isomorphism. Thus, with a sufficiently expressive sequence processor, Flock can approximate any link-invariant function.

Invariance Despite the stochastic nature of our framework, beyond randomized node embeddings [Abboud et al., 2021], Flock can be provably guaranteed to satisfy probabilistic invariance:

Proposition 16. *Suppose that the walk sampling protocol η is invariant in probability and both the recording protocol w and the consensus protocol c are invariant. Then, regardless of the choice of the deterministic sequence processor f_θ, the corresponding Flock model is invariant in probability.*

Proof sketch. Since each of these components is invariant (in probability), and invariance of individual component is preserved under composition, we have that Flock is invariant.

Moreover, the designs of Flock's components provided earlier in this section satisfy the conditions of Proposition 16. Therefore, the suggested pipeline is indeed invariant in probability:

Proposition 17. *Any Flock model with components as outlined in this section, and detailed in Section A.8.2 is invariant in probability.*

4.2.5 Experiments

We test Flock over a wide range of KGs for both entity and relation predictions, aiming to answer:

Q1. Can Flock approximate functions that existing KGFMs cannot?

Q2. How does Flock generalize to unseen entities and relations compared to existing KGFMs?

Q3. How does performance scale with the sizes of pretraining graph mix and test-time ensemble?

Q4. What is the impact of choices of each component on the behavior and performance of Flock?

Q5. How do current KGFMs perform under noise injection, and how do they compare with Flock?

Synthetic dataset

To validate the limitations of KGFMs with node-relation equivariance (**Q1**), we construct a synthetic benchmark Petals. It contains 220 instances, each including: **(1)** a KG $G = (V, E, R)$ consisting of a 'central' node s, a 'stem' $T \subset V$ with query relation r_0, and multiple cyclic 'petals', each 'colored' with a different pair of relations in $R \setminus \{r_0\}$, **(2)** an entity prediction query $(h, r_0, ?)$ with $h \in \{s\} \cup T$, and **(3)** two candidate targets t_1 and t_2 from the same 'petal', located at the same distance from s. An example is in Figure 4.7.

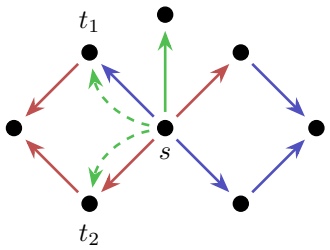


Figure 4.7: Example KG from Petals. KGFMs with relational invariants must equate blue r_1 and red r_2, thus predicting the same scores for both dashed queries with r_0.

Petals is designed such that each instance always admits non-trivial automorphisms, meaning that swapping relations occurring in the same 'petal' results in an isomorphic KG. Consequently, any model computing relation invariants will not be able to distinguish between potential links (s, r_0, t_1) and (s, r_0, t_2). However, the samples are constructed so that these links are not isomorphic from the graph perspective, making them distinguishable for general link-invariant functions. We say a model *solves* an instance if it can classify (s, r_0, t_1) as TRUE and (s, r_0, t_2) as FALSE.

We train ULTRA [Galkin et al., 2024], MOTIF($\mathcal{F}^3_{\text{Path}}$) [Huang et al., 2025], TRIX [Zhang et al., 2024], and Flock from scratch and validate them on the training instances.

The results are in Table 4.12. As expected, all existing KGFMs relying on learning deterministic relational invariants fail to distinguish between the candidate target triplets completely, achieving 50% accuracy due to random guesses. In contrast, Flock succeeds on *all* considered instances, displaying that, while remaining invariant in probability, it can differentiate between non-isomorphic links, even with isomorphic relations.

Table 4.12: Accuracies of KGFMs on the Petals benchmark.

Model	Petals
ULTRA	50%
MOTIF($\mathcal{F}^3_{\text{Path}}$)	50%
TRIX	50%
Flock	**100%**

Table 4.13: Average entity prediction MRR and Hits@10 over 54 KGs from distinct domains.

Model	Inductive e, r (23 graphs)		Inductive e (18 graphs)		Transductive (13 graphs)		Total Avg (54 graphs)		Pretrained (3 graphs)	
	MRR	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**	**MRR**	**H@10**
ULTRA (zero-shot)	0.345	0.513	0.431	0.566	0.312	0.458	0.366	0.518	-	-
TRIX (zero-shot)	0.368	0.540	0.455	0.592	0.339	0.500	0.390	0.548	-	-
Flock (zero-shot)	**0.369**	**0.554**	**0.456**	**0.604**	**0.340**	**0.509**	**0.391**	**0.560**	-	-
ULTRA (finetuned)	0.397	0.556	0.440	0.582	0.379	0.543	0.408	0.562	0.407	**0.568**
TRIX (finetuned)	0.401	0.556	0.459	0.595	**0.390**	**0.558**	0.418	0.569	**0.415**	0.564
Flock (finetuned)	**0.417**	**0.576**	**0.473**	**0.619**	0.383	0.544	**0.427**	**0.582**	**0.415**	0.561

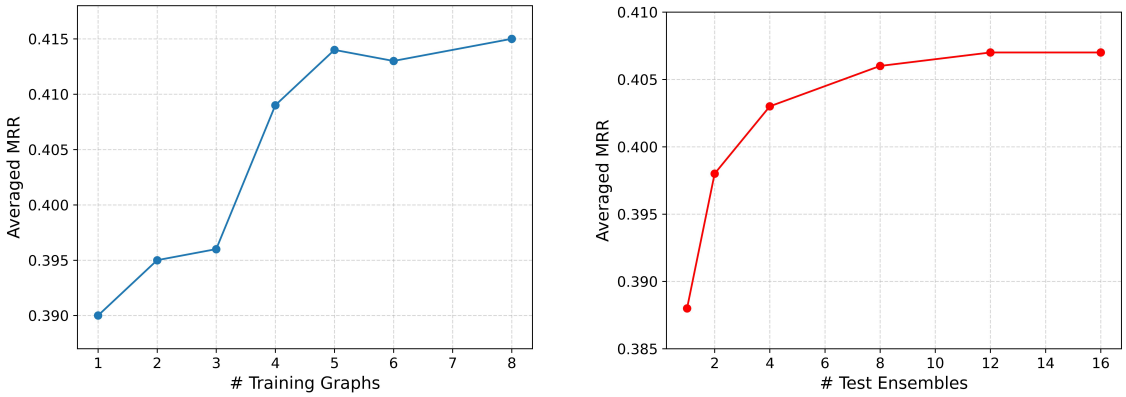
Table 4.14: Average relation prediction MRR and Hits@1 over 54 KGs from distinct domains.

Model	Inductive e, r (23 graphs)		Inductive e (18 graphs)		Transductive (13 graphs)		Total Avg (54 graphs)		Pretrained (3 graphs)	
	MRR	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**	**MRR**	**H@1**
ULTRA (zero-shot)	0.785	0.691	0.714	0.590	0.629	0.507	0.724	0.613	-	-
TRIX (zero-shot)	0.842	0.770	0.756	0.611	0.752	0.647	0.792	0.687	-	-
Flock (zero-shot)	**0.898**	**0.846**	**0.864**	**0.782**	**0.873**	**0.813**	**0.881**	**0.817**	-	-
ULTRA (finetuned)	0.823	0.741	0.716	0.591	0.707	0.608	0.759	0.659	0.876	0.817
TRIX (finetuned)	0.850	0.785	0.759	0.615	0.785	0.693	0.804	0.706	0.879	0.797
Flock (finetuned)	**0.929**	**0.889**	**0.887**	**0.808**	**0.897**	**0.844**	**0.907**	**0.851**	**0.977**	**0.959**

Entity and relation prediction over knowledge graphs

To answer **Q2**, we follow the protocol of Galkin et al. [2024], Zhang et al. [2024] and pretrain Flock on FB15k-237 [Toutanova and Chen, 2015], WN18RR [Dettmers et al., 2018], and CoDEx Medium [Safavi and Koutra, 2020]. We then evaluate its zero-shot and finetuned inference performance with the test set of 54 KGs. These KGs are extracted from diverse domains across three settings: inductive on nodes and relations (**Inductive** e, r), inductive on nodes (**Inductive** e), and **transductive**. Note that these settings differ only during finetuning setup; in zero-shot setup, all entities and relations are unseen. We choose state-of-the-art KGFMs ULTRA [Galkin et al., 2024] and TRIX [Zhang et al., 2024] as baselines, as they are pretrained on the same KGs, to ensure a fair comparison. For evaluation, we use the filtered ranking protocol [Bordes et al., 2013], reporting mean reciprocal rank (MRR) and Hits@10 for entity prediction, and Hits@1 for relation prediction, as some KGs have fewer than 10 relations.

Entity prediction results We present the entity prediction results in Table 4.13. In the zero-shot setting, Flock consistently outperforms ULTRA and TRIX on all metrics, demonstrating its strong generalization on KGs over diverse domains. Notably, on *Metafam* [Zhou et al., 2023], a dataset designed to challenge models with conflicting and compositional relational patterns, Flock roughly doubles MRR over ULTRA and achieves about a 40% MRR gain over TRIX in zero-shot setting. We find that Flock distinguishes structurally similar but semantically conflicting relations while ULTRA fails, which explains the gain (Figure 4.9). These findings align with our hypothesis that probabilistic node–relation equivariance improves expressivity without sacrificing generalization. In the finetuning setting, we observe a similar pattern: Flock maintains a consistent improvement over all datasets except transductive ones, where the underlying KGs are generally larger. We hypothesize that this gap stems from random walk coverages. Unlike ULTRA and TRIX whose message passing guarantees a full k-hop neighborhood



(a) Zero-shot MRR vs. #pretraining graphs.

(b) Zero-shot MRR vs. #ensembled predictions.

Figure 4.8: Pretraining and test-time scaling of Flock on 41 inductive KG datasets.

coverage over the queried node, Flock relies on sampled walks, which may not fully cover the target nodes of interest. We find that Flock favors sparse KGs (Figure 4.10), which agrees with this hypothesis as random walks cover sparse graphs faster.

Relation prediction results Table 4.14 presents the relation prediction results. Flock substantially outperforms all existing KGFMs across all categories in the zero-shot setting, achieving an 11.2% relative improvement in MRR compared to the best baseline TRIX. Flock shows a further performance boost of 12.8% in the finetuned setting. We attribute this huge gain to Flock's joint encoding of entities and relations during the updating step via the sequence encoder, while existing KGFMs, ULTRA and TRIX, have separate update steps for entities and relations. This joint update mechanism yields more holistic representations of both entities and relations with minimal information loss.

Scaling analysis

To assess whether Flock benefits from more pretraining graph and data (**Q3**), we follow the setup of Galkin et al. [2024], and pretrain Flock on an increasing number of KGs. We then evaluate them on all 41 inductive benchmarks for a fair comparison. As shown in Figure 4.8a, Flock's generalization improves consistently as the number of pretraining KGs increases, exhibiting clear scaling behavior, which is a core characteristic of being a foundation model.

Then, to assess how test-time ensemble size P affects performance (**Q3**), we take the pretrained Flock and run zero-shot entity prediction on 41 inductive KGs by increasing the number of ensembled passes. As shown in Figure 4.8b, performance improves from 1 to 8 passes and then begins to plateau beyond 12. This indicates a clear scaling behavior: larger ensembles provide a more accurate estimate of the underlying node and relation distributions.

Ablation studies

To understand the impact of design choices on the performance and behavior of Flock (**Q4**), we conduct a series of ablation studies spanning random walks, sequence processor and the consensus protocol, in the entity prediction task in the zero-shot setting.

Table 4.15: Ablation study of adaptive test-time walks with zero-shot entity prediction task. We show the average number of entities $|V|$, triples $|E|$, base walks n, MRR, and Hits@10.

Dataset split	Statistics		Flock			Flock w/o Adap.		
	$\|V\|$	$\|E\|$	n	MRR	Hits@10	n	MRR	Hits@10
Inductive e, r	5,303	10,838	28.40	**0.369**	**0.554**	128	0.357	0.551
Inductive e	7,578	29,090	18.08	**0.456**	**0.604**	128	0.441	0.596
Transductive	47,810	387,491	214.15	**0.340**	**0.509**	128	0.334	0.493

Table 4.16: Detailed ablation study with zero-shot entity prediction task. For the transductive split, considering resource limits, we test NELL995, NELL23k, WDsinger, ConceptNet100k, and YAGO310.

Model	**Inductive** e, r (23 graphs)		**Inductive** e (18 graphs)		**Transductive** (5 graphs)		**Total Avg** (46 graphs)	
	MRR	**Hits@10**	**MRR**	**Hits@10**	**MRR**	**Hits@10**	**MRR**	**Hits@10**
Flock ($\ell = 128$)	0.369	0.554	0.456	0.604	0.360	0.542	**0.395**	**0.567**
w/o non-backtracking	0.370	0.549	0.456	0.605	0.334	0.499	0.386	0.551
$\ell = 64$	0.372	0.556	0.459	0.606	0.351	0.534	0.394	0.565
$\ell = 256$	0.360	0.548	0.458	0.605	0.338	0.508	0.385	0.553
w/o diverse starts	0.360	0.539	0.448	0.596	0.319	0.488	0.385	0.553
transformer f_θ	0.356	0.542	0.410	0.591	0.312	0.477	0.359	0.537
w/o weighted consensus	0.351	0.526	0.448	0.593	0.361	0.515	0.387	0.545

Adaptive test-time walks Recall that we employ *test-time adaptation of walk counts*, which adaptively selects the base walk count n based on the graph size, computed via the harmonic-mean rule shown in Equation (4.8) during inference. Table 4.15 compares this adaptive setting with a fixed setting that uses 128 base walks per sample for all datasets, matching the pretraining setup ($n_{\text{train}} = 128$). As expected, the average selected base count n is smaller on both inductive splits and larger on the transductive split, yet the adaptive mechanism improves performance across all settings. This is consistent with the intuition that adaptive n scales up walks on larger KGs to improve coverage while allocating fewer walks on smaller KGs to reduce redundant visits; Flock maintains comparable *visiting rates* and *coverage* to those seen during pretraining, thereby producing representations closer to the pretraining distribution and resulting in consistent performance gains.

Non-backtracking walks Flock employs *non-backtracking* uniform random walk, which has an effect of faster exploration and coverage of distant regions [Alon et al., 2007]. In Table 4.16, we compare this with uniform walk that may backtrack and hence is slower in global exploration. While non-backtracking does not alter results much on inductive splits, it significantly improves performance on the transductive split. This is consistent with the idea that improving coverage especially benefits performances on large KGs, which Flock achieves via non-backtracking.

Walk lengths Flock uses random walks of length $\ell = 128$, a choice made by finding the lowest ℓ reliably visiting target node and relation on various KGs. Table 4.16 compares this with shorter and longer walks by a factor of two. As expected from coverage, shorter walks show degraded results on the transductive split with large KGs. Longer walks are overall worse, which is explained by higher learning complexity of the sequence processor that has a small hidden dimension (64) for scalability. Flock finds a balance of coverage and learnability, achieving robust results on diverse KGs.

Table 4.17: Noise injection over the best performing KGFM baseline TRIX.

(a) Zero-shot entity prediction.

	MRR	Hits@10
TRIX	0.366	0.518
+ noise	0.385	0.545
Flock	**0.391**	**0.560**

(b) Zero-shot relation prediction.

	MRR	Hits@1
TRIX	0.792	0.687
+ noise	0.739	0.643
Flock	**0.881**	**0.817**

(c) Accuracy on Petals.

	Accuracy
TRIX	50%
+ noise	52%
Flock	**100%**

Diverse starting locations of walks We recall that Flock uses a *diversification* strategy of starting locations of walks, with n walks from the query node, n walks from random relation, and n walks from random node, adding up to $3n$ walks capturing both local context near query and global information of KG. Table 4.16 compares this against all $3n$ walks starting at the query node. As expected, this causes degradations on all splits, showing the benefit of using both local and global information.

Sequence processor Flock uses a sequence processor f_θ with bidirectional GRU. In Table 4.16, we compare this against a transformer f_θ with a similar SwiGLU-RMSNorm architecture [Dubey et al., 2024] and parameter count. This alternative does not deliver good results, which is explained by the restrictions on model scales that are enforced to scale to large KGs. Flock benefits from reasoning efficiency of GRU in limited parameter regime, gaining good performance and scalability together.

Consensus protocol Flock uses softmax-weighted averaging to pool sequence processor outputs into state updates for nodes and relations, under the intuition that this can suppress uninformative proposals from the sequence processor better than simple, unweighted averaging. Table 4.16 provides a comparison, showing that weighted consensus outperforms the unweighted counterpart. This verifies our intuition on how the design of the consensus protocol strengthens Flock.

Noise injection over existing KGFMs

Since noise injection is a possible way to build a probabilistic equivariant KGFM differently from our approach [Gao et al., 2023], it is natural to ask how such KGFMs would perform compared to Flock (**Q5**). To answer this, we apply noise injection over the best performing KGFMs baselines TRIX. In each forward pass, we add element-wise noise sampled from a uniform distribution $\epsilon \sim \mathcal{U}[-0.5, 0]$ to all relation and entity embeddings after the initialization stage. Note that the addition of noise technically breaks deterministic node-relation equivariance, but the resulting model (TRIX + noise) still respects probabilistic node-relation equivariance. We then pretrain TRIX using the same experimental setup as in Table 4.13, and compare with TRIX without noise injection and Flock. To minimize the variance induced by injected noise and to ensure a fair comparison, we report ensembled prediction results with 16 samples for both TRIX + noise and Flock. This is a strong baseline implementing the ideas of prior work on noise injection and test-time ensembling for message passing on KGs [Lee et al., 2023, Gao et al., 2023].

We report the average zero-shot performance for entity prediction and relation prediction over 54 KGs in Tables 4.17a and 4.17b, respectively, as well as trained performance for Petals in Table 4.17c. Across all tasks, TRIX with naive noise injection fails to close the gap between Flock. In particular, TRIX + noise degrades compared with vanilla TRIX without noise injection in relation prediction, while boosting the performance in the entity prediction task. We hypothesize that such a difference lies in the added randomization breaks symmetry among entity embeddings more than among relation embeddings, and entity prediction depends more on having distinguishable entity representations than

Figure 4.9: PCA of relation embeddings on Metafam. ULTRA (Left) maps several inverse pairs (e.g., `fatherOf` vs. `sonOf`) to almost similar embeddings, where Flock (Right) yields clearly separated embeddings, indicating that its probabilistic equivariance allow Flock to distinguish between these semantically different relations, explaining its strong zero-shot performances.

relation prediction does. Additionally, we attribute this performance gap between Flock and TRIX + noise to the source of randomization. Flock introduces stochasticity through random walks, which induces *structure-informed* perturbations that respect the underlying topology. In contrast, TRIX with naive noise injection attempts to break deterministic node-relation equivariance by introducing structure-agonistic noise, which might, in turn, hurt the model's generalization. Together, these findings suggest that simply adding structure-agonistic noise is insufficient; performance gains only arise when stochasticity is topology-aware and is induced from the graph structure in a principled way.

Analysis of relation embedding on Metafam

To further showcase why expressivity matters for zero-shot generalization, we present a case study on the Metafam dataset [Zhou et al., 2023]. Metafam is built from a fixed family-relations ontology: during training, models observe edges with relations `motherOf`, `fatherOf`, `daughterOf`, `sonOf`, while the test queries only involve `motherOf` and `fatherOf`. Up to gender symmetries, this reduces to two effective predictive patterns (*parent_of* vs. *child_of*), and the test set focuses on a single one (*parent_of*).

In the zero-shot setting, KGFMs cannot adapt to this ontology and must rely on their pretrained relation representations. Notice that here, Metafam is challenging: many relations are structurally similar (e.g., `fatherOf`, `sonOf`, `sisterOf`, `nieceOf`) yet encode opposite predictive patterns.

Figure 4.9 shows that ULTRA's relation embeddings largely collapse these families, placing `fatherOf` and `sonOf` in almost identical positions in the PCA plane. This collapse makes it difficult to distinguish who is the parent and who is the child, leading to poor zero-shot performance. Flock, in contrast, can distinguish between these relations even if they are structurally similar, thanks to its walk sampling which introduce probabilistic equivariance on nodes and relations. As a results, Flock can produce distinct embeddings to `fatherOf` and `sonOf` and achieves much stronger zero-shot performance on Metafam.

Analysis of KG sparsity and performance

While the previous analysis explains Flock's high performance for Metafam, understanding its performances for other general KGs would be beneficial. Thus, we present an additional analysis on the 53 remaining KGs by identifying a structural property that is indicative of the performance of Flock. For

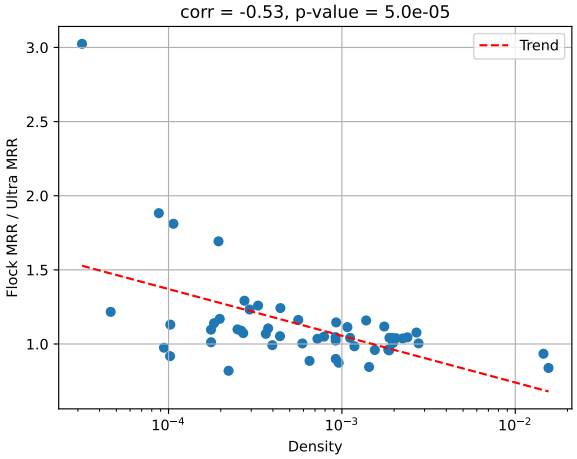


Figure 4.10: The zero-shot entity prediction performance of Flock relative to ULTRA, plotted against the densities of the 53 KGs. Performance of Flock and log-density of KGs have a Pearson correlation coefficient of -0.53 with p-value 5.0e-5, showing a statistically significant negative correlation.

the performance measure, we use the relative gain of Flock's zero-shot entity prediction MRR compared to ULTRA. For the structural property, we focus on density of KGs defined by $\frac{|E|}{|V|(|V|-1)}$ which affects the speed of random walks traversing all edges of a KG (intuitively because more edges means more time needed to traverse all of them), and hence is relevant in the context of our theory in Section 4.2.4.

Figure 4.10 shows that Flock tends to perform well on sparse KGs, while less so on dense KGs, and the tendency is statistically significant. Interestingly, this agrees with our theoretical analysis in Section 4.2.4, in which a necessary condition for universality is that the random walk covers all edges of a KG with a high probability (Proposition 15). The time taken until covering all edges is called the edge cover time, and it is known to be e.g., $O(|V||E|)$ for uniform walks [Zuckerman, 1991], which is proportional to the density of a graph. This suggests the performance of Flock is associated with the easiness to visit as many edges as possible rapidly, which is more challenging for dense KGs. This analysis is consistent with the observations of recent works on graph learning based on random walks, e.g., Wang and Cho [2024, Section 6] and Kim et al. [2025c].

4.2.6 Discussion

We introduced Flock, as a knowledge graph foundation model that respects probabilistic node-relation equivariance. Flock iteratively samples query-conditioned random walks from the input KG, records encountered nodes and relations via a recording protocol, and relies on a sequence processor and consensus protocol to obtain node and relation representations. We empirically evaluate Flock over 54 KGs across different domains for both entity and relation prediction tasks, demonstrating its superior zero-shot and finetuned performances. We further construct a synthetic dataset Petals to validate our theoretical findings. One limitation is scalability: ensuring coverage of the sampled random walk in a large KG requires an extensive number of longer walks, which can quickly become computationally infeasible. A future direction is to develop approximation strategies [Chamberlain et al., 2023, Łącki et al., 2020] that reduce the cost of random walk sampling while retaining Flock's downstream performance. Another avenue for future work is studying the families of approximable functions when the walk lengths are restricted, for example based on connections to subgraph-based reconstructions [Cotta et al., 2021].

Chapter 5. Conclusion

In this thesis, we developed architecture-agnostic methods for incorporating symmetry into neural networks, demonstrating that the generalization benefits of invariance and equivariance need not come at the cost of architectural flexibility. Across three lines of work, we showed that symmetry can be decoupled from architecture design through complementary strategies applied to the input data: tokenization, group transformation, and restructuring into alternative representations.

In Chapter 2, we established that standard transformers can serve as powerful graph and hypergraph learners through appropriate tokenization, matching or exceeding the expressivity of specialized architectures and achieving competitive performance on large-scale computational chemistry benchmarks. In Chapter 3, we introduced probabilistic symmetrization, a framework that endows arbitrary base models with equivariance through input-conditional distributions over symmetry groups, enabling transfer of pre-trained vision models to graphs, training-free warp-invariant vision, and neural PDE solving for unseen initial conditions. In Chapter 4, we provided a unified foundation for random walk neural networks, showing that stochastic graph traversals yield provable isomorphism invariance while enabling direct application of pre-trained language models to graph tasks, leading to the development of a knowledge graph foundation model capable of zero-shot link prediction on entirely unseen entities and relations.

Together, these contributions expand the scope of geometric deep learning by enabling practitioners to incorporate symmetry into any model. The methods developed here have immediate practical implications: they allow foundation models trained on large-scale data to be transferred to domains with different symmetry requirements, combining the benefits of scale with the generalization guarantees of invariance.

Several directions remain open for future investigation. A natural extension is to broaden the class of transformations beyond symmetry groups. Many domains exhibit structured transformations that preserve semantic content but do not form groups. Program rewriting and natural language paraphrasing are prominent examples. These transformations may lack inverses or fail to compose associatively, precluding direct application of group-theoretic tools. Extending architecture-agnostic invariance methods to such transformations could yield new insights into the generalization behavior of code and language models. A complementary direction is symmetry discovery: settings where the relevant symmetry is not known a priori but must be inferred from data. The general formulations developed in this thesis, which accommodate diverse symmetry groups, may provide a basis for searching the space of symmetries itself. Together, these extensions point toward a broader understanding of how structure (whether known or latent, group-theoretic or otherwise) can be harnessed for improved generalization of neural networks.

Bibliography

Ralph Abboud, İsmail İlkan Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In *NeurIPS*, 2020.

Ralph Abboud, İsmail İlkan Ceylan, Martin Grohe, and Thomas Lukasiewicz. The surprising power of graph neural networks with random node initialization. In *IJCAI*, 2021.

Mohammed Amin Abdullah, Colin Cooper, and Moez Draief. Speeding up cover time of sparse graphs using local knowledge. In *IWOCA*, 2015.

Tara Akhound-Sadegh, Laurence Perreault-Levasseur, Johannes Brandstetter, Max Welling, and Siamak Ravanbakhsh. Lie point symmetry and physics-informed networks. *Advances in Neural Information Processing Systems*, 36:42468–42481, 2023.

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Lemos, Cheng-Hao Liu, Marcin Sendera, Siamak Ravanbakhsh, Gauthier Gidel, Yoshua Bengio, et al. Iterated denoising energy matching for sampling from boltzmann densities. *arXiv preprint arXiv:2402.06121*, 2024.

Marjan Albooyeh, Daniele Bertolini, and Siamak Ravanbakhsh. Incidence networks for geometric deep learning. *arXiv*, 2019.

Romas Aleliunas, Richard M. Karp, Richard J. Lipton, László Lovász, and Charles Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Annual Symposium on Foundations of Computer Science*, 1979.

James Allingham, Bruno Mlodozeniec, Shreyas Padhy, Javier Antorán, David Krueger, Richard Turner, Eric Nalisnick, and José Miguel Hernández-Lobato. A generative model of symmetry transformations. *Advances in Neural Information Processing Systems*, 37:91091–91130, 2024.

Noga Alon, Itai Benjamini, Eyal Lubetzky, and Sasha Sodin. Non-backtracking random walks mix faster. *Communications in Contemporary Mathematics*, 2007.

Nurudin Alvarez-Gonzalez, Andreas Kaltenbrunner, and Vicenç Gómez. Improving subgraph-GNNs via edge-level ego-network encodings. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856.

Jean-Marc Andreoli. Convolution, attention and structure embedding. *arXiv*, 2019.

Devanshu Arya, Deepak K. Gupta, Stevan Rudinac, and Marcel Worring. Hypersage: Generalizing inductive representation learning on hypergraphs. *arXiv*, 2020.

Song Bai, Feihu Zhang, and Philip H. S. Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognit.*, 2021.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*, 2019.

Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *ICML*, 2021.

Jacob Bamberger, Federico Barbero, Xiaowen Dong, and Michael Bronstein. Bundle neural networks for message diffusion on graphs. *arXiv*, 2024.

Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise. *Advances in Neural Information Processing Systems*, 36:41259–41282, 2023.

Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *ICLR*, 2020.

Pablo Barceló, Mikhail Galkin, Christopher Morris, and Miguel Romero. Weisfeiler and leman go relational. In *LoG*, 2022.

Sourya Basu, Prasanna Sattigeri, Karthikeyan Natesan Ramamurthy, Vijil Chenthamarakshan, Kush R. Varshney, Lav R. Varshney, and Payel Das. Equi-tuning: Group equivariant fine-tuning of pretrained models. *arXiv*, 2022.

Sourya Basu, Prasanna Sattigeri, Karthikeyan Natesan Ramamurthy, Vijil Chenthamarakshan, Kush R. Varshney, Lav R. Varshney, and Payel Das. Equi-tuning: Group equivariant fine-tuning of pretrained models. In *AAAI*, 2023.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018.

M Faisal Beg, Michael I Miller, Alain Trouvé, and Laurent Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International journal of computer vision*, 61(2):139–157, 2005.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 2003.

Serge J. Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

Anna Ben-Hamou, Roberto I. Oliveira, and Yuval Peres. Estimating graph parameters via random walks with restarts. In *Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, abs/1308.3432, 2013.

Itai Benjamini, Gady Kozma, Laszlo Lovasz, Dan Romik, and Gabor Tardos. Waiting for a bat to fly by (in polynomial time). *Combinatorics, Probability and Computing*, 2006.

Suman K Bera and C Seshadhri. How to count triangles, without seeing the whole graph. In *KDD*, 2020.

Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics.*, 2010.

C. Berge and E. Minieka. *Graphs and Hypergraphs.* North-Holland Publishing Company, 1981.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR*, 2022a.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR*, 2022b.

Satwik Bhattamishra, Arkil Patel, and Navin Goyal. On the computational power of transformers and its implications in sequence modeling. In *CoNLL*, 2020.

Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *ICML*, 2020.

Mitchell Black, Zhengchao Wan, Amir Nayyeri, and Yusu Wang. Understanding oversquashing in gnns through the lens of effective resistance. In *ICML*, 2023.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020.

Alexander Bogatskiy, Sanmay Ganguly, Thomas Kipf, Risi Kondor, David W Miller, Daniel Murnane, Jan T Offermann, Mariel Pettee, Phiala Shanahan, Chase Shimmin, et al. Symmetry group equivariant architectures for physics. *arXiv preprint arXiv:2203.06153*, 2022.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, and Bruno Lévy. Geometric modeling based on polygonal meshes. In *Eurographics Tutorials*, 2008.

L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.

Johannes Brandstetter, Max Welling, and Daniel E Worrall. Lie point symmetry data augmentation for neural pde solvers. In *International Conference on Machine Learning*, pages 2241–2256. PMLR, 2022.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv*, 2019.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*, 2017.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Velickovic. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv*, 2021.

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv*, 2024.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.

Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: Combining social media information and music content. In *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, 2010.

D. Burago, I.U.D. Burago, and S. Ivanov. *A Course in Metric Geometry*. American Mathematical Society, 2001.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. *arXiv*, 2020.

Timoteo Carletti, Federico Battiston, Giulia Cencetti, and Duccio Fanelli. Random walks on hypergraphs. *Physical Review E*, 2020.

carnage. Expected hitting time for simple random walk from origin to point (x,y) in 2d-integer-grid. MathOverflow, 2012. URL https://mathoverflow.net/questions/112470. [Online:] https://mathoverflow.net/questions/112470.

Elena Celledoni, Matthias J Ehrhardt, Christian Etmann, Brynjulf Owren, Carola-Bibiane Schönlieb, and Ferdia Sherry. Equivariant neural networks for inverse problems. *Inverse Problems*, 37(8):085006, 2021.

Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Yannick Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. In *ICLR*, 2023.

I. Chavel. *Riemannian Geometry: A Modern Introduction*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.

Dexiong Chen, Till Hendrik Schulz, and Karsten Borgwardt. Learning long range dependencies on graphs via random walks. In *ICLR*, 2025.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020a.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: Large language and graph assistant. *arXiv*, 2024a.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *NeurIPS*, 2019.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *NeurIPS*, 2020b.

Zhikai Chen, Haitao Mao, Jingzhe Liu, Yu Song, Bingheng Li, Wei Jin, Bahare Fatemi, Anton Tsitsulin, Bryan Perozzi, Hui Liu, and Jiliang Tang. Text-space graph foundation models: Comprehensive benchmarks and new insights. *arXiv*, 2024b.

Maixent Chenebaux. graph-walker: Fastest random walks generator on networkx graphs. https://github.com/kerighan/graph-walker, 2020.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *ICLR*, 2022.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *ICCV*, 2013.

Sungjun Cho, Dae-Woong Jeong, Sung Moon Ko, Jinwoo Kim, Sehui Han, Seunghoon Hong, Honglak Lee, and Moontae Lee. 3d denoisers are good 2d teachers: molecular pretraining via denoising and cross-modal distillation. In *AAAI*, 2025.

Krzysztof Marcin Choromanski, Mark Rowland, and Adrian Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *NeurIPS*, 2017.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

Taco S. Cohen and Max Welling. Steerable cnns. In *ICLR*, 2017.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020.

Rob Cornish. Stochastic neural network symmetrisation in markov categories. *arXiv preprint arXiv:2406.11814*, 2024.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In *NeurIPS*, 2021.

Leonardo Cotta, Gal Yehuda, Assaf Schuster, and Chris J. Maddison. Probabilistic invariant learning with randomized linear classifiers. In *NeurIPS*, 2023.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. In *NeurIPS*, 2024.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 1989.

Zihang Dai, Hanxiao Liu, Quoc V. Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*, 2021.

Anirban Dasgupta, Ravi Kumar, and Tamas Sarlos. On estimating the average degree. In *WWW*, 2014.

Roee David and Uriel Feige. Random walks with the minimum degree local rule have $o(n^2)$ cover time. *SIAM J. Comput.*, 2018.

Valentin De Bortoli, Emile Mathieu, Michael Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. *Advances in Neural Information Processing Systems*, 35:2406–2422, 2022.

Valentin De Bortoli, Michael Hutchinson, Peter Wirnsberger, and Arnaud Doucet. Target score matching. *arXiv preprint arXiv:2402.08667*, 2024.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

Boris Nikolaevich Delaunay. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles. 6: 793–800.*, 1934.

Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulenard, Andrea Tagliasacchi, and Leonidas J. Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *ICCV*, 2021.

Harm Derksen and Gregor Kemper. *Computational Invariant Theory*. Springer, 2015.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *AAAI*, 2018.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In *NeurIPS*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. Be more with less: Hypergraph attention networks for inductive text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, 2020.

Yihe Dong, Will Sawin, and Yoshua Bengio. HNHN: hypergraph networks with hyperedge neurons. *arXiv*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, and et al. The llama 3 herd of models. *arXiv*, abs/2407.21783, 2024.

Ioana Dumitriu, Prasad Tetali, and Peter Winkler. On playing golf with two balls. *SIAM J. Discret. Math.*, 2003.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv*, 2020.

Vijay Prakash Dwivedi, Chaitanya K. Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv*, 2020.

Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *NeurIPS*, 2022.

Nadav Dym and Steven J. Gortler. Low dimensional invariant embeddings for universal geometric learning. *arXiv*, 2022.

Nadav Dym, Hannah Lawrence, and Jonathan W Siegel. Equivariant frames and the impossibility of continuous canonicalization. *arXiv preprint arXiv:2402.16077*, 2024.

Bryn Elesedy. Group symmetry in pac learning. In *ICLR workshop on geometrical and topological representation learning*, 2022.

Bryn Elesedy. *Symmetry and Generalisation in Machine Learning*. PhD thesis, University of Oxford, 2023.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010.

William Falcon. Pytorch lightning. https://github.com/Lightning-AI/lightning, 2019.

Abdul Fatir. Metrics for evaluating gans (pytorch). https://github.com/abdulfatir/gan-metrics-pytorch, 2018.

Uriel Feige. A tight upper bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 1995.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *AAAI*, 2019.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Ben Finkelshtein, Xingyue Huang, Michael M Bronstein, and Ismail Ilkan Ceylan. Cooperative graph neural networks. In *ICML*, 2024.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *ICML*, 2020.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. In *ICML*, 2021.

Robert Fitzner and Remco van der Hofstad. Non-backtracking random walk. *Journal of Statistical Physics*, 2013.

G.B. Folland. *A Course in Abstract Harmonic Analysis*. Studies in Advanced Mathematics. Taylor & Francis, 1994.

Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *NIPS*, 2017.

Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *NeurIPS*, 2020.

Mikhail Galkin, Xinyu Yuan, Hesham Mostafa, Jian Tang, and Zhaocheng Zhu. Towards foundation models for knowledge graph reasoning. In *ICLR*, 2024.

Jianfei Gao, Yangze Zhou, Jincheng Zhou, and Bruno Ribeiro. Double equivariance for inductive link prediction for both new nodes and new relation types. In *arXiv*, 2023.

Yue Gao, Meng Wang, Dacheng Tao, Rongrong Ji, and Qionghai Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Trans. Image Process.*, 2012.

Yuxia Geng, Jiaoyan Chen, Jeff Z. Pan, Mingyang Chen, Song Jiang, Wen Zhang, and Huajun Chen. Relational message passing for fully inductive knowledge graph completion. In *ICDE*, 2023.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

Jhony H. Giraldo, Konstantinos Skianis, Thierry Bouwmans, and Fragkiskos D. Malliaros. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *CIKM*, 2023.

Leo J. Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2006.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. *arXiv preprint arXiv:1912.03263*, 2019.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. In *ICLR*, 2019.

Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. Implicit graph neural networks. In *NeurIPS*, 2020.

Shi Gu, Muzhi Yang, John D Medaglia, Ruben C Gur, Raquel E Gur, Theodore D Satterthwaite, and Danielle S Bassett. Functional hypergraph uncovers novel covariant structures over neurodevelopment. *Human Brain Mapp.*, 2017.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *ICLR*, 2017.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv*, 2017.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 2002.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 2016.

Jason S. Hartford, Devon R. Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *ICML*, 2018.

Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *arXiv*, 2021a.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *ICLR*, 2021b.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. In *ICLR*, 2023.

Jeffery Hein. Wald's identity.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. *arXiv preprint arXiv:2410.20587*, 2024.

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *ICLR*, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020b.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. *arXiv*, 2021.

Chin-Wei Huang, Milad Aghajohari, Joey Bose, Prakash Panangaden, and Aaron C Courville. Riemannian diffusion models. *Advances in Neural Information Processing Systems*, 35:2750–2761, 2022.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In *IJCAI*, 2021.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv*, 2020.

Xingyue Huang, Miguel Romero Orth, İsmail İlkan Ceylan, and Pablo Barceló. A theory of link prediction via relational weisfeiler-leman on knowledge graphs. In *NeurIPS*, 2023.

Xingyue Huang, Pablo Barcelo, Michael M. Bronstein, İsmail İlkan Ceylan, Mikhail Galkin, Juan L Reutter, and Miguel Romero Orth. How expressive are knowledge graph foundation models? In *ICML*, 2025.

Md. Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv*, 2021.

Katsuhiko Ishiguro, Shin ichi Maeda, and Masanori Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. *arXiv*, 2019.

Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In *ICML*, 2018.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. *arXiv*, 2021a.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021b.

Svante Janson and Yuval Peres. Hitting times for random walks with restarts. *SIAM J. Discret. Math.*, 2012.

Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the expressive power of geometric graph neural networks. *arXiv*, 2023.

Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning*, pages 15546–15566. PMLR, 2023.

Zahra Kadkhodaie and Eero P Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *arXiv preprint arXiv:2007.13640*, 2020.

Jari P Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*. Springer, 2005.

Gil Kalai. Linear programming, the simplex algorithm and simple polytopes. *Math. Program.*, 1997.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. In *NeurIPS*, 2019.

Piotr Kicki, Piotr Skrzypczynski, and Mete Ozay. A new approach to design symmetry invariant neural networks. In *IJCNN*, 2021.

Eun-Sol Kim, Woo-Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. Hypergraph attention networks for multimodal learning. In *CVPR*, 2020a.

Ildoo Kim, Younghoon Kim, and Sungwoong Kim. Learning loss for test-time augmentation. In *Advances in Neural Information Processing Systems*, 2020b.

Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. Transformers generalize deepsets and can be extended to graphs and hypergraphs. In *NeurIPS*, 2021a.

Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15059–15068, 2021b.

Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *NeurIPS*, 2022a.

Jinwoo Kim, Saeyoon Oh, Sungjun Cho, and Seunghoon Hong. Equivariant hypergraph neural networks. In *ECCV*, 2022b.

Jinwoo Kim, Tien Dat Nguyen, Ayhan Suleymanzade, Hyeokjun An, and Seunghoon Hong. Learning probabilistic symmetrization for architecture agnostic equivariance. In *NeurIPS*, 2023.

Jinwoo Kim, Max Beier, Petar Bevanda, Nayun Kim, and Seunghoon Hong. Sequence modeling with spectral mean flows. *arXiv preprint arXiv:2510.15366*, 2025a.

Jinwoo Kim, Xingyue Huang, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, and İsmail İlkan Ceylan. Flock: A knowledge graph foundation model via learning on random walks. *arXiv preprint arXiv:2510.01510*, 2025b.

Jinwoo Kim, Olga Zaghen, Ayhan Suleymanzade, Youngmin Ryou, and Seunghoon Hong. Revisiting random walks for learning on graphs. In *ICLR*, 2025c.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *ICML*, 2018.

Alexander A Kirillov. *An introduction to Lie groups and Lie algebras*, volume 113. Cambridge University Press, 2008.

Florian Klimm, Charlotte M Deane, Gesine Reinert, and Ernesto Estrada. Hypergraphs for predicting essential genes using multiprotein complex data. *Journal of Complex Networks*, 2021.

Filip Klubicka, Alfredo Maldonado, Abhijit Mahalunkar, and John D. Kelleher. Synthetic, yet natural: Properties of wordnet random walk corpora and the impact of rare words on embedding performance. In *GWC*, 2019.

Filip Klubicka, Alfredo Maldonado, Abhijit Mahalunkar, and John D. Kelleher. English wordnet random walk pseudo-corpora. In *LREC*, 2020.

Boris Knyazev, Graham W. Taylor, and Mohamed R. Amer. Understanding attention and generalization in graph neural networks. In *NeurIPS*, 2019.

Marin Kobilarov. Discrete optimal control on lie groups and applications to robotic vehicles. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 5523–5529. IEEE, 2014a.

Marin Kobilarov. Solvability of geometric integrators for multi-body systems. In *Multibody Dynamics: Computational Methods and Applications*, pages 145–174. Springer, 2014b.

Eleftherios Kofidis and Phillip A. Regalia. On the best rank-1 approximation of higher-order supersymmetric tensors. *Siam J. Matrix Anal. Appl*, 2002.

Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: Sampling configurations for multi-body systems with symmetric energies. *arXiv*, 2019.

Zico Kolter, Shaojie Bai, Devin Wilmott, Mordechai Kornbluth, and Jonathan Mailoa. First place solution of 2019 champs predicting molecular properties challenge. https://www.kaggle.com/c/champs-scalar-coupling/discussion/106575, 2019.

Lingkai Kong and Molei Tao. Convergence of kinetic langevin monte carlo on lie groups. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 3011–3063. PMLR, 2024.

Masanori Koyama, Kenji Fukumizu, Kohei Hayashi, and Takeru Miyato. Neural fourier transform: A general approach to equivariant representation learning. *arXiv preprint arXiv:2305.18484*, 2023.

Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *NeurIPS*, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Hugo Larochelle, Dumitru Erhan, Aaron C. Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007.

Hannah Lawrence, Vasco Portilheiro, Yan Zhang, and Sékou-Oumar Kaba. Improving equivariant networks with probabilistic symmetry breaking. In *The Thirteenth International Conference on Learning Representations*, 2025.

J. Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York, 2012.

Jaejun Lee, Chanyoung Chung, and Joyce Jiyoung Whang. Ingram: Inductive knowledge graph embedding via relation graphs. In *ICML*, 2023.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

Taeyoung Lee and Gregory S. Chirikjian. Geometric interpretation of brownian motion on riemannian manifolds. *arXiv preprint arXiv:2510.19991*, 2025.

Mario Lezcano Casado. Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32, 2019.

Dong Li, Zhiming Xu, Sheng Li, and Xin Sun. Link prediction in social networks based on hypergraph. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, 2013.

Junying Li, Deng Cai, and Xiaofei He. Learning graph-level representation for drug discovery. *arXiv*, 2017.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Valerii Likhosherstov, Krzysztof Choromanski, and Adrian Weller. On the expressive power of self-attention matrices. *arXiv*, 2021.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess E. Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv*, 2022.

Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *ICCV*, 2021.

Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *NeurIPS*, 2020.

Marco Lorenzi and Xavier Pennec. Geodesics, parallel transport & one-parameter subgroups for diffeomorphic image registration. *International journal of computer vision*, 105(2):111–127, 2013.

Steph-Yves M. Louis, Yong Zhao, Alireza Nasiri, Xiran Wong, Yuqi Song, Fei Liu, and Jianjun Hu. Global attention based graph convolutional neural networks for improved materials property prediction. *arXiv*, 2020.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *ICLR*, 2020.

David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. In *AAAI*, 2022.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks. *arXiv*, 2020.

Lachlan Ewen MacDonald, Sameera Ramasinghe, and Simon Lucey. Enabling equivariance for arbitrary lie groups. *arXiv preprint arXiv:2111.08251*, 2022.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, 2019a.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *ICLR*, 2019b.

Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *ICML*, 2019c.

Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. In *ICML*, 2020.

Karolis Martinkus, Pál András Papp, Benedikt Schesch, and Roger Wattenhofer. Agent-based graph neural networks. In *ICLR*, 2023.

Brendan D. McKay, Mehmet Aziz Yirik, and Christoph Steinbeck. Surge: a fast open-source chemical graph generator. *J. Cheminformatics*, 2022.

Nathan McNew. Random walks with restarts, 3 examples, 2013.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *ICLR*, 2018.

Francesco Mezzadri. How to generate random matrices from the classical compact groups. *Notices of the American Mathematical Society*, 2006.

Silvio Micali and Zeyuan Allen Zhu. Reconstructing markov processes from independent and anonymous experiments. *Discret. Appl. Math.*, 2016.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. In *NeurIPS*, 2020.

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv*, 2022.

Thomas Mitchel, Michael J Taylor, and Vincent Sitzmann. Neural isometries: Taming transformations for equivariant ml. *Advances in Neural Information Processing Systems*, 37:7311–7338, 2024.

Arnab Kumar Mondal, Siba Smarak Panigrahi, Oumar Kaba, Sai Rajeswar Mudumba, and Siamak Ravanbakhsh. Equivariant adaptation of large pretrained models. *Advances in Neural Information Processing Systems*, 36:50293–50309, 2023.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019a.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019b.

S. Chandra Mouli and Bruno Ribeiro. Neural networks for learning counterfactual g-invariances from single environments. In *ICLR*, 2021.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *ICLR*, 2019a.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *ICML*, 2019b.

Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks. In *WWW*, 2022.

Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley J. Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *ICML*, 2023a.

Tien Dat Nguyen, Jinwoo Kim, Hongseok Yang, and Seunghoon Hong. Learning symmetrization for equivariance with orbit distance minimization. *arXiv*, 2023b.

Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. In *NeurIPS*, 2020.

Netta Ollikka, Amro Abbas, Andrea Perin, Markku Kilpeläinen, and Stéphane Deny. A comparison between humans and ai at recognizing objects in unusual poses. *arXiv preprint arXiv:2402.03973*, 2024.

Peter J Olver. *Applications of Lie groups to differential equations*, volume 107. Springer Science & Business Media, 1993.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020.

Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, et al. The pagerank citation ranking: Bringing order to the web, 1999.

Horace Pan and Risi Kondor. Permutation equivariant layers for higher order interactions. In *AISTATS*, 2022.

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. In *NeurIPS*, 2021.

Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, and Seung won Hwang. Grpe: Relative positional encoding for graph transformer. *arXiv*, 2022.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, 2014.

Oleg Platonov, Denis Kuznedelev, Artem Babenko, and Liudmila Prokhorenkova. Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond. In *NeurIPS*, 2023a.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv*, 2023b.

Sebastian Prillo and Julian Martin Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *ICML*, 2020.

Omri Puny, Heli Ben-Hamu, and Yaron Lipman. Global attention improves graph networks generalization. *arXiv*, 2020.

Omri Puny, Matan Atzmon, Edward J. Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *ICLR*, 2022.

Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *KDD*, 2018.

Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*, 2022.

Brian Rappaport, Anuththari Gamage, and Shuchin Aeron. Faster clustering via non-backtracking random walks. *arXiv*, 2017.

Siamak Ravanbakhsh, Jeff G. Schneider, and Barnabás Póczos. Equivariance through parameter-sharing. In *ICML*, 2017.

Lawrence A. Ray. 2-d and 3-d image registration for medical, remote sensing, and industrial applications. *J. Electronic Imaging*, 2005.

Scott Reed, Konrad Żolna, Emilio Parisotto, Sergio G´ ǁmez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv*, 2022.

Lorenz Richter and Julius Berner. Improved sampling via learned diffusions. *arXiv preprint arXiv:2307.01198*, 2023.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations, 1996.

Michal Rolinek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vit Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. In *ECCV*, 2020.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, 2020.

Danielle Rothermel, Margaret Li, Tim Rocktäschel, and Jakob Foerster. Don't sweep your learning rate under the rug: A closer look at cross-modal transfer of pretrained transformers. *arXiv*, 2021.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *EMNLP*, 2020.

Akiyoshi Sannai, Makoto Kawano, and Wataru Kumagai. Equivariant and invariant reynolds networks. *arXiv*, 2021.

Ryoma Sato. Training-free graph neural networks and the power of labels as features. *arXiv*, 2024.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *ICML*, 2021.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *ESWC*, 2018.

Johann Schmidt and Sebastian Stober. Tilt your head: Activating the hidden spatial-invariance of classifiers. *arXiv preprint arXiv:2405.03730*, 2024.

Hadar Serviansky, Nimrod Segol, Jonathan Shlomi, Kyle Cranmer, Eilam Gross, Haggai Maron, and Yaron Lipman. Set2graph: Learning graphs from sets. In *NeurIPS*, 2020.

Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. When and why test-time augmentation works. *arXiv preprint arXiv:2011.11156*, 2020.

Jun Shao. *Mathematical Statistics*. Springer, 2003.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Junhong Shen, Liam Li, Lucio M. Dery, Corey Staten, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. Cross-modal fine-tuning: Align then refine. *arXiv*, 2023.

Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016.

Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arXiv*, 2022.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. *arXiv*, 2023.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6 (1):1–48, 2019.

Zakhar Shumaylov, Peter Zaika, James Rowbottom, Ferdia Sherry, Melanie Weber, and Carola-Bibiane Schönlieb. Lie algebra canonicalization: Equivariant neural operators under arbitrary lie groups. *arXiv preprint arXiv:2410.02698*, 2024.

Yuval Sieradzki, Nitzan Hodos, Gal Yehuda, and Assaf Schuster. Coin flipping neural networks. In *ICML*, 2022.

Amit Singer. Angular synchronization by eigenvectors and semidefinite programming. *Applied and computational harmonic analysis*, 30(1):20–36, 2011.

Utkarsh Singhal, Ryan Feng, Stella X Yu, and Atul Prakash. Test-time canonicalization by foundation models for robust perception. *arXiv preprint arXiv:2507.10375*, 2025.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

Balasubramaniam Srinivasan and Bruno Ribeiro. On the equivalence between positional node embeddings and structural graph representations. In *ICLR*, 2020.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv*, 2021.

Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

Behrooz Tahmasebi, Derek Lim, and Stefanie Jegelka. The power of recursion in graph neural networks for counting substructures. In *AISTATS*, 2023.

Shulong Tan, Jiajun Bu, Chun Chen, and Xiaofei He. Using rich social media information for music recommendation via hypergraph model. In *Social Media Modeling and Computing*. Springer, 2011.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: A uniform language model fine-tuning framework for attributed graph embedding. In *NeurIPS*, 2023a.

Yanchao Tan, Zihao Zhou, Hang Lv, Weiming Liu, and Carl Yang. Walklm: a uniform language model fine-tuning framework for attributed graph embedding. In *NeurIPS*, 2023b.

Molei Tao and Tomoki Ohsawa. Variational optimization on lie groups, with examples of leading (generalized) eigenvalue problems. In *International Conference on Artificial Intelligence and Statistics*, pages 4269–4280. PMLR, 2020.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv*, 2020.

Komal K. Teru, Etienne G. Denis, and William L. Hamilton. Inductive relation prediction by subgraph reasoning. In *ICML*, 2020.

Erik H. Thiede, Wenda Zhou, and Risi Kondor. Autobahn: Automorphism-based graph neural nets. In *NeurIPS*, 2021.

Nathaniel Thomas, Tess E. Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv*, 2018.

Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *Transactions on Machine Learning Research*, 2023.

Jan Tönshoff, Martin Ritzert, Eran Rosenbluth, and Martin Grohe. Where did the gap go? reassessing the long-range graph benchmark. *Trans. Mach. Learn. Res.*, 2024, 2024.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Workshop on Continuous Vector Space Models and their Compositionality*, 2015.

L.W. Tu. *An Introduction to Manifolds*. Universitext. Springer New York, 2010.

Elise van der Pol, Daniel E. Worrall, Herke van Hoof, Frans A. Oliehoek, and Max Welling. MDP homomorphic networks: Group symmetries in reinforcement learning. In *NeurIPS*, 2020.

Francisco Vargas, Will Grathwohl, and Arnaud Doucet. Denoising diffusion samplers. *arXiv preprint arXiv:2302.13834*, 2023.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *ICLR*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Singanallur V Venkatakrishnan, Charles A Bouman, and Brendt Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE global conference on signal and information processing*, pages 945–948. IEEE, 2013.

Soledad Villar, David W. Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *NeurIPS*, 2021.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019a.

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1, 000 layers. *arXiv*, 2022.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, et al. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv*, 2019b.

Runzhong Wang, Junchi Yan, and Xiaokang Yang. Neural graph matching network: Learning lawler's quadratic assignment problem with extension to hypergraph and multiple-graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. Self-attention with structural position representations. In *EMNLP-IJCNLP*, 2019c.

Yuanqing Wang and Kyunghyun Cho. Non-convolutional graph neural networks. *NeurIPS*, 2024.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Hermann Weyl. *The Classical Groups: Their Invariants and Representations*. Princeton University Press, 1946.

Noam Wies, Yoav Levine, Daniel Jannai, and Amnon Shashua. Which transformer architecture fits my data? A vocabulary bottleneck in self-attention. In *ICML*, 2021.

Robin Winter, Frank Noé, and Djork-Arné Clevert. Permutation-invariant variational autoencoder for graph-level representation learning. In *NeurIPS*, 2021.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *arXiv*, 2019.

Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *arXiv*, 2021a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 2021b.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *NeurIPS*, 2019.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. In *2022 IEEE International Conference on Data Mining (ICDM)*, 2022.

Jianke Yang, Nima Dehmamy, Robin Walters, and Rose Yu. Latent space symmetry discovery. *arXiv preprint arXiv:2310.00105*, 2023.

Dmitry Yarotsky. Universal approximations of invariant maps by neural networks. *Constructive Approximation*, 55(1): 407–474, 2022.

Mohsen Yavartanoo, Shih-Hsuan Hung, Reyhaneh Neshatavar, Yue Zhang, and Kyoung Mu Lee. Polynet: Polynomial neural network for 3d shape recognition with polyshape representation. In *3DV*, 2021.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *NeurIPS*, 2021.

Felix X. Yu, Ananda Theertha Suresh, Krzysztof Marcin Choromanski, Daniel N. Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. In *NeurIPS*, 2016.

Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. In *ICCV*, 2021.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *ICLR*, 2020.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *NeurIPS*, 2017.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. In *ICML*, 2023a.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. *arXiv*, 2023b.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018a.

Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. Labeling trick: A theory of using graph neural networks for multi-node representation learning. In *NeurIPS*, 2021.

Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-sagnn: a self-attention based graph neural network for hypergraphs. In *ICLR*, 2020.

Yongqi Zhang and Quanming Yao. Knowledge graph reasoning with relational digraph. In *WebConf*, 2022.

Yongqi Zhang, Zhanke Zhou, Quanming Yao, Xiaowen Chu, and Bo Han. Adaprop: Learning adaptive propagation for graph neural network based knowledge graph reasoning. In *KDD*, 2023c.

Yucheng Zhang, Beatrice Bevilacqua, Mikhail Galkin, and Bruno Ribeiro. TRIX: A more expressive model for zero-shot domain transfer in knowledge graphs. In *LoG*, 2024.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv*, 2018b.

Jianan Zhao, Meng Qu, Chaozhuo Li, Hao Yan, Qian Liu, Rui Li, Xing Xie, and Jian Tang. Learning on large-scale text-attributed graphs via variational inference. *arXiv*, 2022a.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael M. Bronstein, Zhaocheng Zhu, and Jian Tang. Graphtext: Graph reasoning in text space. *arXiv*, 2023.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. *arXiv*, 2019.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *ICLR*, 2022b.

Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI*, 2010.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv*, 2018.

Jincheng Zhou, Beatrice Bevilacqua, and Bruno Ribeiro. A multi-task perspective for link prediction with new relation types and nodes. In *NeurIPS GLFrontiers*, 2023.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*, 2002.

Xiaojin Jerry Zhu. Semi-supervised learning literature survey, 2005.

Yuchen Zhu, Tianrong Chen, Lingkai Kong, Evangelos A. Theodorou, and Molei Tao. Trivialized momentum facilitates diffusion generative modeling on lie groups. In *International Conference on Learning Representations*, 2025.

Yun Zhu, Yaoke Wang, Haizhou Shi, and Siliang Tang. Efficient tuning and inference for large language models on textual graphs. In *IJCAI*, 2024.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS*, 2021.

Zhaocheng Zhu, Xinyu Yuan, Mikhail Galkin, Sophie Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. A*net: A scalable path-based reasoning approach for knowledge graphs. In *NeurIPS*, 2023.

Barbara Zitova and Jan Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

Markus Zopf. 1-wl expressiveness is (almost) all you need. *arXiv*, 2022.

David Zuckerman. On the time to traverse all edges of a graph. *Inf. Process. Lett.*, 1991.

Jakub Łącki, Slobodan Mitrović, Krzysztof Onak, and Piotr Sankowski. Walking randomly, massively, and efficiently. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 364–377, 2020.

Chapter A. Appendix

A.1 Appendix of higher-order transformers

A.1.1 Proofs

Proof of Proposition 1

We start with the following lemmas:

Lemma 4. *Let μ an equivalence class of order-$(k+l)$ multi-indices. Then, the set of all $\mathbf{i} \in [n]^k$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ for some $\mathbf{j} \in [n]^l$ is an equivalence class of order-k multi-indices. Likewise, the set of all $\mathbf{j}$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ for some $\mathbf{i}$ is an equivalence class of order-l multi-indices.*

Proof. We only prove for $\mathbf{i}$, as proof for $\mathbf{j}$ is analogous. For some $(\mathbf{i}_1, \mathbf{j}_1) \in \mu$, let us denote $\mathbf{i}_1$'s equivalence class as μ_k. It is sufficient that we prove $\mathbf{i} \in \mu_k \Leftrightarrow (\mathbf{i}, \mathbf{j}) \in \mu$ for some $\mathbf{j}$.

($\Rightarrow$) For all $\mathbf{i} \in \mu_k$, as $\mathbf{i}_1 \sim \mathbf{i}$ we have $\mathbf{i} = \pi(\mathbf{i}_1)$ for some $\pi \in S_n$. As π acts on multi-indices entry-wise, we have $\pi(\mathbf{i}_1, \mathbf{j}_1) = (\mathbf{i}, \pi(\mathbf{j}_1))$. As the equivalence pattern is invariant to node permutation by definition, we have $\pi(\mathbf{i}_1, \mathbf{j}_1) \sim (\mathbf{i}, \pi(\mathbf{j}_1)) \sim (\mathbf{i}_1, \mathbf{j}_1)$, and thus $(\mathbf{i}, \pi(\mathbf{j}_1)) \in \mu$. Therefore, for all $\mathbf{i} \in \mu_k$, we always have $(\mathbf{i}, \mathbf{j}) \in \mu$ when we set $\mathbf{j} = \pi(\mathbf{j}_1)$.

($\Leftarrow$) For all $(\mathbf{i}, \mathbf{j}) \in \mu$, as $(\mathbf{i}, \mathbf{j}) \sim (\mathbf{i}_1, \mathbf{j}_1)$ we have $(\mathbf{i}, \mathbf{j}) = \pi(\mathbf{i}_1, \mathbf{j}_1)$ for some $\pi \in S_n$. We have equivalently $\mathbf{i} = \pi(\mathbf{i}_1)$ and $\mathbf{j} = \pi(\mathbf{j}_1)$ for the π, which leads to $\mathbf{i} \sim \mathbf{i}_1$ and therefore $\mathbf{i} \in \mu_k$. □

Lemma 5. *Let μ an equivalence class of order-k multi-indices. Then, every $\mathbf{i} \in \mu$ contains the same number of unique elements, which is equal to $|\mu|$ i.e., the number of nonempty subsets in μ's partition.*

Proof. All $\mathbf{i} \in \mu$ have the same equality pattern, specified by μ's representative partition. Specifically, for all $\mathbf{i} \in \mu$, $\mathbf{i}_a = \mathbf{i}_b$ holds iff $\mathbf{i}_a$ and $\mathbf{i}_b$ belong to the same subset within μ's partition. Therefore, each nonempty subset within μ's partition specifies exactly one value within $\mathbf{i}$, and any $\mathbf{i}_a, \mathbf{i}_b$ s.t. $\mathbf{i}_a \neq \mathbf{i}_b$ are contained in distinct subsets within μ's partition. Thus, each subset in μ's partition specifies one unique element in $\mathbf{i}$, and we have the number of unique elements in $\mathbf{i}$ equal to $|\mu|$ for all $\mathbf{i} \in \mu$. □

Now, we prove Proposition 1.

Proof. From Lemma 4, let us denote the set of all $\mathbf{i} \in [n]^k$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ as an order-k equivalence class μ_k, and denote the set of all $\mathbf{j}$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ as an order-l equivalence class μ_q.

Then, in Equation (2.7), to compute $\boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}}$ $\forall (\mathbf{i}, \mathbf{j}) \in \mu$ it is sufficient that we have $\mathbf{K}^{\mu}_{\mathbf{i}}$ $\forall \mathbf{i} \in \mu_k$ and $\mathbf{Q}^{\mu}_{\mathbf{j}}$ $\forall \mathbf{j} \in \mu_q$. Based on the fact, we now analyze and reduce $\mathbf{Q}^{\mu} = L^{\mu}_{k \to l}(\mathbf{A})$ ($\mathbf{K}^{\mu} = L^{\mu}_{k \to k}(\mathbf{A})$ can be reduced analogously by letting $l = k$). From Equation (2.1) and Equation (2.2), we can write the computation of $\mathbf{Q}^{\mu}$ as follows, with α, λ equivalence classes of order-$(k+l)$ and order-l multi-indices and $\mathbf{k} \in [n]^k$:

$$\mathbf{Q}^{\mu}_{\mathbf{j}} = \sum_{\alpha} \sum_{\mathbf{k}} \mathbf{B}^{\alpha}_{\mathbf{k},\mathbf{j}} \mathbf{A}_{\mathbf{k}} w_{\alpha} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{j}} b_{\lambda},$$

$$\text{where} \quad \mathbf{B}^{\alpha}_{\mathbf{i},\mathbf{j}} = \begin{cases} 1 & (\mathbf{k}, \mathbf{j}) \in \alpha \\ 0 & \text{otherwise} \end{cases}; \mathbf{C}^{\lambda}_{\mathbf{j}} = \begin{cases} 1 & \mathbf{j} \in \lambda \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

A key idea is that, when we want $\mathbf{Q}^\mu_\mathbf{j}$ only for $\mathbf{j} \in \mu_q$, only a subset of equivalence classes among α or λ does effective computation and we can discard the rest. Specifically, we can discard an equivalence class α if it contains some $(\mathbf{k}, \mathbf{j})$ with $\mathbf{j} \notin \mu_q$. This is because, for such α, $(\mathbf{k}, \mathbf{j}) \notin \alpha$ if $\mathbf{j} \in \mu_q$, leading to $\mathbf{B}^\alpha_{\mathbf{k},\mathbf{j}} = 0$ if $\mathbf{j} \in \mu_q$. Therefore, such α does not contribute to $\mathbf{Q}^\mu_\mathbf{j}$ $\forall \mathbf{j} \in \mu_q$ and can be discarded. On the other hand, an equivalence class α containing some $(\mathbf{k}, \mathbf{j})$ with $\mathbf{j} \in \mu_q$ does effective computation and should be kept.

From that, it turns out that the number of effective α is $\leq \mathrm{b}(k + u_q)$, where $u_q = u(\mathbf{j}) = |\mu_q|$ is the number of unique entries within some $\mathbf{j} \in \mu_q$ (see Lemma 5). Recall that for an effective α, $\mathbf{j} \in \mu_q$ holds for all $(\mathbf{k}, \mathbf{j}) \in \alpha$. Within α's representative partition, as each $\mathbf{j} \in \mu_q$ has exactly u_q unique values, we always have $\{\mathbf{j}_1, ..., \mathbf{j}_l\}$ contained in exactly u_q distinct subsets. Thus, the possible number of effective α is upper-bounded by the number of ways of partitioning a set with $k + |\mu_q|$ elements, which is $\mathrm{b}(k + u_q)$. As for the bias, we can repeat the analysis with $k = 0$ and the number of effective λ is $\leq \mathrm{b}(u_q)$.

We now show that a lower-order linear layer $L^\mu_{k \to u_q}$ can compute $\mathbf{Q}^\mu_\mathbf{j}$ $\forall \mathbf{j} \in \mu_q$ in Equation (A.1). Let us denote $\mathcal{A}$ the set of all effective α and $\mathcal{L}$ the set of all effective λ. Then we can rewrite Equation (A.1) as:

$$\mathbf{Q}^\mu_\mathbf{j} = \sum_{\alpha \in \mathcal{A}} \sum_\mathbf{k} \mathbf{B}^\alpha_{\mathbf{k},\mathbf{j}} \mathbf{A}_\mathbf{k} w_\alpha + \sum_{\lambda \in \mathcal{L}} \mathbf{C}^\lambda_\mathbf{j} b_\lambda, \tag{A.2}$$

where $\mathcal{A}$ has $\leq \mathrm{b}(k + u_q)$ elements and $\mathcal{L}$ has $\leq \mathrm{b}(u_q)$ elements. Assume we have a linear layer $L^\mu_{k \to u_q}$. With $\mathbf{j}' \in [n]^{u_q}$, and β, θ equivalence classes of order-$(k + u_q)$ and order-u_q multi-indices respectively, we can write:

$$\begin{gathered}
\tilde{\mathbf{Q}}^\mu_{\mathbf{j}'} = \sum_\beta \sum_\mathbf{k} \mathbf{B}^\beta_{\mathbf{k},\mathbf{j}'} \mathbf{A}_\mathbf{k} w_\beta + \sum_\theta \mathbf{C}^\theta_{\mathbf{j}'} b_\theta, \\
\text{where} \quad \mathbf{B}^\beta_{\mathbf{k},\mathbf{j}'} = \begin{cases} 1 & (\mathbf{k}, \mathbf{j}') \in \beta \\ 0 & \text{otherwise} \end{cases} \quad ; \quad \mathbf{C}^\theta_{\mathbf{j}'} = \begin{cases} 1 & \mathbf{j}' \in \theta \\ 0 & \text{otherwise} \end{cases}
\end{gathered} \tag{A.3}$$

We now identify the condition that $\tilde{\mathbf{Q}}^\mu$ contains all $\mathbf{Q}^\mu_\mathbf{j}$ $\forall \mathbf{j} \in \mu_q$. For that, we need to define a mapping between index space of $\tilde{\mathbf{Q}}^\mu$ and $\mathbf{Q}^\mu$. To this end, we define a surjection $g : [k] \to [u_q]$ that satisfies $\mathbf{j}_a = \mathbf{j}_b \Leftrightarrow g(a) = g(b)$. We can always define such g due to the property of equivalence classes that, for all $a, b \in [l]$, $\mathbf{j}_a = \mathbf{j}_b$ holds iff $\mathbf{j}_a, \mathbf{j}_b$ belong to a same subset within μ_q's partition. By indexing the subsets within μ_q's partition, we define $g(a)$ $\forall a \in [l]$ as the index of the subset that a belongs. Then, for every $\mathbf{j} \in \mu_q$, we can find an order-u_q compact form $\mathbf{j}' \in [n]^{u_q}$ containing u_q unique elements through g: for $c \in [u_q]$ that $c = g(a) = g(b) = \cdots$, we construct $\mathbf{j}'$ such that $\mathbf{j}'_c = \mathbf{j}_a = \mathbf{j}_b = \cdots$. We define $f^q_\mu : [n]^q \to [n]^{u_q}$ as a mapping that gives $\mathbf{j}' = f^q_\mu(\mathbf{j})$ $\forall \mathbf{j} \in \mu_q$.

We now reduce Equation (A.2) into Equation (A.3). First, for each $\alpha \in \mathcal{A}$, we assign a distinct order-$(k + u_q)$ equivalence class β that satisfies: $(\mathbf{k}, \mathbf{j}) \in \alpha \Leftrightarrow (\mathbf{k}, \mathbf{j}') \in \beta$ with $\mathbf{j}' = f^q_\mu(\mathbf{j})$. This can be done by changing α's partition into β's, by merging each set of $\mathbf{j}_a = \mathbf{j}_b = ...$ into corresponding $\mathbf{j}'_c$ following f^q_μ. We similarly assign a distinct θ to each $\lambda \in \mathcal{L}$. Then, we set $w_\alpha = w_\beta$ for all paired α and β, and set $w_\beta = 0$ for every β not paired with any α. We similarly set $b_\theta = b_\lambda$ for all paired θ and λ, and $b_\theta = 0$ for every θ not paired with any λ. From the definition of basis tensors, $\mathbf{B}^\alpha_{\mathbf{k},\mathbf{j}} = \mathbf{B}^\beta_{\mathbf{k},\mathbf{j}'}$ for all paired α and β, and $\mathbf{C}^\lambda_\mathbf{j} = \mathbf{C}^\theta_{\mathbf{j}'}$ for all paired λ and θ. Therefore, we have $\tilde{\mathbf{Q}}^\mu_{\mathbf{j}'} = \mathbf{Q}^\mu_\mathbf{j}$ for all $\mathbf{j} \in \mu_q$. Conclusively, we can always construct $L^\mu_{k \to u_q}$ and $f^\mu_q : [n]^l \to [n]^{u_q}$ that gives $\mathbf{Q}^\mu_\mathbf{j}$ $\forall \mathbf{j} \in \mu_q$.

As noted in the beginning of the proof, we can perform the same analysis with $k = l$ to show the

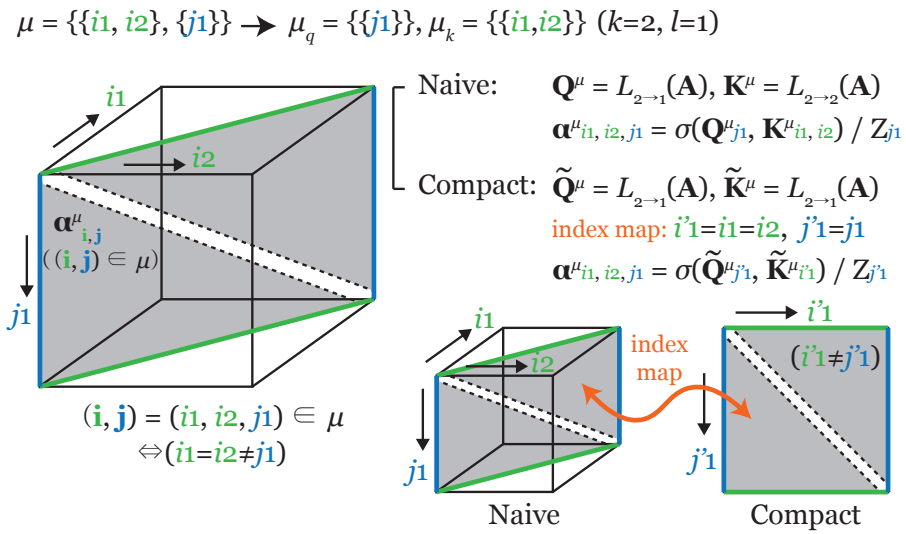


Figure A.1: Exemplar illustration of computing $\boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}}$ $\forall(\mathbf{i},\mathbf{j}) \in \mu$ with lower-order query and key, for $k = 2$, $l = 1$, $\mathbf{i} = (i_1, i_2)$, $\mathbf{j} = (j_1)$, and $\mu = \{\{i_1, i_2\}, \{j_1\}\}$.

analogous result for $\mathbf{K}^{\mu}$. We now have all entries $\mathbf{K}^{\mu}_{\mathbf{i}}$ $\forall \mathbf{i} \in \mu_k$ and $\mathbf{Q}^{\mu}_{\mathbf{j}}$ $\forall \mathbf{j} \in \mu_q$ to compute $\boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}}$ $\forall(\mathbf{i},\mathbf{j}) \in \mu$ (Equation (2.7)) and therefore Proposition 1 holds. □

In Figure A.1 we provide an example of computing $\boldsymbol{\alpha}^{\mu}$ with lower-order query and key.

Proof of Property 1

Proof. We begin by analyzing the complexity of an equivariant linear layer $L_{k\to l}$ (Equation (2.1)). In the inner summation $\sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}}$, with u_k and u_q the number of unique entries in $\mathbf{i}$ and $\mathbf{j}$ respectively, we have $n^{\leq u_k}$ effective multiplication and summations for each output index $\mathbf{j}$. Inequality is when $\mathbf{j}_b = \mathbf{i}_a$ for some a, b, which corresponds to indexing operation rather than summation. Thus, the number of operations done by the summation is $n^{u_q} n^{\leq u_k}$. With outer summation over μ, we have $\sum_{\mu} n^{u_q} n^{\leq u_k}$ operations. As $u_q \leq l$ and $u_k \leq k$ by definition, we have inequality $\sum_{\mu} n^{u_q} n^{\leq u_k} \leq \mathrm{b}(k+l) n^{k+l}$. Application of w_{μ} gives us $dd' \sum_{\mu} n^{u_q} n^{u_k} \leq \mathrm{b}(k+l) dd' n^{k+l}$ number of operations. For the bias, in the inner term $\mathbf{C}^{\lambda}_{\mathbf{j}}$, we need a single addition for each $\mathbf{j}$ and thus the number of operations for a λ is $n^{u_q} \leq n^l$. Summation over λ and application of bias parameters gives us $\mathrm{b}(l) d' n^{u_q} \leq \mathrm{b}(l) d' n^l$ operations. Collectively, we have $\leq \mathrm{b}(k+l) dd' n^{k+l} + \mathrm{b}(l) d' n^l$ number of operations. As k, l, d, d' are constants that does not depend on n, we obtain $\mathcal{O}(n^{k+l})$ complexity.

Computation of $\mathrm{Enc}_{k\to l}(\mathbf{A})$ (Equation (2.5)) involves computing $\mathrm{Attn}_{k\to l}(\mathbf{A})$, $\mathrm{MLP}_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A}))$ and adding them. Let us analyze $\mathrm{Attn}_{k\to l}(\mathbf{A})$ first. To compute $\boldsymbol{\alpha}^{h,\mu}$ from input, we need to compute $L^{\mu}_{k\to u_q}(\mathbf{A})$ and $L^{\mu}_{k\to u_k}(\mathbf{A})$, followed by pairwise similarity computation and re-indexing. Assuming that each pairwise similarity computation and indexing has constant complexity, we have $\mathcal{O}(n^{k+u_q} + n^{k+u_k} + n^{u_q+u_k})$.[1] As $u_q \leq l$ and $u_k \leq k$, we have $\mathcal{O}(n^{k+l} + n^{2k})$. It is worth to note that $\mathcal{O}(n^{2k})$ term comes from computation of keys from input. Having computed $\boldsymbol{\alpha}^{h,\mu}$, the inner summation $\sum_{\mathbf{i}} \boldsymbol{\alpha}^{h,\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}}$, similar to in $L_{k\to l}$, has $n^{u_q} n^{\leq u_k}$ computations. With outer summation over μ, we have $\sum_{\mu} n^{u_q} n^{\leq u_k} \leq \mathrm{b}(k+l) n^{k+l}$ operations. Summation over heads and application of weight matrices gives us $\leq \mathrm{b}(k+l) H d_H^2 d n^{k+l}$ operations, which is $\mathcal{O}(n^{k+l})$. For application of $\mathrm{MLP}_{l\to l}(\cdot)$, we sum the complexity of two linear

[1] Normalization over keys gives an additive complexity $\mathcal{O}(n^{u_q+u_k})$, which can be absorbed to the formula.

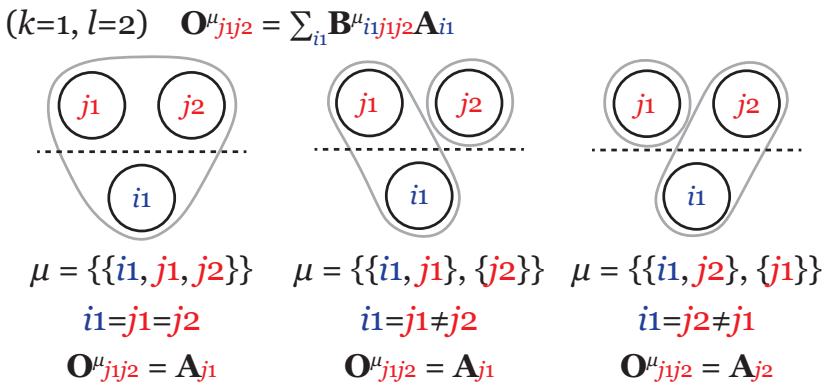


Figure A.2: Exemplar illustration of all equivalence classes included in lightweight linear layer $\bar{L}_{1\to 2}$.

layers $L_{l\to l}$ and element-wise ReLU, which gives us $\mathcal{O}(n^{2l})$. Conclusively, the complexity of $\mathrm{Enc}_{k\to l}$ is $\mathcal{O}(n^{2k} + n^{k+l} + n^{2l})$. □

Proof of Proposition 2

Proof. We assume $k, l > 0$. Among the equivalence classes μ of order-$(k+l)$ multi-indices, let us select a subset $\mathcal{M}$ that all $\mu \in \mathcal{M}$ satisfies the following: for all $(\mathbf{i}, \mathbf{j}) \in \mu$, $\mathbf{i}_a = \mathbf{j}_b$ holds for all $a \in [k]$ and some $b \in [l]$. In other words, every element in $\mathbf{i}$ is identical with at least one element in $\mathbf{j}$, and $\mathbf{i}$ becomes a single fixed multi-index when we fix $\mathbf{j}$ (we denote the fixed $\mathbf{i} = \mathrm{fix}(\mathbf{j})$). This renders $\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} = 1 \Leftrightarrow \mathbf{i} = \mathrm{fix}(\mathbf{j})$ for such μ, and consequently the inner-summation $\sum_{\mathbf{i}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}} w_{\mu}$ in Equation (2.9) reduces to elementwise indexing $\mathbf{A}_{\mathrm{fix}(\mathbf{j})} w_{\mu}$. As the size of $\mathcal{M}$ is upper-bounded by a constant $\mathrm{b}(k+l)$, we have $\mathcal{O}(n^l)$ complexity when computing Equation (2.9). With the trivial case $\mu = \{\{i_1, ..., i_k, j_1, ..., j_l\}\} \in \mathcal{M}$, we can always find nonempty $\mathcal{M}$. □

To provide some intuition, we illustrate all $\mu \in \mathcal{M}$ for $k = 1$, $l = 2$ in Figure A.2.

Proof of Property 2

Proof. We begin from sparse equivariant linear layer $L_{k\to l}$ (Equation (2.10)). In the inner summation $\sum_{\mathbf{i}\in E} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}}$, the number of multiplication and addition for each $\mathbf{j}$ is upper-bounded by $m = |E|$. As the number of output multi-indices $\mathbf{j}$ is bounded by $|E'| \le m\binom{k}{l}$, the effective number of operations are $\le m^2\binom{k}{l}$. With outer summation over μ, we have $\le \mathrm{b}(k+l)\binom{k}{l}m^2$ operations, leading to complexity $\mathcal{O}(m^2)$. For the lightweight linear layers $\bar{L}$ (Proposition 2) that precludes summation over input, we trivially have $\mathcal{O}(m)$ complexity as we do not sum over $\mathbf{i}$.

Now, we analyze the complexity of sparse self-attention computation (Equation (2.11)). To compute $\boldsymbol{\alpha}^{\mu}$ from input, we need to compute lightweight linear layers $\bar{L}^{\mu}_{k\to u_q}(\mathbf{A}, E)$ and $\bar{L}^{\mu}_{k\to u_k}(\mathbf{A}, E)$, followed by pairwise similarity computation of nonzero entries. As $u_q, u_k \le k$, we have complexity $\mathcal{O}(m)$ for the linear layers and $\mathcal{O}(m^2)$ for pairwise computation. Having computed $\boldsymbol{\alpha}^{\mu}$, the inner summation $\sum_{\mathbf{i}\in E} \boldsymbol{\alpha}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{A}_{\mathbf{i}}$ has $\le m$ computations. Enumerating over $\mathbf{j}$, we have $\mathcal{O}(m^2)$.

Finally, we analyze the complexity of $\mathrm{Enc}_{k\to l}$ composed of the sparse linear layers and self-attention. This involves adding the outputs of $\mathrm{Attn}_{k\to l}(\mathbf{A}, E)$ (which we already addressed) and $\mathrm{MLP}_{l\to l}(\mathrm{Attn}_{k\to l}(\mathbf{A}, E), E)$. For application of $\mathrm{MLP}_{l\to l}$, we sum the complexity of two lightweight linear layers $\bar{L}_{l\to l}$ and element-wise ReLU, which gives us $\mathcal{O}(m)$. In summary, the complexity of sparse $\mathrm{Enc}_{k\to l}$ is $\mathcal{O}(m^2)$. □

Proof of Property 3

Proof. Summation over $\mathbf{i} \in \mathcal{I}$ decouples $\mathbf{i}$ from $\mathbf{j}$ and allows reuse of computation over $\mathbf{j}$. As the summation over $\mathbf{i} \in \mathcal{I}$ involves $\mathcal{O}(n^k)$ operations and we share it over all query indices $\mathbf{j}$, self-attention reduces to elementwise application and we obtain $\mathcal{O}(n^k + n^l)$ complexity. As computation of $\tilde{\mathbf{Q}}^\mu$, $\tilde{\mathbf{K}}^\mu$ and application of $\mathrm{MLP}_{l\to l}$ are $\mathcal{O}(n^k + n^l)$ with lightweight linear layers, we have $\mathcal{O}(n^k + n^l)$ collective complexity for $\mathrm{Enc}_{k\to l}$. When adopted into sparse $\mathrm{Enc}_{k\to l}$ (Equation (2.11)), summation over $\mathbf{i}$ and enumeration over $\mathbf{j}$ all reduce to $\mathcal{O}(m)$ and we thereby have $\mathcal{O}(m)$ complexity. □

Proof of Theorem 2

Proof. With message function $M : \mathbb{R}^{2d_v+d_e} \to \mathbb{R}^{d_m}$ and update function $U : \mathbb{R}^{d_v+d_m} \to \mathbb{R}^d$, a message passing step takes node features $\mathbf{X} \in \mathbb{R}^{n\times d_v}$ and edge features $\mathbf{E} \in \mathbb{R}^{n\times n\times d_e}$ as input and outputs node features $\mathbf{H} \in \mathbb{R}^{n\times d}$ according to following. With $i, j \in [n]$:

$$\mathbf{M}_j = \sum_{i\in\mathcal{N}(j)} M(\mathbf{X}_j, \mathbf{X}_i, \mathbf{E}_{ij})$$
$$\mathbf{H}_j = U(\mathbf{X}_j, \mathbf{M}_j),$$

where $\mathcal{N}(j)$ denotes incoming neighbors of j-th node, *i.e.,* $\{i|(i,j) \in E\}$.

We now show how a composition of two $\mathrm{Enc}_{2\to 2}$ can approximate above computation.

1. As a first step, we encode $\mathbf{X}$ and $\mathbf{E}$ into a single $\mathbf{A} \in \mathbb{R}^{n\times n\times(2d_v+d_e)}$ [Maron et al., 2019b]. In the first d_v channels, we replicate $\mathbf{X}$ on the rows. In the next d_v channels, we replicate $\mathbf{X}$ on the columns. In the last d_e channels, we put $\mathbf{E}$. Additionally, to account for output positions (node features), we augment E with self-loops and make $E' = E \cup \{(i,i)\ \forall i \in [n]\}$.

2. Then, we make the first $\mathrm{Enc}_{2\to 2}$ approximate the message function $M(\cdot)$, so that $\mathrm{Enc}_{2\to 2}(\mathbf{A})_{ij} \approx M(\mathbf{X}_j, \mathbf{X}_i, \mathbf{E}_{ij})$. To do this, we first reduce $\mathrm{Attn}_{2\to 2}(\mathbf{A})_{ij} = \mathbf{A}_{ij}$ and apply $\mathrm{MLP}_{l\to l}$ on top of it. We reduce $\mathrm{MLP}_{l\to l}$ to entry-wise MLP. As $\mathrm{Attn}_{2\to 2}(\mathbf{A})_{ij} = \mathbf{A}_{ij}$ is a concatenation of $\mathbf{X}_i$, $\mathbf{X}_j$, $\mathbf{E}_{ij}$, with universal approximation theorem [Hornik et al., 1989], we can have the output of the first $\mathrm{Enc}_{2\to 2}(\mathbf{A})_{ij} = M(\mathbf{X}_j, \mathbf{X}_i, \mathbf{E}_{ij}) + \epsilon_1\ \forall i, j$ where ϵ_1 is approximation error. This also holds when we leverage sparsity and restrict the index scope to $(i,j) \in E'$; in this case, we make $(i,j) \in E' \setminus E$ contain zero vectors.

3. Before feeding the output to the second $\mathrm{Enc}_{2\to 2}$, we concatenate the original input $\mathbf{A}$ with the output of the first layer in the channel dimension to make $\mathbf{A}' \in \mathbb{R}^{n\times n\times(2d_v+d_e+d_m)}$. This gives $\mathbf{X}_i, \mathbf{X}_j, \mathbf{E}_{ij}$ encoded in the first $2d_v + d_e$ channels and $M(\mathbf{X}_j, \mathbf{X}_i, \mathbf{E}_{ij}) + \epsilon_1$ encoded in the last d_m channels of $\mathbf{A}'$. This operation can be trivially absorbed within the MLP of the first layer, but we separate for simplicity.

4. Now, we make the second $\mathrm{Enc}_{2\to 2}$ jointly approximate summation of messages over neighbors $\sum_{i\in\mathcal{N}(j)}(\cdot)$ and update function $U(\cdot)$, so that $\mathrm{Enc}_{2\to 2}(\mathbf{A}')_{jj} \approx \mathbf{H}_j = U(\mathbf{X}_j, \mathbf{M}_j)$. First, we reduce $\mathrm{Attn}_{2\to 2}(\mathbf{A}')$ to summation over neighbors. For this we only need two equivalence classes $\mu_1 = \{\{1\}, \{2,3,4\}\}$ and $\mu_2 = \{\{1,2,3,4\}\}$. Omitting normalization, we can write Equation (2.14) as follows. For μ_1 we set $u_q = 1$, $u_k = 2$, $\mathbf{i} = (i,j)$, $\mathbf{j} = (j,j)$, $\mathbf{i}' = (i,j)$, $\mathbf{j}' = j$, and for μ_2 we set

$u_q = 1$, $u_k = 1$, $\mathbf{i} = (j, j)$, $\mathbf{j} = (j, j)$, $\mathbf{i}' = j$, $\mathbf{j}' = j$.

$$\text{Attn}_{2\to 2}(\mathbf{A}')_{jj} = \phi(\tilde{\mathbf{Q}}_j^{\mu_1})^\top \sum_{\{i|(i,j)\in E'\}} \phi(\tilde{\mathbf{K}}_{ij}^{\mu_1})\mathbf{A}'_{ij} w_{\mu_1} + \phi(\tilde{\mathbf{Q}}_j^{\mu_2})^\top \phi(\tilde{\mathbf{K}}_j^{\mu_2})\mathbf{A}'_{jj} w_{\mu_2}. \tag{A.4}$$

Let entries in $\phi(\tilde{\mathbf{K}}^\mu)$, $\phi(\tilde{\mathbf{Q}}^\mu)$ be $\frac{1_{d_K}}{\sqrt{d_K}}$ so that their dot product is 1. Equation (A.4) reduces to:

$$\text{Attn}_{2\to 2}(\mathbf{A}')_{jj} = \sum_{\{i|(i,j)\in E'\}} \mathbf{A}'_{ij} w_{\mu_1} + \mathbf{A}'_{jj} w_{\mu_2}$$

We make w_{μ_1} zero-out the first $2d_v + d_e$ channels and w_{μ_2} zero-out the last $d_e + d_m$ channels. Then, we have $\text{Attn}_{2\to 2}(\mathbf{A}')_{jj}$ contain $\mathbf{X}_j$ in the first d_v channels and $\mathbf{M}_j + \epsilon_2 = \sum_{i|(i,j)\in E'}(M(\mathbf{X}_j, \mathbf{X}_i, \mathbf{E}_{ij})) + \epsilon_2$ in the last d_m channels where ϵ_2 is approximation error[2]. We then apply $\text{MLP}_{l\to l}$ on top of it, which can approximate the update function by universal approximation theorem [Hornik et al., 1989] and we have $\text{Attn}_{2\to 2}(\mathbf{A}')_{jj} = U(\mathbf{X}_j, \mathbf{M}_j) + \epsilon_3$ where ϵ_3 is approximation error.

Overall, the approximation error ϵ_i at each step depends on ϵ_{i-1} $(i > 1)$, the MLP that approximates relevant function, and uniform bounds and uniform continuity of the approximated functions [Hornik et al., 1989].

In the opposite, message passing cannot approximate some of the operations done by a single transformer layer $\text{Enc}_{2\to 2}$. This can be seen from the fact that, given a graph with diameter $d(E)$, we need at least $d(E)$ message passing operations to approximate output of $\text{Enc}_{2\to 2}$. This is because a single $\text{Enc}_{2\to 2}$ can impose dependency between any pair of input and output indices $\mathbf{i}, \mathbf{j}$, while message passing requires $d(E)$ steps in the worst case. Consequently, the approximation becomes impossible when the graph contains > 1 disconnected components, which leads to $d(E) \to +\infty$. □

[2]Note that message summation over $\{i|(i,j) \in E'\}$ is equivalent to summation over $\{i|(i,j) \in E\} = \mathcal{N}(j)$ because we set message zero at $(i,j) \in E' \setminus E$.

A.2 Appendix of equivariant hypergraph neural networks

A.2.1 Proofs

In Equation (2.15), we recall that the equivalence classes are defined upon equivalence relation $\sim$ of multi-indices that partitions the multi-index space $[n]^k$ into $[n]^k/_\sim$ [Maron et al., 2019b, Kim et al., 2021a]. The equivalence relation $\sim$ is defined as follows: for $\mathbf{i}, \mathbf{j} \in [n]^k$, the equivalence relation sets $\mathbf{i} \sim \mathbf{j}$ if and only if $(i_1, ..., i_k) = (\pi(j_1), ..., \pi(j_k))$ holds for some node permutation $\pi \in S_n$. As a result, a multi-index $\mathbf{i}$ and all elements $\mathbf{j}$ of its equivalence class μ have an identical (permutation-invariant) equality pattern: $\mathbf{i}_a = \mathbf{i}_b \Leftrightarrow \mathbf{j}_a = \mathbf{j}_b$ for all $\mathbf{i}, \mathbf{j} \in \mu$. Each equivalence class μ (or λ) is thus described as a specific set of all order-$(k+l)$ (order-l) multi-indices with an identical (permutation-invariant) equality pattern.

For Proposition 4 and Theorem 3, it is convenient that we first derive the maximally expressive equivariant linear layers for *symmetric* input and output tensors. This is because all tensors under consideration in this work are symmetric due to the unorderedness of hypergraphs (see Definition 2).

We show that if input and output of $L_{k\to l}$ (Equation (2.15)) are constrained to be *symmetric*, the layer is described with *coarser* partitioning of index spaces (in terms of partition refinement) specified by equivalence relations that are invariant to axis permutation on top of node permutation:

Lemma 6. *If input and output of an equivariant linear layer $L_{k\to l}$ (Equation (2.15)) are constrained to be symmetric tensors, it can be reduced to the following $L_{[k]\to[l]}$:*

$$L_{[k]\to[l]}(\mathbf{A})_{\mathbf{j}} = \sum_{\alpha} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}_{\mathbf{i}} w_\alpha + \sum_{\beta} \mathbb{1}_{\mathbf{j}\in\beta} b_\beta, \tag{A.5}$$

where α and β are equivalence classes that specify the partitioning $[n]^{k+l}/_{\sim_\alpha}$ and $[n]^l/_{\sim_\beta}$ respectively, defined as the following:

1. *The equivalence classes α are defined upon the equivalence relation $\sim_\alpha$ that, for $\mathbf{i}, \mathbf{j} \in [n]^{k+l}$, relates $\mathbf{i} \sim_\alpha \mathbf{j}$ if and only if the following holds for some node permutation $\pi \in S_n$ and axis permutations $\pi_k \in S_k$, $\pi_l \in S_l$:*

$$(i_1, ..., i_{k+l}) = \left(\pi(j_{\pi_k(1)}), ..., \pi(j_{\pi_k(k)}), \pi(j_{k+\pi_l(1)}), ..., \pi(j_{k+\pi_l(l)})\right). \tag{A.6}$$

2. *The equivalence classes β are defined upon the equivalence relation $\sim_\beta$ that, for $\mathbf{i}, \mathbf{j} \in [n]^l$, relates $\mathbf{i} \sim_\beta \mathbf{j}$ if and only if the following holds for some node permutation $\pi \in S_n$ and axis permutation $\pi_l \in S_l$:*

$$(i_1, ..., i_l) = \left(\pi(j_{\pi_l(1)}), ..., \pi(j_{\pi_l(l)})\right). \tag{A.7}$$

Proof. We begin from $L_{k\to l}$ (Equation (2.15)) and reduce its parameters without affecting the output based on symmetry of input and output. For $\pi_k \in S_k$, we denote *axis permutation* of a multi-index $\mathbf{i} \in [n]^k$ as $\pi_k(i_1, ..., i_k) = (i_{\pi_k(1)}, ..., i_{\pi_k(k)})$ and denote axis permutation of a tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$ as $(\pi_k \cdot \mathbf{A})_{\mathbf{i}} = \mathbf{A}_{\pi_k^{-1}(\mathbf{i})}$. With symmetry, input and output of $L_{k\to l}$ are constrained to be axis permutation invariant, i.e., the following holds:

$$L_{k\to l}(\mathbf{A}) = \pi_l \cdot L_{k\to l}(\pi_k \cdot \mathbf{A}),$$

for all $\pi_k \in S_k$, $\pi_l \in S_l$, and symmetric $\mathbf{A} \in \mathbb{R}^{n^k \times d}$. In other words:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\pi_l^{-1}(\mathbf{j}))\in\mu} \mathbf{A}_{\pi_k^{-1}(\mathbf{i})} w_\mu + \sum_{\lambda} \mathbb{1}_{\pi_l^{-1}(\mathbf{j})\in\lambda} b_\lambda,$$

which, denoting π_l^{-1} as π_l and noting π_k is a bijection on $[n]^k$, leads to following:

$$L_{k\to l}(\mathbf{A})_{\mathbf{j}} = \sum_{\mu} \sum_{\mathbf{i}} \mathbb{1}_{(\pi_k(\mathbf{i}),\pi_l(\mathbf{j}))\in\mu} \mathbf{A}_{\mathbf{i}} w_\mu + \sum_{\lambda} \mathbb{1}_{\pi_l(\mathbf{j})\in\lambda} b_\lambda. \tag{A.8}$$

We now reduce biases. From Equation (A.8), we see that by constraining the output to be symmetric, $\sum_\lambda \mathbb{1}_{\mathbf{j}\in\lambda} b_\lambda = \sum_\lambda \mathbb{1}_{\pi_l(\mathbf{j})\in\lambda} b_\lambda$ holds for all $\mathbf{j} \in [n]^l$ and $\pi_l \in S_l$. We can see that this holds if and only if $b_{\lambda_1} = b_{\lambda_2}$ for all (λ_1, λ_2) such that, for some $\mathbf{j} \in \lambda_1$, $\pi_l(\mathbf{j}) \in \lambda_2$ holds for some $\pi_l \in S_l$. By writing $b_\beta = b_{\lambda_1} = b_{\lambda_2}$, we can interpret β as a new equivalence class of multi-indices formed as a union of all such $\{\lambda_1, \lambda_2, ...\}$ (thereby specifying a coarser partitioning than $\sim$), or equivalently, by collecting all multi-indices related by node and axis permutations, i.e., equivalence relation $\sim_\beta$. Thus, we can reduce biases as $\sum_\beta \mathbb{1}_{\mathbf{j}\in\beta} b_\beta$.

We now reduce weights. From Equation (A.8), we see that by constraining the input and output to be symmetric, $\sum_\mu \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\mu} \mathbf{A}_{\mathbf{i}} w_\mu = \sum_\mu \sum_{\mathbf{i}} \mathbb{1}_{(\pi_k(\mathbf{i}),\pi_l(\mathbf{j}))\in\mu} \mathbf{A}_{\mathbf{i}} w_\mu$ holds for all $\mathbf{i} \in [n]^k$, $\mathbf{j} \in [n]^l$, $\pi_k \in S_k$, $\pi_l \in S_l$, and symmetric $\mathbf{A} \in \mathbb{R}^{n^k \times d}$. Similar to biases, this holds if and only if $w_{\mu_1} = w_{\mu_2}$ for all (μ_1, μ_2) such that, for some $(\mathbf{i}, \mathbf{j}) \in \mu_1$, $(\mathbf{i}, \pi_l(\mathbf{j})) \in \mu_2$ holds for some $\pi_l \in S_l$. By writing $w_\beta = w_{\mu_1} = w_{\mu_2}$, we can interpret β as a new equivalence class of multi-indices formed as a union of all such $\{\mu_1, \mu_2, ...\}$, or equivalently, by collecting all multi-indices related by node and output axis permutations. On top of that, the symmetry of input tensor gives us another room to reduce the parameters. We can see that, for all (β_1, β_2) such that $(\pi_k(\mathbf{i}), \mathbf{j}) \in \beta_2$ holds for some $(\mathbf{i}, \mathbf{j}) \in \beta_1$ and $\pi_k \in S_k$, we can replace both w_{β_1} and w_{β_2} by $w_\alpha = (w_{\beta_1} + w_{\beta_2})/2$ and obtain exactly same output. Extending, for any collection $\{\beta_1, \beta_2, ...\}$ of all such pairwise related β's, we can replace $w_{\beta_1}, w_{\beta_2}, ...$ by $w_\alpha = (w_{\beta_1} + w_{\beta_2} + ...)/|\{\beta_1, \beta_2, ...\}|$ and obtain exactly same output. Thus, we can interpret α as a new equivalence class of multi-indices formed as a union of all such $\{\beta_1, \beta_2, ...\}$, or equivalently, by collecting all multi-indices related by node, input axis, and output axis permutations i.e., equivalence relation $\sim_\alpha$. Thus, we can reduce weights as $\sum_\alpha \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}_{\mathbf{i}} w_\alpha$. □

Proof of Proposition 4

We begin from a simple lemma:

Lemma 7. *For a symmetric order-k tensor $\mathbf{A}^{(k)}$ that represents a k-uniform hypergraph (Equation (2.16)), all indices $\mathbf{i}$ of nonzero entries contain k distinct elements.*

Proof. In Equation (2.16), recall that $\mathbf{A}^{(k)}_{(i_1,...,i_k)} \neq 0$ only if $\{i_1, ..., i_k\} \in E^{(k)}$. As $E^{(k)}$ is a set of k-uniform hyperedges that contain k distinct node indices, every multi-indices $(i_1, ..., i_k)$ of nonzero entries of $\mathbf{A}^{(k)}$ contains k distinct elements. □

Now, we prove Proposition 4.

Proof. We begin from $L_{[k]\to[l]}$ in Equation (A.5) and reduce the parameters without affecting the output. By further constraining the (already symmetric) input and output to symmetric tensors that represent

uniform hypergraphs, we first write:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\alpha}\sum_{\mathbf{i}}\mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\alpha} + \sum_{\beta}\mathbb{1}_{\mathbf{j}\in\beta}b_{\beta}\right). \tag{A.9}$$

Note that two constraints are added, $\mathbb{1}_{|\mathbf{i}|=k}$ multiplied to the input and $\mathbb{1}_{|\mathbf{j}|=l}$ multiplied to the output. This comes from the fact that the input/output of the layer are order-k/l tensors that represent k/l-uniform hypergraphs, respectively. As Lemma 7 states, the input/output must contain nonzero entry only for indices that contain k/l distinct elements, respectively.

We first reduce biases. The constraint $\mathbb{1}_{|\mathbf{j}|=l}$ leaves only a single bias b_{β_l} of equivalence class β_l and discards the rest (i.e., $|\mathbf{j}| = l$ if and only if $\mathbf{j} \in \beta_l$). This particular equivalence class β_l specifies that all multi-index entries $j_1, ..., j_l$ are unique. For any other equivalence class $\beta' \neq \beta_l$, the partition that represents it ties at least two entries of $\mathbf{j} \in \beta'$ ($j_a = j_b$ for $a \neq b$). This gives $|\mathbf{j}| < l$ for all $\mathbf{j} \in \beta'$, which leads to $\mathbf{j} \notin \beta'$ for all $|\mathbf{j}| = l$. Thus we have $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{\mathbf{j}\in\beta'} = 0$ for all $\mathbf{j}$, meaning $b_{\beta'}$ cannot affect the output in Equation (A.9). We can safely remove all $\beta' \neq \beta_l$ from the bias, and the layer reduces to the following where $b_l = b_{\beta_l}$:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\alpha}\sum_{\mathbf{i}}\mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\alpha} + b_l\right). \tag{A.10}$$

We now reduce weights. Similar to bias, the joint constraint $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}|=k}$ leaves exactly $1 + \min(k, l)$ weights and discards the rest. We derive this by removing all equivalence classes α that never affect the output. For any equivalence class α' that ties at least two entries within $\mathbf{i}$ or $\mathbf{j}$ for some $(\mathbf{i}, \mathbf{j}) \in \alpha'$ ($i_a = i_b$ or $j_a = j_b$ for $a \neq b$), we have $|\mathbf{i}| < k$ or $|\mathbf{j}| < l$ for all $(\mathbf{i}, \mathbf{j}) \in \alpha'$, which leads to $(\mathbf{i}, \mathbf{j}) \notin \alpha'$ for all $|\mathbf{i}| = k$ and $|\mathbf{j}| = l$. Thus we have $\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}|=k}\mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha'} = 0$ for all $(\mathbf{i}, \mathbf{j})$, meaning $w_{\alpha'}$ cannot affect the output in Equation (A.9) and can be safely removed. We now have a reduced set of equivalence classes α such that for any $(\mathbf{i}, \mathbf{j}) \in \alpha$, $|\mathbf{i}| = k$ and $|\mathbf{j}| = l$. Notably, due to axis permutation symmetry described in Lemma 6, we can see that each equivalence class α can be described exactly by the number of equivalences $i_a = j_b$ between $\mathbf{i}$ and $\mathbf{j}$, which we denote as $\mathcal{I}$ (i.e., if α is described by $\mathcal{I}$, $|\mathbf{i} \cap \mathbf{j}| = \mathcal{I}$ if and only if $(\mathbf{i}, \mathbf{j}) \in \alpha$). Thus, by discarding irrelevant equivalence classes and rewriting in terms of the overlap $\mathcal{I}$, we can reduce the weight as follows, where $w_{\mathcal{I}} = w_{\alpha}$ for α that corresponds to $\mathcal{I}$:

$$\sum_{\alpha}\sum_{\mathbf{i}}\mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\alpha} = \sum_{\mathcal{I}=0}^{\min(k,l)}\sum_{\mathbf{i}}\mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\mathcal{I}}.$$

With this, the layer in Equation (A.10) reduces to:

$$L_{(k)\to(l)}(\mathbf{A}^{(k)})_{\mathbf{j}} = \mathbb{1}_{|\mathbf{j}|=l}\left(\sum_{\mathcal{I}=0}^{\min(k,l)}\sum_{\mathbf{i}}\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\mathbf{A}^{(k)}_{\mathbf{i}}w_{\mathcal{I}} + b_l\right). \tag{A.11}$$

Note that $\mathbb{1}_{|\mathbf{i}|=k}$ in weight application was removed as $\mathbf{A}^{(k)}$ already contains nonzero entries only for $|\mathbf{i}| = k$ (see Lemma 7).

We finish by handling $\mathcal{I} = 0$ in weight application as a special case:

$$\begin{aligned}
\mathbb{1}_{|\mathbf{j}|=l} \sum_{\mathcal{I}=0}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}} &= \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w_{\mathcal{I}} + \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=0} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right) \\
&= \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} (w_{\mathcal{I}} - w_0) + \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|\geq 0} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right) \\
&= \mathbb{1}_{|\mathbf{j}|=l} \left(\sum_{\mathcal{I}=1}^{\min(k,l)} \sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} \mathbf{A}_{\mathbf{i}}^{(k)} w'_{\mathcal{I}} + \sum_{\mathbf{i}} \mathbf{A}_{\mathbf{i}}^{(k)} w_0 \right).
\end{aligned}$$

This modification is not strictly required, but makes implementation much easier as $\mathcal{I} = 0$ case reduces to global pooling. By rewriting $w'_{\mathcal{I}}$ as $w_{\mathcal{I}}$, we finally arrive at the reduced layer in Equation (2.17). □

Proof of Theorem 3

As a direct proof upon tensor sequences is not straightforward, we first *pack* the entire sequence of tensors $\mathbf{A}^{(:K)}$ that represent a hypergraph into an equivalent symmetric order-K tensor $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$. Then, we show that the maximally expressive equivariant linear layer $L_{[K]\to[L]} : \mathbb{R}^{n^K \times d} \to \mathbb{R}^{n^L \times d}$ for the packed tensors is equivalent to $L_{(:K)\to(:L)}$ for tensor sequences, and thus they are maximally expressive.

We first characterize the packed tensors with the following lemma:

Lemma 8. *A sequence of tensors $\mathbf{A}^{(:K)} = \left(\mathbf{A}^{(k)}\right)_{k\leq K}$ that represent a hypergraph (Definition 3) can be represented as an equivalent symmetric order-K tensor $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$.*

Proof. Let us define $\mathbf{A}^{[K]} \in \mathbb{R}^{n^K \times d}$ as follows:

$$\mathbf{A}^{[K]}_{(i_1,\ldots,i_K)} = \mathbf{A}^{(|\{i_1,\ldots,i_K\}|)}_{\{i_1,\ldots,i_K\}}, \tag{A.12}$$

where $\{i_1, ..., i_K\}$ denotes the set of unique entries in $(i_1, ..., i_K)$ ordered arbitrarily (note that arbitrary ordering always gives the same value as $\mathbf{A}^{(k)}$ is symmetric). From $\mathbf{A}^{[K]}$, we can retrieve the original sequence of tensors $\left(\mathbf{A}^{(k)}\right)_{k\leq K}$ by using all order-K multi-indices that contain k unique entries to index $\mathbf{A}^{[K]}$ to construct each $\mathbf{A}^{(k)}$. A convenient property of $\mathbf{A}^{[K]}$ is that, for any pair of multi-indices $\mathbf{i}, \mathbf{j} \in [n]^K$ that contain the same number of unique elements, $\mathbf{A}^{[K]}_{\mathbf{i}} = \mathbf{A}^{[K]}_{\mathbf{j}}$ holds. □

Now, we prove Theorem 3. The proof is analogous to Lemma 6 as we reduce the weights and biases by leveraging the constraint on input and output (Equation (A.12)).

Proof. We prove by showing that $L_{[K]\to[L]}$ (Equation (A.5)) is equivalent with collection $\left(L_{(k)\to(l)}\right)_{k\leq K, l\leq L}$ (Equation (A.11)). We begin by writing $L_{[K]\to[L]}$ as a maximally expressive equivariant linear layer for symmetric tensors (Equation (A.5)):

$$L_{[K]\to[L]}(\mathbf{A}^{[K]})_{\mathbf{j}} = \sum_{\alpha} \sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}^{[K]}_{\mathbf{i}} w_\alpha + \sum_{\beta} \mathbb{1}_{\mathbf{j}\in\beta} b_\beta. \tag{A.13}$$

Let us first reduce the biases. By definition, the bias $\sum_\beta \mathbb{1}_{\mathbf{j}\in\beta} b_\beta$ (Equation (A.13)) is constrained to be an instantiation of the tensor described in Equation (A.12). Due to the property of such tensors, bias entries at all multi-indices having a specific number of unique elements are constrained to have an identical value. This holds if and only if $b_{\beta_1} = b_{\beta_2}$ for all (β_1, β_2) such that, any $\mathbf{j}_1 \in \beta_1$ and $\mathbf{j}_2 \in \beta_2$

have a same number (say, l) of unique elements. By writing $b_{\beta_l} = b_{\beta_1} = b_{\beta_2}$, β_l can be interpreted as a new equivalence class formed as a union of all such $\{\beta_1, \beta_2, ...\}$, which is equivalently the collection of all multi-indices with l unique elements. With this, the biases in Equation (A.5) can be rewritten as follows:

$$\sum_{\beta} \mathbb{1}_{\mathbf{j}\in\beta} b_\beta = \sum_{l\le L} \mathbb{1}_{|\mathbf{j}|=l} b_{\beta_l},$$

which is equivalent to the collection of biases in $\left(L_{(k)\to(l)}\right)_{k\le K, l\le L}$ (Equation (A.11)).

We now move to the weight. By definition, the input and output of $L_{[K]\to[L]}$ are constrained to be tensors described in Equation (A.12). As a result, input entries at all multi-indices with a specific number of unique elements have an identical value; and the same constrained is applied for output as well. Similar to the bias, this holds if and only if $w_{\alpha_1} = w_{\alpha_2}$ for all (α_1, α_2) such that, for some $(\mathbf{i}, \mathbf{j}_1) \in \alpha_1$, $(\mathbf{i}, \mathbf{j}_2) \in \alpha_2$ where $\mathbf{j}_1$ and $\mathbf{j}_2$ have an identical number (l) of unique elements. By writing $w_{\alpha'} = w_{\alpha_1} = w_{\alpha_2}$, α' is interpreted as a new equivalence class formed as a union of all such $\{\alpha_1, \alpha_2, ...\}$. In addition, we can leverage the structure of the input to further reduce the weight parameters. For all (α'_1, α'_2) such that, for some $(\mathbf{i}_1, \mathbf{j}) \in \alpha'_1$, $(\mathbf{i}_2, \mathbf{j}) \in \alpha'_2$ where $\mathbf{i}_1$ and $\mathbf{i}_2$ have an identical number (k) of unique elements and identical number ($\mathcal{I}$) of overlapping unique elements to $\mathbf{j}$, we can replace both $w_{\alpha'_1}$ and $w_{\alpha'_2}$ by $w_{\alpha_{k,l,\mathcal{I}}} = (w_{\alpha'_1} + w_{\alpha'_2})/2$ and obtain exactly same output. Extending this, for any collection $\{\alpha'_1, \alpha'_2, ...\}$ of all such pairwise related α', we can replace $w_{\alpha'_1}, w_{\alpha'_2}, ...$ by $w_{\alpha_{k,l,\mathcal{I}}} = (w_{\alpha'_1} + w_{\alpha'_2} + ...)/|\{\alpha'_1, \alpha'_2, ...\}|$ and still obtain the exact same output. $\alpha_{k,l,\mathcal{I}}$ is thus interpreted as a new equivalence class formed as a union of all such $\{\alpha'_1, \alpha'_2, ...\}$, or equivalently, formed as a collection of all multi-indices $(\mathbf{i}, \mathbf{j})$ where $\mathbf{i} \in [n]^K, \mathbf{j} \in [n]^L$ with k unique entries in $\mathbf{i}$, l unique entries in $\mathbf{j}$, and $\mathcal{I}$ tying relations $i_a = i_b = ... = j_c = j_d$. With this, the weights in Equation (A.5) can be rewritten as follows:

$$\sum_{\alpha}\sum_{\mathbf{i}} \mathbb{1}_{(\mathbf{i},\mathbf{j})\in\alpha} \mathbf{A}^{[K]}_{\mathbf{i}} w_\alpha = \sum_{k\le K}\sum_{l\le L}\sum_{\mathcal{I}=0}^{\min(K,L)}\sum_{\mathbf{i}} \mathbb{1}_{|\mathbf{j}|=l}\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}}\mathbb{1}_{|\mathbf{i}|=k}\mathbf{A}^{[K]}_{\mathbf{i}} w_{\alpha_{k,l,\mathcal{I}}},$$

which is computationally equivalent to the collection of weight applications in $\left(L_{(k)\to(l)}\right)_{k\le K, l\le L}$. □

Proof of Theorem 4

Proof. Our target of approximation is the output of weight application from Equation (2.20):

$$w(\mathbf{A}^{(:K)})_{l,\mathbf{j}} = \sum_{\mathcal{I}=0}^{K}\sum_{k\le K}\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}} \mathcal{W}(k,l,\mathcal{I}) \qquad \text{where} \qquad \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} = \begin{cases} \mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} & \text{if } \mathcal{I} \ge 1 \\ 1 & \text{if } \mathcal{I} = 0 \end{cases}.$$

Note that we moved the summation $\sum_{\mathcal{I}=1}^{\min(k,l)}$ out of $\sum_{k\le K}$, and adjusted the summation range to $\sum_{\mathcal{I}=1}^{K}$. This does not change the output because for any $\mathcal{I} \ge \min(k,l)$ we have $\mathbb{1}_{|\mathbf{i}\cap\mathbf{j}|=\mathcal{I}} = 0$.

We now introduce MLPs $\phi_1 : \mathbb{N}\times\mathbb{R}^d \to \mathbb{R}^{Kd}$, $\phi_2 : \mathbb{N}\times\mathbb{R}^{Kd} \to \mathbb{R}^{(1+K)Kd}$, and $\phi_3 : \mathbb{N}\times\mathbb{R}^{(1+K)Kd} \to \mathbb{R}^d$ to model appropriate functions based on universal approximation [Hornik et al., 1989].

First, we make ϕ_1 output the following, where ϵ_1 is approximation error:

$$\phi_1(k, \mathbf{X})_{(k-1)d+1:kd} = \mathbf{X} + \epsilon_1, \tag{A.14}$$

so that $\phi_1(0, \mathbf{X})$ places $\mathbf{X}$ on the first d channels of the output, $\phi_1(1, \mathbf{X})$ places $\mathbf{X}$ on the $d+1 : 2d$ channels of the output, and so on. Then, $\sum_{k\le K}\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}})$ gives channel concatenation of

$\left(\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}}\right)_{k \leq K}$, which we denote $\mathbf{Y}_{\mathcal{I}}$.

Then, we make ϕ_2 output the following, where ϵ_2 is approximation error:

$$\phi_2(\mathcal{I}, \mathbf{Y})_{\mathcal{I}Kd+1:(\mathcal{I}+1)Kd} = \mathbf{Y} + \epsilon_2. \tag{A.15}$$

Then, $\sum_{\mathcal{I}=0}^{K} \phi_2(\mathcal{I}, \mathbf{Y}_{\mathcal{I}})$ gives a concatenation of $\mathbf{Y}_{0:K}$, which is equivalently concatenation of $\left(\left(\sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \mathbf{A}^{(k)}_{\mathbf{i}}\right)_{k \leq K}\right)_{\mathcal{I} \leq K}$.

Finally, we let ϕ_3 output the following, where ϵ_3 is approximation error:

$$\phi_3(l, \mathbf{Z}) = \sum_{\mathcal{I}=0}^{K} \sum_{k \leq K} \mathbf{Z}_{(\mathcal{I}K+k-1)d+1:(\mathcal{I}K+k)d} \mathcal{W}(k, l, \mathcal{I}) + \epsilon_3, \tag{A.16}$$

which, by indexing back through $\mathbf{Z}$, $\mathbf{Y}$, and $\mathbf{X}$, gives $w(\mathbf{A}^{(:K)})_{l,\mathbf{j}}$.

We finish by rewriting $\phi_3(l, \mathbf{Z})$ as follows:

$$\phi_3(l, \mathbf{Z}) = \phi_3\left(l, \sum_{\mathcal{I} \geq 0} \phi_2\left(\mathcal{I}, \sum_{k \leq K} \sum_{\mathbf{i}} \mathbf{B}^{\mathcal{I}}_{\mathbf{i},\mathbf{j}} \phi_1(k, \mathbf{A}^{(k)}_{\mathbf{i}})\right)\right),$$

which is equivalent to the output of ϕ_3 of EHNN-MLP in Equation (2.21). Therefore, with ϕ_1, ϕ_2, and ϕ_3 approximating the functions in Equations (A.14), (A.15) and (A.16) respectively, the output of ϕ_3 of EHNN-MLP can approximate the output of weight application of EHNN (Equation (2.20)) to arbitrary precision. The overall error depend on the approximation error of MLPs (ϕ_1, ϕ_2, ϕ_3), and uniform bounds and continuity of the modeled functions. □

Proof of Theorem 5

Proof. We can reduce an EHNN-MLP layer to an AllDeepSets layer as follows. First, from Equation (2.21), we let $\phi_1(k, \mathbf{X}) = \phi'_1(\mathbf{X})$, $\phi_3(l, \mathbf{X}) = \phi'_3(\mathbf{X})$, and $\mathcal{B}(l) = 0$ to remove conditioning on hyperedge orders k and l. Then we let $\phi_2(\mathcal{I}, \mathbf{X}) = \mathbb{1}_{\mathcal{I}}\mathbf{X}$ to ablate the global interaction ($\mathcal{I} = 0$) and remove conditioning on $\mathcal{I} \geq 1$. By renaming ϕ'_1 to ϕ_1 and ϕ'_3 to ϕ_2, we get the AllDeepSets layer (Equation (2.24)). Yet, an AllDeepSets layer cannot reduce to an EHNN-MLP layer as it cannot model interactions between non-overlapping hyperedges ($\mathcal{I} = 0$). □

Proof of Theorem 6

Proof. We can reduce an EHNN-transformer layer to an AllSetTransformer layer as follows. First, from Equation (2.22), we let $\phi_1(k, \mathbf{X}) = \phi'_1(\mathbf{X})$, $\phi_3(l, \mathbf{X}) = \mathbf{X}$, and $\mathcal{B}(l) = 0$ to remove conditioning on hyperedge orders k and l. Then we let $\phi_2(\mathcal{I}, \mathbf{X}) = \mathbb{1}_{\mathcal{I}}\mathbf{X}$ to ablate the global interaction ($\mathcal{I} = 0$) and remove conditioning on $\mathcal{I} \geq 1$. Lastly, from Equation (2.23), we let $\mathcal{Q}(\mathcal{I}) = Q$ and $\mathcal{K}(\mathcal{I}, \mathbf{X}) = \mathcal{K}(\mathbf{X})$. By renaming ϕ'_1 to ϕ_1, we get the AllSetTransformer layer (Equation (2.25)). Yet, an AllSetTransformer cannot reduce to an EHNN-transformer as it cannot model interactions between non-overlapping hyperedges ($\mathcal{I} = 0$). □

A.3 Appendix of tokenized graph transformers

A.3.1 Preliminary

We begin by formally defining multihead self-attention and transformer. Our definition is equivalent to Vaswani et al. [2017], except we omit layer normalization for simplicity as in Yun et al. [2020], Hanin and Sellke [2017]. A multihead self-attention layer $\mathrm{MSA} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is defined as:

$$\boldsymbol{\alpha}^h = \mathrm{softmax}\left(\mathbf{X}w_h^Q(\mathbf{X}w_h^K)^\top/\sqrt{d_H}\right), \tag{A.17}$$

$$\mathrm{MSA}(\mathbf{X})_i = \sum_{h=1}^{H}\sum_{j=1}^{n}\boldsymbol{\alpha}_{ij}^h\mathbf{X}_j w_h^V w_h^O, \tag{A.18}$$

where H is number of heads, d_H is head size, and $w_h^Q, w_h^K \in \mathbb{R}^{d\times d_H}$, $w_h^V \in \mathbb{R}^{d\times d_v}$ $w_h^O \in \mathbb{R}^{d_v\times d}$. In our proofs, we use biases for query and key projections as in Yun et al. [2020] but omit them here for brevity. With multihead self-attention, a transformer layer $\mathcal{T} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is defined as:

$$\mathbf{H} = \mathbf{X} + \mathrm{MSA}(\mathbf{X}), \tag{A.19}$$

$$\mathcal{T}(\mathbf{X}) = \mathbf{H} + \mathrm{MLP}(\mathbf{H}), \tag{A.20}$$

where $\mathrm{MSA} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is a multihead self-attention layer with H heads of size d_H and $\mathrm{MLP} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is a tokenwise MLP with hidden dimension d_F.

We now complete the missing details in the definition of invariant graph networks (IGNs) and maximally expressive equivariant linear layers [Maron et al., 2019b,a] summarized in Definitions 4 and 5.

We first define *equivalence classes* and *basis tensors* mentioned briefly in Definition 5. The equivalence classes are defined upon a specific *equivalence relation* $\sim$ on the index space of higher-order tensors as:

Definition 8. *An order-l equivalence class $\gamma \in [n]^l/_\sim$ is an equivalence class of $[n]^l$ under the equivalence relation $\sim$, where the equivalence relation $\sim$ on multi-index space $[n]^l$ relates $\mathbf{i} \sim \mathbf{j}$ if and only if $(i_1, ..., i_l) = (\pi(j_1), ..., \pi(j_l))$ for some node permutation $\pi \in S_n$.*

We note that a multi-index $\mathbf{i}$ has the same permutation-invariant *equality pattern* to any $\mathbf{j}$ that satisfies $\mathbf{i} \sim \mathbf{j}$, *i.e.*, $\mathbf{i}_a = \mathbf{i}_b \Leftrightarrow \mathbf{j}_a = \mathbf{j}_b$ for all $a, b \in [k]$. Consequently, each equivalence class γ in Definition 8 is a distinct set of all order-l multi-indices having a specific equality pattern.

Now, for each equivalence class, we define the corresponding *basis tensor* as follows:

Definition 9. *An order-l basis tensor $\mathbf{B}^\gamma \in \mathbb{R}^{n^l}$ corresponding to an order-l equivalence class γ is a binary tensor defined as follows:*

$$\mathbf{B}^\gamma_{\mathbf{i}} = \begin{cases} 1 & \mathbf{i} \in \gamma \\ 0 & \textit{otherwise} \end{cases} \tag{A.21}$$

For a given l, it is known that there exist $\mathrm{bell}(l)$ order-l equivalence classes $\{\gamma_1, ..., \gamma_{\mathrm{bell}(l)}\} = [n]^l/_\sim$ regardless of n [Maron et al., 2019b]. This gives $\mathrm{bell}(l)$ order-l basis tensors $\mathbf{B}^{\gamma_1}, ..., \mathbf{B}^{\gamma_{\mathrm{bell}(l)}}$ accordingly. Thus, an equivariant linear layer $L_{k\to l}$ in Definition 5 has $\mathrm{bell}(l+k)$ weights and $\mathrm{bell}(l)$ biases.

Let us consider the first-order equivariant layer $L_{1\to 1}$ as an example. We have $\mathrm{bell}(2) = 2$ second-order equivalence classes γ_1 and γ_2 for the weight, with γ_1 the set of all (i_1, i_2) with $i_1 = i_2$ and γ_2 the set of all

(i_1, i_2) with $i_1 \neq i_2$. From Definition 9, their corresponding basis tensors are $\mathbf{B}^{\gamma_1} = \mathbf{I}$ and $\mathbf{B}^{\gamma_2} = \mathbf{1}\mathbf{1}^\top - \mathbf{I}$. Given a set of features $\mathbf{X} \in \mathbb{R}^{n \times d}$,

$$L_{1\to 1}(\mathbf{X}) = \mathbf{I}\mathbf{X}w_1 + (\mathbf{1}\mathbf{1}^\top - \mathbf{I})\mathbf{X}w_2 + \mathbf{1}b^\top,$$

with two weights $w_1, w_2 \in \mathbb{R}^{d \times d'}$, and a single bias $b \in \mathbb{R}^{d'}$. For graphs ($k = l = 2$), we have $\text{bell}(4) = 15$ weights and $\text{bell}(2) = 2$ biases.

A.3.2 Proofs

Proof of Lemma 1

To prove Lemma 1, we need to show that each basis tensor $\mathbf{B}^\mu$ (Equation (A.21)) in weights of equivariant linear layers (Equation (2.26)) can be approximated by the self-attention coefficient $\boldsymbol{\alpha}^h$ (Equation (A.17)) to arbitrary precision up to normalization if its input is augmented by node and type identifiers (Section 2.3.2).

From Definition 9, each entry of basis tensor $\mathbf{B}^\mu_{\mathbf{i},\mathbf{j}}$ encodes whether $(\mathbf{i}, \mathbf{j}) \in \mu$ or not. Here, our key idea is to break down the inclusion test $(\mathbf{i}, \mathbf{j}) \in \mu$ into equivalent but simpler Boolean tests that can be implemented in self-attention (Equation (A.18)) as dot product of $\mathbf{i}$-th query and $\mathbf{j}$-th key followed by softmax.

To achieve this, we show some supplementary lemmas. We start with Lemma 9, which comes from Kim et al. [2021a, Lemma 1] (we repeat the proof here for completeness).

Lemma 9. *For any order-$(l+k)$ equivalence class μ, the set of all $\mathbf{i} \in [n]^l$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ for some $\mathbf{j} \in [n]^k$ forms an order-l equivalence class. Likewise, the set of all $\mathbf{j}$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ for some $\mathbf{i}$ forms an order-k equivalence class.*

Proof. We only prove for $\mathbf{i}$ as proof for $\mathbf{j}$ is analogous. For some $(\mathbf{i}^1, \mathbf{j}^1) \in \mu$, let us denote the equivalence class of $\mathbf{i}^1$ as γ^l (*i.e.*, $\mathbf{i}^1 \in \gamma^l$). It is sufficient that we prove $\mathbf{i} \in \gamma^l \Leftrightarrow \exists \mathbf{j} : (\mathbf{i}, \mathbf{j}) \in \mu$.

($\Rightarrow$) For all $\mathbf{i} \in \gamma^l$, as $\mathbf{i}^1 \sim \mathbf{i}$, there exists some $\pi \in S_n$ such that $\mathbf{i} = \pi(\mathbf{i}^1)$ by definition. As π acts on multi-indices entry-wise, we have $\pi(\mathbf{i}^1, \mathbf{j}^1) = (\mathbf{i}, \pi(\mathbf{j}^1))$. As $\pi(\mathbf{i}^1, \mathbf{j}^1) \sim (\mathbf{i}^1, \mathbf{j}^1)$ holds by definition, we have $(\mathbf{i}, \pi(\mathbf{j}^1)) \sim (\mathbf{i}^1, \mathbf{j}^1)$, and thus $(\mathbf{i}, \pi(\mathbf{j}^1)) \in \mu$. Therefore, for all $\mathbf{i} \in \gamma^l$, by setting $\mathbf{j} = \pi(\mathbf{j}^1)$ we can always obtain $(\mathbf{i}, \mathbf{j}) \in \mu$.

($\Leftarrow$) For all $(\mathbf{i}, \mathbf{j}) \in \mu$, as $(\mathbf{i}, \mathbf{j}) \sim (\mathbf{i}^1, \mathbf{j}^1)$, there exists some $\pi \in S_n$ such that $(\mathbf{i}, \mathbf{j}) = \pi(\mathbf{i}^1, \mathbf{j}^1)$. This gives $\mathbf{i} = \pi(\mathbf{i}^1)$ and $\mathbf{j} = \pi(\mathbf{j}^1)$, leading to $\mathbf{i} \sim \mathbf{i}^1$ and therefore $\mathbf{i} \in \gamma^l$. □

Lemma 9 states that the equivalence classes γ^l of $\mathbf{i}$ and γ^k of $\mathbf{j}$ are identical for all $(\mathbf{i}, \mathbf{j}) \in \mu$. Based on this, we appropriately break down the test $(\mathbf{i}, \mathbf{j}) \in \mu$ into a combination of several simpler tests, in particular including $\mathbf{i} \in \gamma^l$ and $\mathbf{j} \in \gamma^k$:

Lemma 10. *For a given order-$(l+k)$ equivalence class μ, let γ^l and γ^k be equivalence classes of some $\mathbf{i}^1 \in [n]^l, \mathbf{j}^1 \in [n]^k$ respectively that satisfies $(\mathbf{i}^1, \mathbf{j}^1) \in \mu$. Then, for any $\mathbf{i} \in [n]^l$ and $\mathbf{j} \in [n]^k$, $(\mathbf{i}, \mathbf{j}) \in \mu$ holds if and only if the following conditions both hold:*

1. *$\mathbf{i} \in \gamma^l$ and $\mathbf{j} \in \gamma^k$*
2. *$\mathbf{i}_a = \mathbf{j}_b \Leftrightarrow \mathbf{i}^2_a = \mathbf{j}^2_b$ for all $a \in [l]$, $b \in [k]$, and $(\mathbf{i}^2, \mathbf{j}^2) \in \mu$*

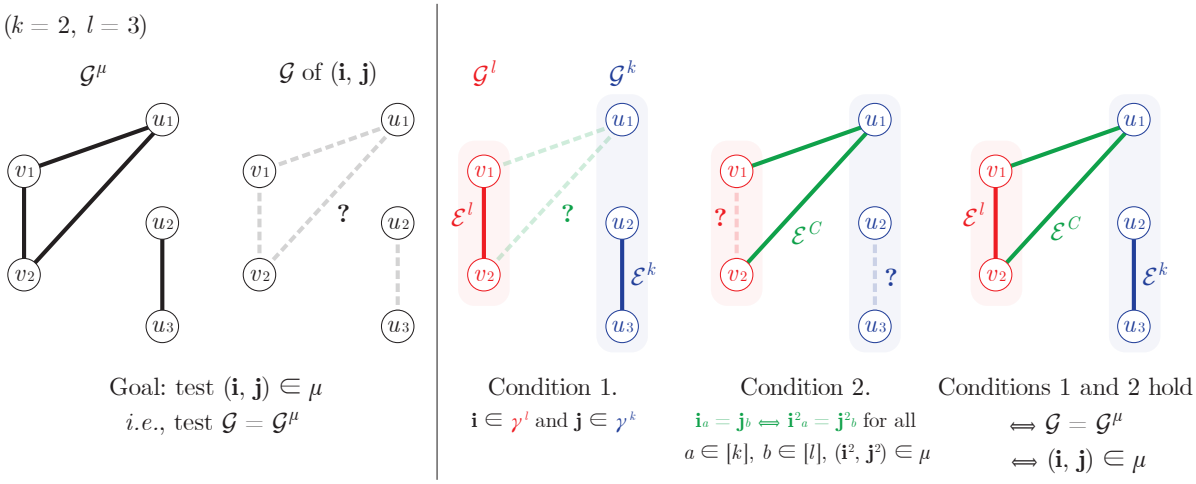


Figure A.3: An exemplary illustration of testing $(\mathbf{i},\mathbf{j}) \in \mu$ as a combination of simpler tests, based on equivalence classes μ, γ^l, and γ^k represented as graphs $\mathcal{G}^\mu$, $\mathcal{G}^l$, and $\mathcal{G}^k$, respectively.

Proof. ($\Rightarrow$) If $(\mathbf{i},\mathbf{j}) \in \mu$, from Lemma 9 it follows that $\mathbf{i} \in \gamma^l$ and $\mathbf{j} \in \gamma^k$. Also, as all $(\mathbf{i}^2,\mathbf{j}^2) \in \mu$ including $(\mathbf{i},\mathbf{j})$ have the same equality pattern, it follows that for all $a \in [l]$, $b \in [k]$, and $(\mathbf{i}^2,\mathbf{j}^2) \in \mu$, if $\mathbf{i}^2_a = \mathbf{j}^2_b$ then $\mathbf{i}_a = \mathbf{j}_b$ and if $\mathbf{i}^2_a \neq \mathbf{j}^2_b$ then $\mathbf{i}_a \neq \mathbf{j}_b$.

($\Leftarrow$) We show that the conditions specify that the equivalence class of $(\mathbf{i},\mathbf{j})$ is μ.

For this, it is convenient to represent an order-l equivalence class γ as an equivalent *undirected graph* $\mathcal{G} = (\mathcal{V},\mathcal{E})$ defined on vertex set $\mathcal{V} = \{v_1,...,v_l\}$ where the vertices v_a and v_b are connected, *i.e.*, $(v_a,v_b) \in \mathcal{E}$ if and only if the equivalence class γ specifies $\mathbf{i}_a = \mathbf{i}_b \forall \mathbf{i} \in \gamma$. Then, for some multi-index $\mathbf{i}' \in [n]^l$, the inclusion $\mathbf{i}' \in \gamma$ holds if and only if the equivalence class of $\mathbf{i}'$ is represented as $\mathcal{G}$.

Given this, let us represent the equivalence classes γ^l, γ^k, and μ as graphs $\mathcal{G}^l$, $\mathcal{G}^k$, and $\mathcal{G}^\mu$ respectively:

$$\begin{aligned}
&\mathcal{G}^l = (\mathcal{V}^l,\mathcal{E}^l) \text{ where } \mathcal{V}^l = \{v_1,...,v_l\},\\
&\mathcal{G}^k = (\mathcal{V}^k,\mathcal{E}^k) \text{ where } \mathcal{V}^k = \{u_1,...,u_k\},\\
&\mathcal{G}^\mu = (\mathcal{V}^\mu,\mathcal{E}^\mu) \text{ where } \mathcal{V}^\mu = \mathcal{V}^l \cup \mathcal{V}^k = \{v_1,...,v_l,u_1,...,u_k\}.
\end{aligned}$$

From the precondition that γ^l and γ^k are equivalence classes of $\mathbf{i}^1 \in [n]^l, \mathbf{j}^1 \in [n]^k$ that satisfies $(\mathbf{i}^1,\mathbf{j}^1) \in \mu$, we can see that $(v_a,v_b) \in \mathcal{E}^l \Leftrightarrow (v_a,v_b) \in \mathcal{E}^\mu$ and $(u_a,u_b) \in \mathcal{E}^k \Leftrightarrow (u_a,u_b) \in \mathcal{E}^\mu$. That is, if we consider $\mathcal{V}^l$ and $\mathcal{V}^k$ as a *graph cut* of $\mathcal{G}^\mu$ and write the cut-set (edges between $\mathcal{V}^l$ and $\mathcal{V}^k$) as $\mathcal{E}^C = \{(v_a,u_b)|(v_a,u_b) \in \mathcal{E}^\mu\}$, we obtain a partition $\{\mathcal{E}^l,\mathcal{E}^k,\mathcal{E}^C\}$ of the edge set $\mathcal{E}^\mu$.

We now move to the conditions.

Let us assume the first condition that $\mathbf{i} \in \gamma^l$ and $\mathbf{j} \in \gamma^k$, with the equivalence classes represented as $\mathcal{G}^l$ and $\mathcal{G}^k$, respectively. Now, let us consider the equivalence class of $(\mathbf{i},\mathbf{j})$ represented by (unknown) graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$. Considering $\mathcal{V}^k$ and $\mathcal{V}^l$ as a graph cut of $\mathcal{G}$, we can see that $\mathcal{E}$ is partitioned as $\{\mathcal{E}^l,\mathcal{E}^k,\mathcal{E}^D\}$ where $\mathcal{E}^D$ is the cut-set (edges between $\mathcal{V}^l$ and $\mathcal{V}^k$).

Let us also assume the second condition $\mathbf{i}_a = \mathbf{j}_b \Leftrightarrow \mathbf{i}^2_a = \mathbf{j}^2_b$ for all $a \in [l]$, $b \in [k]$, and $(\mathbf{i}^2,\mathbf{j}^2) \in \mu$. This directly implies that $e \in \mathcal{E}^C \Leftrightarrow e \in \mathcal{E}^D$, meaning that $\mathcal{E}^C = \mathcal{E}^D$. As a result, we see that $\mathcal{G}$ and $\mathcal{G}^\mu$ are identical graphs, and therefore the equivalence class of $(\mathbf{i},\mathbf{j})$ is μ and $(\mathbf{i},\mathbf{j}) \in \mu$ holds. □

In Figure A.3, we provide an exemplary illustration of testing $(\mathbf{i},\mathbf{j}) \in \mu$ following the last part of the proof.

With Lemma 10, we have a decomposition of $(\mathbf{i},\mathbf{j}) \in \mu$ into independent conditions on $\mathbf{i}$ and $\mathbf{j}$ combined with pairwise conditions between $\mathbf{i}$ and $\mathbf{j}$. In the following Definition 10 and Property 4, we encode these tests into a single *scoring function* that can be later implemented by self-attention.

Definition 10. *A scoring function $\delta(\mathbf{i},\mathbf{j};\mu,\epsilon)$ is a map that, given an order-$(l+k)$ equivalence class μ and $\epsilon > 0$, takes multi-indices $\mathbf{i} \in [n]^l, \mathbf{j} \in [n]^k$ and gives the following:*

$$\delta(\mathbf{i},\mathbf{j};\mu,\epsilon) = \mathbb{1}_{\mathbf{i}\in\gamma^l} + (1-\epsilon)\mathbb{1}_{\mathbf{i}\notin\gamma^l} + \mathbb{1}_{\mathbf{j}\in\gamma^k} + (1-\epsilon)\mathbb{1}_{\mathbf{j}\notin\gamma^k} + \sum_{a\in[l]}\sum_{b\in[k]} \mathrm{sgn}(a,b)\mathbb{1}_{\mathbf{i}_a=\mathbf{j}_b}, \tag{A.22}$$

where $\mathbb{1}$ is indicator, γ^l and γ^k are equivalence classes of $\mathbf{i}^1 \in [n]^l, \mathbf{j}^1 \in [n]^k$ such that $(\mathbf{i}^1,\mathbf{j}^1) \in \mu$, and the sign function $\mathrm{sgn}(\cdot,\cdot)$ is defined as follows:

$$\mathrm{sgn}(a,b) = \begin{cases} +1 & \mathbf{i}^2_a = \mathbf{j}^2_b \forall (\mathbf{i}^2,\mathbf{j}^2) \in \mu \\ -1 & \mathbf{i}^2_a \neq \mathbf{j}^2_b \forall (\mathbf{i}^2,\mathbf{j}^2) \in \mu \end{cases}. \tag{A.23}$$

An important property of the scoring function $\delta(\mathbf{i},\mathbf{j};\mu)$ is that it gives the maximum possible value if and only if the input satisfies $(\mathbf{i},\mathbf{j}) \in \mu$, as shown in the below Property 4.

Property 4. *For given order-$(l+k)$ equivalence class μ and positive real number $\epsilon > 0$, for any $\mathbf{i} \in [n]^l$ and $\mathbf{j} \in [n]^k$, $(\mathbf{i},\mathbf{j}) \in \mu$ holds if and only if the scoring function $\delta(\mathbf{i},\mathbf{j};\mu,\epsilon)$ (Equation (A.22)) outputs the maximum possible value.*

Proof. As shown in Lemma 10, $(\mathbf{i},\mathbf{j}) \in \mu$ holds if and only if the following two conditions are met.

1. $\mathbf{i} \in \gamma^l$ and $\mathbf{j} \in \gamma^k$
2. $\mathbf{i}_a = \mathbf{j}_b \Leftrightarrow \mathbf{i}^2_a = \mathbf{j}^2_b$ for all $a \in [l]$, $b \in [k]$, and $(\mathbf{i}^2,\mathbf{j}^2) \in \mu$

When both conditions are satisfied, in Equation (A.22), we always have $\mathbb{1}_{\mathbf{i}\in\gamma^l} + (1-\epsilon)\mathbb{1}_{\mathbf{i}\notin\gamma^l} = 1$ and $\mathbb{1}_{\mathbf{j}\in\gamma^k} + (1-\epsilon)\mathbb{1}_{\mathbf{j}\notin\gamma^k} = 1$. We also have $\mathbb{1}_{\mathbf{i}_a=\mathbf{j}_b} = 1$ for $\mathrm{sgn}(a,b) = 1$ and $\mathbb{1}_{\mathbf{i}_a=\mathbf{j}_b} = 0$ for $\mathrm{sgn}(a,b) = -1$ for all $a \in [l], b \in [k]$. As a result, Equation (A.22) gives a constant output for all $(\mathbf{i},\mathbf{j}) \in \mu$.

On the other hand, if given $(\mathbf{i},\mathbf{j})$ violates any of the conditions (thus $(\mathbf{i},\mathbf{j}) \notin \mu$), we either have $\mathbb{1}_{\mathbf{i}\in\gamma^l} + (1-\epsilon)\mathbb{1}_{\mathbf{i}\notin\gamma^l} = (1-\epsilon)$, or $\mathbb{1}_{\mathbf{j}\in\gamma^k} + (1-\epsilon)\mathbb{1}_{\mathbf{j}\notin\gamma^k} = (1-\epsilon)$, or $\mathbb{1}_{\mathbf{i}_a=\mathbf{j}_b} = 0$ for $\mathrm{sgn}(a,b) = 1$ or $\mathbb{1}_{\mathbf{i}_a=\mathbf{j}_b} = 1$ for $\mathrm{sgn}(a,b) = -1$ for some $a \in [l], b \in [k]$. *Any* of these violations decrements the output of Equation (A.22) by a positive (1 or ϵ), resulting in a non-maximum output.

Thus, the scoring function $\delta(\mathbf{i},\mathbf{j};\mu,\epsilon)$ gives the maximum possible output if and only if $(\mathbf{i},\mathbf{j}) \in \mu$. □

Now, we prove Lemma 1.

Lemma 1. *For all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and their augmentation $\mathbf{X}^{in}$, self-attention coefficients $\boldsymbol{\alpha}^h$ (Equation (2.27)) computed with $\mathbf{X}^{in} w^{in}$ can approximate any basis tensor $\mathbf{B}^\mu \in \mathbb{R}^{n^{2k}}$ of order-k equivariant linear layer $L_{k\to k}$ (Definition 5) to arbitrary precision up to normalization.*

Proof. Let us first recall the node and type identifiers (Section 2.3.2) for order-k tensors $\mathbf{X} \in \mathbb{R}^{n^k \times d}$. Node identifier $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ is an orthonormal matrix with n rows, and type identifier is a trainable matrix $\mathbf{E} \in \mathbb{R}^{\mathrm{bell}(k) \times d_e}$ with $\mathrm{bell}(k)$ rows $\mathbf{E}^{\gamma_1}, ..., \mathbf{E}^{\gamma_{\mathrm{bell}(k)}}$, each designated for an order-k equivalence class γ. For each multi-index $\mathbf{i} = (i_1, ..., i_k) \in [n]^k$, we augment the corresponding input tensor entry as $[\mathbf{X}_\mathbf{i}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^\mathbf{i}}]$ where $\mathbf{i} \in \gamma^\mathbf{i}$, obtaining the augmented order-k tensor $\mathbf{X}^{in} \in \mathbb{R}^{n^k \times (d+kd_p+d_e)}$. We use a trainable projection $w^{in} \in \mathbb{R}^{(d+kd_p+d_e)\times d_\mathcal{T}}$ to map them to a hidden dimension $d_\mathcal{T}$.

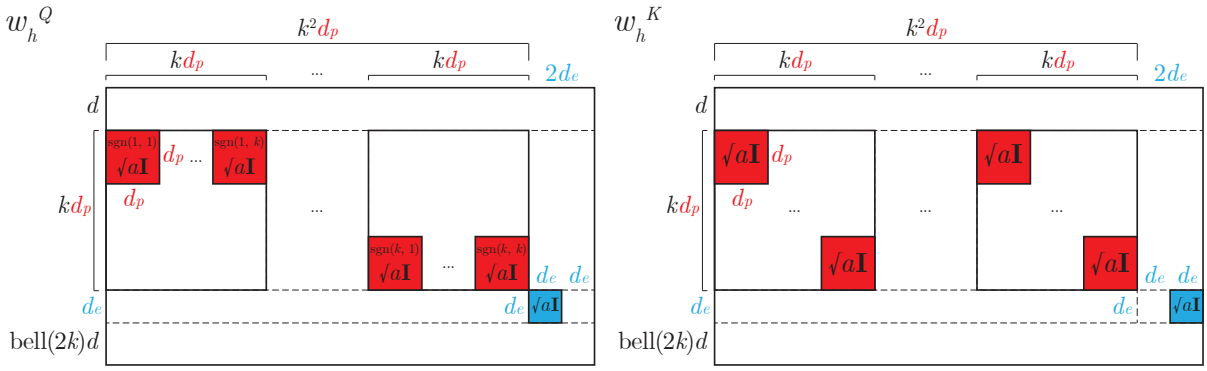


Figure A.4: Query and key projection matrices w_h^Q, w_h^K (Equations (A.24) and (A.25)). Uncolored cells are zeros.

We now use self-attention on $\mathbf{X}^{in}w^{in}$ to perform an accurate approximation of the equivariant basis. Specifically, we use each self-attention matrix $\boldsymbol{\alpha}^h$ (Equation (A.17)) to approximate each basis tensor $\mathbf{B}^{\mu_h}$ of $L_{k\to k}$ (Equation (2.26)) to arbitrary precision up to normalization.

Let us take $d_{\mathcal{T}} = (d + kd_p + d_e) + \text{bell}(2k)d$, putting bell$(2k)d$ extra channels on top of channels of the augmented input $\mathbf{X}^{in}$. We now let $w^{in} = [\mathbf{I}, \mathbf{0}]$, where $\mathbf{I} \in \mathbb{R}^{(d+kd_p+d_e)\times(d+kd_p+d_e)}$ is an identity matrix and $\mathbf{0} \in \mathbb{R}^{(d+kd_p+d_e)\times(d_{\mathcal{T}}-(d+kd_p+d_e))}$ is a matrix filled with zeros. With this, $\mathbf{X}' = \mathbf{X}^{in}w^{in}$ simply contains $\mathbf{X}^{in}$ in the first $(d + kd_p + d_e)$ channels and zeros in the rest.

Now we pass $\mathbf{X}'$ to the self-attention layer in Equation (A.17), where each self-attention matrix is given as $\boldsymbol{\alpha}^h = \text{softmax}((\mathbf{X}'w_h^Q + b_h^Q)(\mathbf{X}'w_h^K + b_h^K)^\top/\sqrt{d_H})$. The key idea is to set query and key projection parameters $w_h^Q, w_h^K \in \mathbb{R}^{d_{\mathcal{T}}\times d_H}$ and $b_h^Q, b_h^K \in \mathbb{R}^{d_H}$ appropriately so that the self-attention matrix $\boldsymbol{\alpha}^h$ approximates a given basis tensor $\mathbf{B}^\mu$ corresponding to an order-$2k$ equivalence class μ. Let γ^Q and γ^K be equivalence classes of some $\mathbf{i}^1, \mathbf{j}^1 \in [n]^k$ respectively that satisfy $(\mathbf{i}^1, \mathbf{j}^1) \in \mu$ (see Lemma 10). We set head dimension $d_H = k^2d_p + 2d_e$ and set $w_h^Q, w_h^K, b_h^Q, b_h^K$ as follows:

$$(w_h^Q)_{ij} = \begin{cases} \text{sgn}(s,r)\sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \le I + d_p & \text{for} \quad I = d + (s-1)d_p, \\ J < j \le J + d_p & \text{for} \quad J = (s-1)kd_p + (r-1)d_p, \\ \text{for all } s, r \in [k] \end{cases} \\ \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \le I + d_e & \text{for} \quad I = d + kd_p, \\ J < j \le J + d_e & \text{for} \quad J = k^2d_p, \end{cases} \\ 0 & \text{otherwise} \end{cases}, \tag{A.24}$$

$$(w_h^K)_{ij} = \begin{cases} \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \le I + d_p & \text{for} \quad I = d + (r-1)d_p, \\ J < j \le J + d_p & \text{for} \quad J = (s-1)kd_p + (r-1)d_p, \\ \text{for all } s, r \in [k] \end{cases} \\ \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \le I + d_e & \text{for} \quad I = d + kd_p, \\ J < j \le J + d_e & \text{for} \quad J = k^2d_p + d_e, \end{cases} \\ 0 & \text{otherwise} \end{cases}, \tag{A.25}$$

$$(b_h^Q)_j = \begin{cases} \sqrt{a}\mathbf{E}^{\gamma^K}_{j-J} & J < j \le J + d_e \quad \text{for} \quad J = k^2d_p \\ 0 & \text{otherwise} \end{cases},$$

$$(b_h^K)_j = \begin{cases} \sqrt{a}\mathbf{E}^{\gamma^Q}_{j-J} & J < j \le J + d_e \quad \text{for} \quad J = k^2d_p + d_e \\ 0 & \text{otherwise} \end{cases},$$

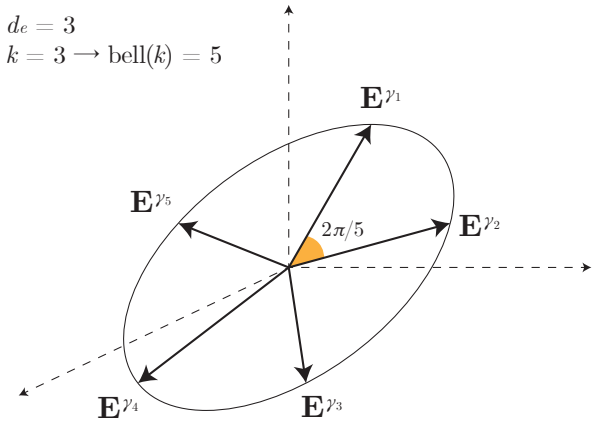


Figure A.5: $k = 3$ case example of $\text{bell}(k) = 5$ type identifiers embedded in $d_e = 3$ dimensional space.

where $a > 0$ is a positive real, $\mathbf{I}$ is an identity matrix, and $\text{sgn}(\cdot, \cdot)$ is the sign function defined in Equation (A.23) (Definition 10). In Figure A.4 we provide an illustration of the query and key weights w_h^Q, w_h^K.

With the parameters, **i**-th query and **j**-th key entries are computed as follows:

$$\mathbf{X}_\mathbf{i}' w_h^Q + b_h^Q = \sqrt{a}[[\text{sgn}(1,1)\mathbf{P}_{i_1}, ..., \text{sgn}(1,k)\mathbf{P}_{i_1}], ..., [\text{sgn}(k,1)\mathbf{P}_{i_k}, ..., \text{sgn}(k,k)\mathbf{P}_{i_k}], \mathbf{E}^{\gamma^\mathbf{i}}, \mathbf{E}^{\gamma^K}],$$
$$\mathbf{X}_\mathbf{j}' w_h^K + b_h^K = \sqrt{a}[\overbrace{[\mathbf{P}_{j_1}, ..., \mathbf{P}_{j_k}], ..., [\mathbf{P}_{j_1}, ..., \mathbf{P}_{j_k}]}^{k \text{ repeats}}, \mathbf{E}^{\gamma^Q}, \mathbf{E}^{\gamma^\mathbf{j}}].$$

Then, scaled pairwise dot product of query and key is given as follows:

$$\frac{(\mathbf{X}_\mathbf{i}' w_h^Q + b_h^Q)^\top (\mathbf{X}_\mathbf{j}' w_h^K + b_h^K)}{\sqrt{d_H}} = \frac{a}{\sqrt{d_H}} \left((\mathbf{E}^{\gamma^\mathbf{i}})^\top \mathbf{E}^{\gamma^Q} + (\mathbf{E}^{\gamma^\mathbf{j}})^\top \mathbf{E}^{\gamma^K} + \sum_{a \in [k]} \sum_{b \in [k]} \text{sgn}(a,b) \mathbf{P}_{i_a}^\top \mathbf{P}_{j_b} \right). \quad \text{(A.26)}$$

We refer to the scaled dot product in Equation (A.26) as the *unnormalized* attention coefficient $\tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h$.

We now let the type identifiers $\mathbf{E}^{\gamma_1}, ..., \mathbf{E}^{\gamma_{\text{bell}(k)}}$ be radially equispaced unit vectors on any two-dimensional subspace (Figure A.5). This guarantees that any pair of type identifiers $\mathbf{E}^{\gamma_1}, \mathbf{E}^{\gamma_2}$ with $\gamma_1 \neq \gamma_2$ have dot product $(\mathbf{E}^{\gamma_1})^\top \mathbf{E}^{\gamma_2} \leq \cos(2\pi/\text{bell}(k))$. By setting $\epsilon = 1 - \cos(2\pi/\text{bell}(k)) > 0$, this can be equivalently written as $(\mathbf{E}^{\gamma_1})^\top \mathbf{E}^{\gamma_2} \leq 1 - \epsilon$. We additionally note that $(\mathbf{E}^{\gamma^\mathbf{i}})^\top \mathbf{E}^{\gamma^Q} = 1$ if and only if $\mathbf{i} \in \gamma^Q$ because $\gamma^\mathbf{i} = \gamma^Q \Leftrightarrow \mathbf{i} \in \gamma^Q$.

Combining the above, Equations (A.26) and (A.22), we have the following:

$$\tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h = \frac{a}{\sqrt{d_H}} \delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon) \quad \text{if } (\mathbf{i}, \mathbf{j}) \in \mu, \qquad \tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h \leq \frac{a}{\sqrt{d_H}} \delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon) \quad \text{otherwise}, \quad \text{(A.27)}$$

where $\epsilon = 1 - \cos(2\pi/\text{bell}(k))$ and $\delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon)$ is the scoring function in Equation (A.22) (Definition 10).

For a given query index $\mathbf{i}$, let us assume there exists at least one key index $\mathbf{j}$ such that $(\mathbf{i}, \mathbf{j}) \in \mu$ [3]. From Property 4 and Equation (A.27), all keys $\mathbf{j}$ that give $(\mathbf{i}, \mathbf{j}) \in \mu$ hold the same maximum value $\tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h = \frac{a}{\sqrt{d_H}} \delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon)$, and any $(\mathbf{i}, \mathbf{j}) \notin \mu$ gives a value smaller at least by $\min(1, \epsilon) > 0$. Then, in softmax normalization, we send $a \to \infty$ by scaling up the query and key projection parameters. This pushes

[3] If such key index $\mathbf{j}$ does not exist, corresponding basis tensor entries are $\mathbf{B}_{\mathbf{i},\mathbf{j}}^\mu = 0 \forall \mathbf{j}$, and approximation target cannot be defined as normalizing denominator $\sum_\mathbf{j} \mathbf{B}_{\mathbf{i},\mathbf{j}}^\mu$ is 0. Thus we do not approximate for such $\mathbf{i}$, let attention row $\boldsymbol{\alpha}_{\mathbf{i},\cdot}$ have any finite values, and later silence their attention output by multiplying zero at MLP.

softmax arbitrarily close to the hardmax operator, leaving only the maximal entries leading to the following:

$$\boldsymbol{\alpha}^h_{\mathbf{i},\mathbf{j}} = \frac{\exp(\tilde{\boldsymbol{\alpha}}^h_{\mathbf{i},\mathbf{j}})}{\sum_{\mathbf{j}} \exp(\tilde{\boldsymbol{\alpha}}^h_{\mathbf{i},\mathbf{j}})} \to \frac{\mathbb{1}_{(\mathbf{i},\mathbf{j}\in\mu)}}{\sum_{\mathbf{j}} \mathbb{1}_{(\mathbf{i},\mathbf{j}\in\mu)}} = \frac{\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}} \text{ as } a \to \infty. \tag{A.28}$$

Thus, as shown in Equation (A.28), the attention coefficient $\boldsymbol{\alpha}^h$ can arbitrarily accurately approximate the normalized basis tensor $\mathbf{B}^{\mu}$ for given equivalence class μ. □

Proof of Theorem 7

Proof. We continue from the proof of Lemma 1 and assume that each attention matrix $\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^{\text{bell}(2k)}$ in Equation (A.17) head-wise approximates each normalized basis tensor $\mathbf{B}^{\mu_1}, ..., \mathbf{B}^{\mu_{\text{bell}(2k)}}$ respectively, *i.e.*, $\boldsymbol{\alpha}^h_{\mathbf{i},\mathbf{j}} = \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} / \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}$.[4]

Then, in Equation (A.18) we use $d_v = d$ and set $w_h^V \in \mathbb{R}^{d_\mathcal{T} \times d}$ to $w_h^V = [\mathbf{I}; \mathbf{0}]$, where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $\mathbf{0} \in \mathbb{R}^{(d_\mathcal{T}-d)\times d}$ is a matrix filled with zeros. With this, the value projection of each $\mathbf{i}$-th entry simply gives the original input features, $\mathbf{X}'_{\mathbf{i}} w_h^V = \mathbf{X}_{\mathbf{i}}$.

Then, we set output projections $w_h^O \in \mathbb{R}^{d\times d_\mathcal{T}}$ as follows:

$$(w_h^O)_{ij} = \begin{cases} (w_{\mu_h})_{i,j-J} & J < j \le J + d \text{ for } J = (d + kd_p + d_e) + (h-1)d \\ 0 & \text{otherwise} \end{cases},$$

where $w_{\mu_1}, ..., w_{\mu_{\text{bell}(2k)}} \in \mathbb{R}^{d\times d}$ are weight matrices of the given equivariant linear layer $L_{k\to k}$ in Equation (2.26) (Definition 5), each corresponding to equivalence classes $\mu_1, ..., \mu_{\text{bell}(2k)}$.

Then, output projection applied after value projection of each $\mathbf{i}$-th input entry gives the following:

$$\mathbf{X}'_{\mathbf{i}} w_h^V w_h^O = \mathbf{X}_{\mathbf{i}} w_h^O = [\mathbf{0}, \mathbf{0}_L, \mathbf{X}_{\mathbf{i}} w_{\mu_h}, \mathbf{0}_R],$$

where $\mathbf{0} \in \mathbb{R}^{(d+kd_p+d_e)}$, $\mathbf{0}_L \in \mathbb{R}^{(h-1)d}$, $\mathbf{0}_R \in \mathbb{R}^{d_\mathcal{T}-(d+kd_p+d_e)-hd}$ are zero vectors.

Based on the results, we compute the MSA with skip connection $\mathbf{H} = \mathbf{X}' + \text{MSA}(\mathbf{X}')$ (Equation (A.19)):

$$\begin{aligned}
\mathbf{H}_{\mathbf{i}} &= \mathbf{X}'_{\mathbf{i}} + \text{MSA}(\mathbf{X}')_{\mathbf{i}} \\
&= \left[\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_1\right] \\
&+ \left[\mathbf{0}_2, \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_1}, ..., \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_{\text{bell}(2k)}}\right] \\
&= \left[\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_1}, ..., \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_{\text{bell}(2k)}}\right],
\end{aligned}$$

where $\mathbf{0}_1 \in \mathbb{R}^{d_\mathcal{T}-(d+kd_p+d_e)}$, $\mathbf{0}_2 \in \mathbb{R}^{(d+kd_p+d_e)}$ are zero vectors.

We use feedforward MLP (Equation (A.20)) to denormalize and combine the result. Specifically, we make the elementwise MLP approximate following $f : \mathbb{R}^{d_\mathcal{T}} \to \mathbb{R}^{d_\mathcal{T}}$ based on universal approxima-

[4] we handle the case $\sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} = 0$ later separately as mentioned in footnote 3.

tion [Hanin and Sellke, 2017, Hornik et al., 1989]:

$$
\begin{aligned}
f(\mathbf{H_i})_j &= \begin{cases} -\mathbf{H}_{\mathbf{i},j} + \sum_{h\in[\text{bell}(2k)]} g(\mathbf{H_i})_h \mathbf{H}_{\mathbf{i},j+J} + b(\mathbf{H_i})_j & j \le d \\ 0 & d < j \le (d + kd_p + d_e) \\ -\mathbf{H}_{\mathbf{i},j} & j > (d + kd_p + d_e) \end{cases}, \\
g(\mathbf{H_i})_h &= \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}, \\
b(\mathbf{H_i})_j &= (b_{\gamma^{\mathbf{i}}})_j = (\sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma)_j,
\end{aligned}
\tag{A.29}
$$

where $J = (d + kd_p + d_e) + (h-1)d$, and $b_{\gamma_1}, ..., b_{\gamma_{\text{bell}(k)}}$ are biases of the given equivariant linear layer $L_{k\to k}$ with corresponding basis tensors $\mathbf{C}^{\gamma_1}, ..., \mathbf{C}^{\gamma_{\text{bell}(k)}}$ (Equation (2.26)).

Within the function f, the auxiliary function $g : \mathbb{R}^{d_\mathcal{T}} \to \mathbb{R}^{\text{bell}(2k)}$ computes head-wise attention denormalization factor[5] and $b : \mathbb{R}^{d_\mathcal{T}} \to \mathbb{R}^d$ computes bias. As n and k are fixed constants, the outputs $g(\mathbf{H_i})$ and $b(\mathbf{H_i})$ only depend on the equivalence class $\gamma^{\mathbf{i}}$ of $\mathbf{i}$. We note that the functions g and b can deduce the equivalence class from the input $\mathbf{H_i}$, by extracting the type identifier $\mathbf{E}^{\gamma^{\mathbf{i}}} = \mathbf{H}_{\mathbf{i}}^\top [\mathbf{0}_3, \mathbf{I}, \mathbf{0}_4]$ with $\mathbf{I} \in \mathbb{R}^{d_e \times d_e}$ an identity matrix and $\mathbf{0}_3 \in \mathbb{R}^{d+kd_p}$, $\mathbf{0}_4 \in \mathbb{R}^{\text{bell}(2k)d}$ zero matrices.

Based on the results, we compute the feedforward MLP with skip connection $\mathcal{T}(\mathbf{X}') = \mathbf{H} + \text{MLP}(\mathbf{H})$ (Equation (A.20)), which is the output of transformer layer $\mathcal{T}$:

$$
\begin{aligned}
\mathcal{T}(\mathbf{X}')_{\mathbf{i}} &= \mathbf{H_i} + \text{MLP}(\mathbf{H})_{\mathbf{i}} \\
&= \mathbf{H_i} + f(\mathbf{H_i}) \\
&= \left[\mathbf{X_i}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{S}^1_{\mathbf{i}}, ..., \mathbf{S}^{\text{bell}(2k)}_{\mathbf{i}}\right] \\
&+ \left[-\mathbf{X_i} + \sum_{h\in[\text{bell}(2k)]} \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} \mathbf{X_j} w_{\mu_h} + \sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma, \mathbf{0}_5, -\mathbf{S}^1_{\mathbf{i}}, ..., -\mathbf{S}^{\text{bell}(2k)}_{\mathbf{i}}\right], \\
&= \left[\sum_{h\in[\text{bell}(2k)]} \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} \mathbf{X_j} w_{\mu_h} + \sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right],
\end{aligned}
\tag{A.30}
$$

where we write $\mathbf{S}^h_{\mathbf{i}} = \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}} \mathbf{X_j} w_{\mu_h}$ and $\mathbf{0}_5 \in \mathbb{R}^{kd_p+d_e}, \mathbf{0}_6 \in \mathbb{R}^{(d_\mathcal{T}-(d+kd_p+d_e))}$ are zeros.

In Equation (A.30), note that the transformer layer $\mathcal{T}(\mathbf{X}')_{\mathbf{i}}$ only updates the first d channels of $\mathbf{X}'_{\mathbf{i}}$ from $\mathbf{X_i}$ to $\sum_\mu \sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{X_j} w_\mu + \sum_\gamma \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma$. Therefore, with a simple projection $w^{out} = [\mathbf{I}; \mathbf{0}] \in \mathbb{R}^{d_\mathcal{T} \times d}$ where $\mathbf{I} \in \mathbb{R}^{d\times d}$ is an identity matrix and $\mathbf{0} \in \mathbb{R}^{(d_\mathcal{T}-d)\times d}$ is a matrix filled with zeros, we can select the first d channels of the output and finally obtain $\mathcal{T}(\mathbf{X}')w^{out} = L_{k\to k}(\mathbf{X})$.

In conclusion, a transformer layer with bell$(2k)$ self-attention heads that operates on augmented $\mathbf{X}'$ can approximate any given $L_{k\to k}(\mathbf{X})$ to arbitrary precision. □

Proof of Theorem 8

Proof. We continue from the proof of Theorem 7, and assume that each transformer layer $\mathcal{T}$ can approximate a given $L_{k\to k}$ by only updating the first d channels.

[5]Note that the $g(\mathbf{H_i})_h$ gives 0 for all $\mathbf{i}$ that $\sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} = 0$, which automatically handles the corner case as discussed at footnotes 3 and 4.

Then, based on Theorem 7 we assume the following for each $t < T$:

$$\mathcal{T}^{(t)}(\mathbf{X}')_{\mathbf{i}} = \left[\sigma(L^{(t)}_{k\to k}(\mathbf{X}))_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right]$$

where $\mathbf{X}'_{\mathbf{i}} = [\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6]$. While Theorem 7 gives $L^{(t)}_{k\to k}(\mathbf{X})$ in the first d channels, we add elementwise activation $\sigma(\cdot)$ by absorbing it into the elementwise MLP in Equation (A.29). Then, leveraging the property that each transformer layer $\mathcal{T}^{(t)}$ only updates the first d channels, we stack $T-1$ transformer layers $\mathcal{T}^{(1)}$, ..., $\mathcal{T}^{(T-1)}$ and obtain the following:

$$\mathcal{T}^{(T-1)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}')_{\mathbf{i}} = \left[\sigma \circ L^{(T-1)}_{k\to k} \circ \sigma \circ ... \circ \sigma \circ L^{(1)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right]. \tag{A.31}$$

For the last layer $\mathcal{T}^{(T)}$, we follow the procedure in the proof of Theorem 7 to approximate $L^{(T)}_{k\to k}$, but slightly tweak Equation (A.29) so that elementwise MLP copies each output entry $L_{k\to k}(\mathbf{X})^{(T)}_{\mathbf{i}}$ in appropriate reserved channels. Specifically, we let the elementwise MLP approximate following f':

$$f'(\mathbf{H}_{\mathbf{i}})_j = \begin{cases} -\mathbf{H}_{\mathbf{i},j} & j \le D \\ -\mathbf{H}_{\mathbf{i},j} + \mathbf{C}^{\gamma_a}_{\mathbf{i}} \mathbf{F}_{\mathbf{i},j-(D+(a-1)d)} & D + (a-1)d < j \le D + ad \text{ for all } a \in [\text{bell}(k)] \\ -\mathbf{H}_{\mathbf{i},j} & D + \text{bell}(k)d < j \end{cases}, \tag{A.32}$$

where $D = (d + kd_p + d_e)$ and we abbreviate $\mathbf{F}_{\mathbf{i},j} = \sum_{h\in[\text{bell}(2k)]} g(\mathbf{H}_{\mathbf{i}})_h \mathbf{H}_{\mathbf{i},j+J} + b(\mathbf{H}_{\mathbf{i}})_j$ with J, g, b defined as same as in Equation (A.29). Recall that $\mathbf{C}^{\gamma_a}_{\mathbf{i}} = 1$ if and only if $\mathbf{i} \in \gamma_a$. Therefore, with Equation (A.32), we are simply duplicating each output entry $\mathbf{F}_{\mathbf{i}} = L^{(T)}_{k\to k}(\mathbf{X})_{\mathbf{i}}$ to spare channel indices reserved for the equivalence class of $\mathbf{i}$ (γ_a that $\mathbf{i} \in \gamma_a$).

With the choice of $\mathcal{T}^{(T)}$, the layer output $\mathcal{T}^{(T)}(\mathbf{X}') = \mathbf{H} + \text{MLP}(\mathbf{H})$ (Equation (A.20)) is computed as:

$$\begin{aligned} \mathcal{T}^{(T)}(\mathbf{X}')_{\mathbf{i}} &= \mathbf{H}_{\mathbf{i}} + \text{MLP}(\mathbf{H})_{\mathbf{i}} \\ &= \mathbf{H}_{\mathbf{i}} + f'(\mathbf{H}_{\mathbf{i}}) \\ &= \left[\mathbf{0}_7, \mathbf{C}^{\gamma_1}_{\mathbf{i}} L^{(T)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, ..., \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} L^{(T)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, \mathbf{0}_8\right], \end{aligned} \tag{A.33}$$

where $\mathbf{0}_7 \in \mathbb{R}^{(d+kd_p+d_e)}, \mathbf{0}_8 \in \mathbb{R}^{d_{\mathcal{T}}-(d+kd_p+d_e)-\text{bell}(k)d}$ are zero vectors.

Then, by applying $\mathcal{T}^{(T)}$ (Equation (A.33)) on top of $\mathcal{T}^{(T-1)} \circ ... \circ \mathcal{T}^{(1)}$ (Equation (A.31)), we obtain the following:

$$\mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}')_{\mathbf{i}} = \left[\mathbf{0}_7, \mathbf{C}^{\gamma_1}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, ..., \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, \mathbf{0}_8\right],$$

where we abbreviate $\mathbf{Y} = L^{(T)}_{k\to k} \circ \sigma \circ ... \circ \sigma \circ L^{(1)}_{k\to k}(\mathbf{X})$.

The remaining step is to utilize MLP $\circ$ sumpool to approximate $\text{MLP}_k \circ L_{k\to 0}$ that tops F_k. By sum-pooling over all indices $\mathbf{i}$, we obtain the following:

$$\text{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') = \left[\mathbf{0}_7, \sum_{\mathbf{i}} \mathbf{C}^{\gamma_1}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, ..., \sum_{\mathbf{i}} \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, \mathbf{0}_8\right]. \tag{A.34}$$

Now, we let the final MLP approximate the following function $f'' : \mathbb{R}^{d\tau} \to \mathbb{R}^d$:

$$f''(\mathbf{X}) = \mathrm{MLP}_k \left(\sum_{a \in [\mathrm{bell}(k)]} \mathbf{X}^a w_{\mu_a} + b_f \right) \text{ where } \mathbf{X}^a_j = \mathbf{X}_{D+(a-1)d+j} \text{ for } j \in [d],$$

where $w_{\mu_1}, ..., w_{\mu_{\mathrm{bell}(k)}} \in \mathbb{R}^{d \times d}$ and $b_f \in \mathbb{R}^d$ are the weights and bias of the given invariant linear layer $L_{k \to 0}$, and each $\mathbf{X}^a \in \mathbb{R}^d$ is a chunk that coincides with reserved channels in Equation (A.32). By plugging in the sum-pooled representation in Equation (A.34), we finally obtain the following:

$$\begin{aligned}
\mathrm{MLP} \circ \mathrm{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') &= f'' \circ \mathrm{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') \\
&= \mathrm{MLP}_k \left(\sum_{a \in [\mathrm{bell}(k)]} \sum_{\mathbf{i}} \mathbf{C}^{\gamma_a}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}} w_{\mu_a} + b_f \right) \\
&= \mathrm{MLP}_k \circ L_{k \to 0}(\mathbf{Y}) \\
&= \mathrm{MLP}_k \circ L_{k \to 0} \circ L^{(T)}_{k \to k} \circ \sigma \circ ... \circ \sigma \circ L^{(1)}_{k \to k}(\mathbf{X}) \\
&= F_k(\mathbf{X}),
\end{aligned}$$

where the last equality comes from Definition 4.

Taken together, we arrive at the conclusion that $\mathrm{MLP} \circ \mathrm{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}')$ can approximate $F_k(\mathbf{X})$ to arbitrary precision. □

A.4 Appendix of probabilistic symmetrization

A.4.1 Proofs

Proof of Theorem 9

Proof. We prove $\phi_{\theta,\omega}(\rho_1(g')\mathbf{x}) = \rho_2(g')\phi_{\theta,\omega}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ and $g' \in G$. From Equation (3.3), we have:

$$\phi_{\theta,\omega}(\rho_1(g')\mathbf{x}) = \mathbb{E}_{p_\omega(g|\rho_1(g')\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\rho_1(g')\mathbf{x})\right].$$

Let us introduce transformed random variable $h = g'^{-1}g \in G$ such that $g = g'h$. Since the distribution p_ω is G-equivariant, we can see that $p_\omega(g|\rho_1(g')\mathbf{x}) = p_\omega(g'^{-1}g|\rho_1(g'^{-1})\rho_1(g')\mathbf{x}) = p_\omega(g'^{-1}g|\mathbf{x}) = p_\omega(h|\mathbf{x})$. Thus, we can rewrite the above expectation with respect to h as follows:

$$\begin{aligned}
\phi_{\theta,\omega}(\rho_1(g')\mathbf{x}) &= \mathbb{E}_{p_\omega(h|\mathbf{x})}\left[\rho_2(g'h) f_\theta(\rho_1(g'h)^{-1}\rho_1(g')\mathbf{x})\right] \\
&= \mathbb{E}_{p_\omega(h|\mathbf{x})}\left[\rho_2(g')\rho_2(h) f_\theta(\rho_1(h)^{-1}\rho_1(g')^{-1}\rho_1(g')\mathbf{x})\right] \\
&= \rho_2(g')\mathbb{E}_{p_\omega(h|\mathbf{x})}\left[\rho_2(h) f_\theta(\rho_1(h)^{-1}\mathbf{x})\right] \\
&= \rho_2(g')\phi_{\theta,\omega}(\mathbf{x}),
\end{aligned}$$

showing the G-equivariance of $\phi_{\theta,\omega}$ for arbitrary f_θ. □

Proof of Theorem 10

Proof. The proof is inspired by universality proofs of Yarotsky [2022], Puny et al. [2022], Kaba et al. [2023]. Let $\psi : \mathcal{X} \to \mathcal{Y}$ be an arbitrary G-equivariant function. By equivariance of ψ, we have:

$$\begin{aligned}
\|\psi(\mathbf{x}) - \phi_{\theta,\omega}(\mathbf{x})\| &= \left\|\psi(\mathbf{x}) - \mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\
&= \left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\psi(\mathbf{x})\right] - \mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\
&= \left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g)\rho_2(g)^{-1}\psi(\mathbf{x})\right] - \mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\
&= \left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g)\psi(\rho_1(g)^{-1}\mathbf{x})\right] - \mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\
&= \left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\rho_2(g)\psi(\rho_1(g)^{-1}\mathbf{x}) - \rho_2(g) f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\|.
\end{aligned}$$

As $\mathcal{Y}$ is finite-dimensional, we can assume that the linear operators in $\mathrm{GL}(\mathcal{Y})$ are bounded and so is the induced operator norm of group representation $\|\rho_2(g)\|$ for all $g \in G$. Thus, we have:

$$\begin{aligned}
\|\psi(\mathbf{x}) - \phi_{\theta,\omega}(\mathbf{x})\| &\leq \max_{h\in G}\|\rho_2(h)\|\left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\psi(\rho_1(g)^{-1}\mathbf{x}) - f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\
&\leq c\left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\psi(\rho_1(g)^{-1}\mathbf{x}) - f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\|,
\end{aligned}$$

for some $c > 0$.

If f_θ is a universal approximator, for any compact set $\mathcal{K} \subseteq \mathcal{X}$ and any $\epsilon > 0$, there exists some θ such that $\|\psi(\mathbf{x}) - f_\theta(\mathbf{x})\| \leq \epsilon$ for all $\mathbf{x} \in \mathcal{K}$. Consider the set $\mathcal{K}_{\mathrm{sym}} = \cup_{g\in G}\rho_1(g)\mathcal{K}$ where $\rho_1(g)\mathcal{K}$ denotes the image of the set $\mathcal{K}$ under linear transformation by $\rho_1(g)$. We use the fact that $\mathcal{K}_{\mathrm{sym}}$ is also a compact set since it is the image of the compact set $G \times \mathcal{K}$ under continuous map $(g, \mathbf{x}) \mapsto \rho_1(g)\mathbf{x}$. As a consequence, for any compact set $\mathcal{K} \subseteq \mathcal{X}$ and any $\epsilon/c > 0$, there exists some θ such that $\max_{g\in G}\|\psi(\rho_1(g)\mathbf{x}) - f_\theta(\rho_1(g)\mathbf{x})\| \leq \epsilon/c$ for all $\mathbf{x} \in \mathcal{K}$. Since a group is closed under inverse, for any

compact set $\mathcal{K} \subseteq \mathcal{X}$ and any $\epsilon > 0$, there exists some θ such that:

$$\begin{aligned}\|\psi(\mathbf{x}) - \phi_{\theta,\omega}(\mathbf{x})\| &\leq c \left\|\mathbb{E}_{p_\omega(g|\mathbf{x})}\left[\psi(\rho_1(g)^{-1}\mathbf{x}) - f_\theta(\rho_1(g)^{-1}\mathbf{x})\right]\right\| \\ &\leq c \max_{g\in G} \left\|\psi(\rho_1(g)^{-1}\mathbf{x}) - f_\theta(\rho_1(g)^{-1}\mathbf{x})\right\| \\ &= \epsilon,\end{aligned}$$

for all $\mathbf{x} \in \mathcal{K}$, showing that $\phi_{\theta,\omega}$ is a universal approximator of G-equivariant functions. □

While we have assumed that the group G is compact in the proof, we conjecture that the results can be extended to non-compact groups if we make an alternative assumption that the distribution $p_\omega(g|\mathbf{x})$ is compactly supported for all $\mathbf{x} \in \mathcal{K}$. We leave proving this as a future work.

Proof of Theorem 11

Proof. We prove $p_\omega(g'g|\rho_1(g')\mathbf{x}) = p_\omega(g|\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ and $g, g' \in G$. In general, we are interested in obtaining a faithful representation ρ, i.e., such that $\rho(g)$ is distinct for each g. We can interpret the probability $p_\omega(g|\mathbf{x}, \boldsymbol{\epsilon})$ as a delta distribution centered at the group representation $\rho(g)$:

$$p_\omega(g|\mathbf{x}, \boldsymbol{\epsilon}) = \delta(\rho(g) = q_\omega(\mathbf{x}, \boldsymbol{\epsilon})).$$

To obtain $p_\omega(g|\mathbf{x})$, we marginalize over $p(\boldsymbol{\epsilon})$:

$$p_\omega(g|\mathbf{x}) = \int_{\boldsymbol{\epsilon}} p_\omega(g|\mathbf{x}, \boldsymbol{\epsilon}) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} = \int_{\boldsymbol{\epsilon}} \delta(\rho(g) = q_\omega(\mathbf{x}, \boldsymbol{\epsilon})) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}.$$

Let us consider $p_\omega(g'g|\rho_1(g')\mathbf{x})$:

$$p_\omega(g'g|\rho_1(g')\mathbf{x}) = \int_{\boldsymbol{\epsilon}} \delta(\rho(g'g) = q_\omega(\rho_1(g')\mathbf{x}, \boldsymbol{\epsilon})) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}.$$

Using the G-equivariance of q_ω, we have:

$$q_\omega(\rho_1(g')\mathbf{x}, \boldsymbol{\epsilon}) = \rho(g') q_\omega(\rho_1(g'^{-1})\rho_1(g')\mathbf{x}, \rho'(g'^{-1})\boldsymbol{\epsilon}) = \rho(g') q_\omega(\mathbf{x}, \rho'(g'^{-1})\boldsymbol{\epsilon})$$

which leads to the following:

$$\begin{aligned}p_\omega(g'g|\rho_1(g')\mathbf{x}) &= \int_{\boldsymbol{\epsilon}} \delta(\rho(g'g) = \rho(g') q_\omega(\mathbf{x}, \rho'(g'^{-1})\boldsymbol{\epsilon})) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \\ &= \int_{\boldsymbol{\epsilon}} \delta(\rho(g) = q_\omega(\mathbf{x}, \rho'(g'^{-1})\boldsymbol{\epsilon})) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon},\end{aligned}$$

where the second equality follows from invertibility of $\rho(g')$.

We now introduce a change of variables $\boldsymbol{\epsilon}' = \rho'(g'^{-1})\boldsymbol{\epsilon}$ that $\boldsymbol{\epsilon} = \rho'(g')\boldsymbol{\epsilon}'$:

$$p_\omega(g'g|\rho_1(g')\mathbf{x}) = \int_{\boldsymbol{\epsilon}'} \delta(\rho(g) = q_\omega(\mathbf{x}, \boldsymbol{\epsilon}')) p(\rho'(g')\boldsymbol{\epsilon}') \frac{1}{|\det \rho'(g'^{-1})|} d\boldsymbol{\epsilon}'.$$

With $|\det \rho'(g'^{-1})| = 1$, and G invariance of $p(\boldsymbol{\epsilon})$ which gives $p(\rho'(g')\boldsymbol{\epsilon}') = p(\boldsymbol{\epsilon}')$, we get:

$$p_\omega(g'g|\rho_1(g')\mathbf{x}) = \int_{\boldsymbol{\epsilon}'} \delta(\rho(g) = q_\omega(\mathbf{x}, \boldsymbol{\epsilon}')) p(\boldsymbol{\epsilon}') d\boldsymbol{\epsilon}' = p_\omega(g|\mathbf{x}),$$

showing the G-equivariance of $p_\omega(g|\mathbf{x})$. □

Proof of validity of implemented equivariant distributions p_ω

We formally show G-equivariance of the implemented distributions $p_\omega(g|\mathbf{x})$ presented in Section 3.1.1. All implementations have a form of noise-outsourced function $q_\omega : (\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \rho(g)$ using distribution $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ and map q_ω which is composed of G-equivariant neural network and postprocessing to $\rho(g)$. From Theorem 11, for G-equivariance of $p_\omega(g|\mathbf{x})$, it is sufficient to show G invariance of $p(\boldsymbol{\epsilon})$ under a representation ρ' such that $|\det \rho'(g)| = 1$ along with G-equivariance of q_ω, which we show below.

Symmetric Group S_n We recall that $p_\omega(g|\mathbf{x})$ for the symmetric group S_n is implemented as below:

1. Sample node-level noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times d}$ from i.i.d. uniform $\mathrm{Unif}[0, \eta]$.
2. Use a GNN to obtain node-level scalar features $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{Z} \in \mathbb{R}^n$.
3. Assuming $\mathbf{Z}$ is tie-free, use argsort [Winter et al., 2021] to obtain group representation $\mathbf{Z} \mapsto \mathbf{P}_g = \rho(g)$.

$$\mathbf{P}_g = \mathrm{eq}(\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{Z})^\top), \tag{A.35}$$

where eq denotes elementwise equality indicator.

We now show the following:

Proposition 18. *The proposed distribution $p_\omega(g|\mathbf{x})$ for the symmetric group S_n is equivariant.*

Proof. Given $p(\boldsymbol{\epsilon})$ is elementwise i.i.d., it is S_n-invariant under the base representation $\rho'(g) = \mathbf{P}_g$ which satisfies $|\det \mathbf{P}_g| = 1$ from orthogonality. As a GNN is S_n-equivariant, we only need to show S_n-equivariance of argsort : $\mathbf{Z} \mapsto \mathbf{P}_g$. This can be shown by transforming $\mathbf{Z}$ with any permutation matrix $\mathbf{P}_{g'}$. Since sort operator and any row replicated matrices are invariant to $\mathbf{P}_{g'}$, we have:

$$\begin{aligned}\mathrm{eq}(\mathbf{P}_{g'}\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{P}_{g'}\mathbf{Z})^\top) &= \mathrm{eq}(\mathbf{P}_{g'}\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{Z})^\top) \\ &= \mathrm{eq}(\mathbf{P}_{g'}\mathbf{Z}\mathbf{1}^\top, \mathbf{P}_{g'}\mathbf{1}\mathrm{sort}(\mathbf{Z})^\top).\end{aligned}$$

Since eq commutes with $\mathbf{P}_{g'}$, we have:

$$\begin{aligned}\mathrm{eq}(\mathbf{P}_{g'}\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{P}_{g'}\mathbf{Z})^\top) &= \mathrm{eq}(\mathbf{P}_{g'}\mathbf{Z}\mathbf{1}^\top, \mathbf{P}_{g'}\mathbf{1}\mathrm{sort}(\mathbf{Z})^\top) \\ &= \mathbf{P}_{g'}\mathrm{eq}(\mathbf{Z}\mathbf{1}^\top, \mathbf{1}\mathrm{sort}(\mathbf{Z})^\top) \\ &= \mathbf{P}_{g'}\mathbf{P}_g,\end{aligned}$$

showing that argsort is S_n-equivariant, i.e., it maps $\mathbf{P}_{g'}\mathbf{Z} \mapsto \mathbf{P}_{g'}\mathbf{P}_g$ for all $\mathbf{P}_{g'} \in \mathrm{S}_n$. Combining the above, by Theorem 11, the distribution $p_\omega(g|\mathbf{x})$ is S_n-equivariant. □

Orthogonal Group $\mathrm{O}(n)$**,** $\mathrm{SO}(n)$ We recall that $p_\omega(g|\mathbf{x})$ for the orthogonal group $\mathrm{O}(n)$ or special orthogonal group $\mathrm{SO}(n)$ is implemented as follows:

1. Sample noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times d}$ from i.i.d. normal $\mathcal{N}(0, \eta^2)$.
2. Use an $\mathrm{O}(n)/\mathrm{SO}(n)$ equivariant neural network to obtain n features $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto \mathbf{Z} \in \mathbb{R}^{n \times n}$.

3. Assuming $\mathbf{Z}$ is full-rank, use Gram-Schmidt process [Kaba et al., 2023] to obtain an orthogonal matrix $\mathbf{Z} \mapsto \mathbf{Q}$.

4. For the $\mathrm{O}(n)$ group, use the obtained matrix as group representation $\mathbf{Q} = \mathbf{Q}_g = \rho(g)$.

5. For the $\mathrm{SO}(n)$ group, use below scale operator to obtain group representation $\mathbf{Q} \mapsto \mathbf{Q}_g^+ = \rho(g)$.

$$\text{scale} : \left[\begin{array}{c|c|c} & & \\ \mathbf{Q}_1 & \ldots & \mathbf{Q}_n \\ & & \end{array}\right] \mapsto \left[\begin{array}{c|c|c} & & \\ \det(\mathbf{Q}) \cdot \mathbf{Q}_1 & \ldots & \mathbf{Q}_n \\ & & \end{array}\right].$$

We now show the following:

Proposition 19. *The proposed distribution $p_\omega(g|\mathbf{x})$ for the orthogonal group $\mathrm{O}(n)$ is equivariant.*

Proof. Without loss of generality, let us omit the scale η for brevity, which gives that each column $\boldsymbol{\epsilon}_i \in \mathbb{R}^n$ of the noise $\boldsymbol{\epsilon}$ independently follows multivariate standard normal $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \mathbf{I}_n)$. Then, the density $p(\boldsymbol{\epsilon}_i) = (2\pi)^{-n/2} \exp\left(-\|\boldsymbol{\epsilon}_i\|_2^2/2\right)$ is invariant under orthogonal transformation $\mathbf{Q}$ since $\|\mathbf{Q}\boldsymbol{\epsilon}_i\|_2^2 = (\mathbf{Q}\boldsymbol{\epsilon}_i)^\top \mathbf{Q}\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}_i^\top \mathbf{Q}^\top \mathbf{Q}\boldsymbol{\epsilon}_i = \boldsymbol{\epsilon}_i^\top \boldsymbol{\epsilon}_i = \|\boldsymbol{\epsilon}_i\|_2^2$. Therefore, the distribution $p(\boldsymbol{\epsilon})$ is invariant under the base representation $\rho'(g) = \mathbf{Q}_g$ which satisfies $|\det \rho'(g)| = 1$ from orthogonality. As we use an equivariant neural network to obtain $\mathbf{Z}$, and Gram-Schmidt procedure $\mathbf{Z} \mapsto \mathbf{Q}_g$ is $\mathrm{O}(n)$-equivariant [Kaba et al., 2023, Theorem 5], by Theorem 11, the distribution $p_\omega(g|\mathbf{x})$ is $\mathrm{O}(n)$-equivariant. □

Proposition 20. *The proposed distribution $p_\omega(g|\mathbf{x})$ for special orthogonal group $\mathrm{SO}(n)$ is equivariant.*

Proof. From the proof of Proposition 19, it follows that the distribution $p(\boldsymbol{\epsilon})$ is invariant under the base representation $\rho'(g) = \mathbf{Q}_g^+$ which satisfies $|\det \rho'(g)| = 1$ due to orthogonality. As we use an equivariant neural network to obtain $\mathbf{Z}$, and Gram-Schmidt procedure $\mathbf{Z} \mapsto \mathbf{Q}$ has $\mathrm{O}(n)$-equivariance which implies $\mathrm{SO}(n)$-equivariance because of $\mathrm{SO}(n) \leq \mathrm{O}(n)$, we only need to show $\mathrm{SO}(n)$-equivariance of $\text{scale} : \mathbf{Q} \mapsto \mathbf{Q}_g^+$. This can be done by transforming $\mathbf{Q}$ with an orthogonal $\mathbf{Q}_{g'}^+$ of determinant $+1$. Since $\det(\mathbf{Q}_{g'}^+ \mathbf{Q}) = \det(\mathbf{Q}_{g'}^+)\det(\mathbf{Q}) = \det(\mathbf{Q})$, we have the following:

$$\text{scale}(\mathbf{Q}_{g'}^+ \mathbf{Q}) = \left[\begin{array}{c|c|c} & & \\ \det(\mathbf{Q}_{g'}^+ \mathbf{Q}) \cdot (\mathbf{Q}_{g'}^+ \mathbf{Q})_1 & \ldots & (\mathbf{Q}_{g'}^+ \mathbf{Q})_n \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ \det(\mathbf{Q}) \cdot (\mathbf{Q}_{g'}^+ \mathbf{Q})_1 & \ldots & (\mathbf{Q}_{g'}^+ \mathbf{Q})_n \\ & & \end{array}\right].$$

Also, scaling the first column of the product $\mathbf{Q}_{g'}^+ \mathbf{Q}$ with $\det(\mathbf{Q})$ is equivalent to scaling the first column of $\mathbf{Q}$ with $\det(\mathbf{Q})$ then computing the product since $(\mathbf{Q}_{g'}^+ \mathbf{Q})_{ij} = \sum_k \mathbf{Q}_{g'ik}^+ \mathbf{Q}_{kj}$. This gives:

$$\text{scale}(\mathbf{Q}_{g'}^+ \mathbf{Q}) = \mathbf{Q}_{g'}^+ \left[\begin{array}{c|c|c} & & \\ \det(\mathbf{Q}) \cdot \mathbf{Q}_1 & \ldots & \mathbf{Q}_n \\ & & \end{array}\right] = \mathbf{Q}_{g'}^+ \text{scale}(\mathbf{Q}),$$

showing that scale operator is $\mathrm{SO}(n)$-equivariant. We also note that $\text{scale}(\mathbf{Q})$ gives orthogonal matrix of determinant $+1$, as it returns $\mathbf{Q}$ if $\det(\mathbf{Q}) = +1$, otherwise ($\det(\mathbf{Q}) = -1$ since $\mathbf{Q}$ is orthogonal) scales the first column by -1 which flips determinant to $+1$ while not affecting orthogonality. Combining the above, by Theorem 11, the distribution $p_\omega(g|\mathbf{x})$ is $\mathrm{SO}(n)$-equivariant. □

Euclidean Group $\mathrm{E}(n)$**,** $\mathrm{SE}(n)$ We recall that, unlike the other groups, we handle the Euclidean group $\mathrm{E}(n)$ and special Euclidean group $\mathrm{SE}(n)$ at symmetrization level as the translation component $\mathrm{T}(n)$ in $\mathrm{E}(n) = \mathrm{O}(n) \ltimes \mathrm{T}(n)$ and $\mathrm{SE}(n) = \mathrm{SO}(n) \ltimes \mathrm{T}(n)$ is non-compact. This is done as follows:

$$\phi_{\theta,\omega}(\mathbf{x}) = \mathbb{E}_{p_\omega(g|\mathbf{x}-\bar{\mathbf{x}}\mathbf{1}^\top)}\left[\bar{\mathbf{x}}\mathbf{1}^\top + g \cdot f_\theta(g^{-1} \cdot (\mathbf{x} - \bar{\mathbf{x}}\mathbf{1}^\top))\right], \tag{A.36}$$

where $\bar{\mathbf{x}} \in \mathbb{R}^n$ is centroid (mean over channels) of data $\mathbf{x} \in \mathbb{R}^{n\times d}$ and distribution p_ω is $\mathrm{O}(n)/\mathrm{SO}(n)$ equivariant for $\mathrm{E}(n)/\mathrm{SE}(n)$ equivariant symmetrization, respectively. We now show the following:

Proposition 21. *The proposed symmetrization $\phi_{\theta,\omega}$ for the Euclidean group $\mathrm{E}(n)$ is equivariant.*

Proof. We prove $\phi_{\theta,\omega}(g' \cdot \mathbf{x}) = g' \cdot \phi_{\theta,\omega}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ and $g' \in \mathrm{E}(n)$. From Equation (A.36), we have:

$$\phi_{\theta,\omega}(g' \cdot \mathbf{x}) = \mathbb{E}_{p_\omega(g|g'\cdot\mathbf{x}-\overline{g'\cdot\mathbf{x}}\mathbf{1}^\top)}\left[\,\overline{g'\cdot\mathbf{x}}\mathbf{1}^\top + g \cdot f_\theta(g^{-1} \cdot (g' \cdot \mathbf{x} - \overline{g'\cdot\mathbf{x}}\mathbf{1}^\top))\right].$$

In general, an element of Euclidean group $g' \in \mathrm{E}(n)$ acts on data $\mathbf{x} \in \mathbb{R}^{n\times d}$ via $g' \cdot \mathbf{x} = \mathbf{Q}_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top$ where $\mathbf{Q}_{g'} \in \mathrm{O}(n)$ is its rotation component and $\mathbf{t}_{g'} \in \mathbb{R}^n$ is its translation component [Puny et al., 2022, Kaba et al., 2023]. With this, the centroid of the transformed data $g' \cdot \mathbf{x}$ is given as follows:

$$\overline{g'\cdot\mathbf{x}} = \overline{\mathbf{Q}_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top} = \overline{\mathbf{Q}_{g'}\mathbf{x}} + \mathbf{t}_{g'} = \mathbf{Q}_{g'}\overline{\mathbf{x}} + \mathbf{t}_{g'},$$

which leads to the following:

$$\begin{aligned} g' \cdot \mathbf{x} - \overline{g'\cdot\mathbf{x}}\mathbf{1}^\top &= \mathbf{Q}_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top - \mathbf{Q}_{g'}\overline{\mathbf{x}}\mathbf{1}^\top - \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}_{g'}(\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top). \end{aligned}$$

Above shows that subtracting centroid eliminates the translation component of the problem and leaves $\mathrm{O}(n)$-equivariance component. Based on that, we have the following:

$$\begin{aligned} \phi_{\theta,\omega}(g' \cdot \mathbf{x}) &= \mathbb{E}_{p_\omega(g|\mathbf{Q}_{g'}(\mathbf{x}-\overline{\mathbf{x}}\mathbf{1}^\top))}\left[\mathbf{Q}_{g'}\bar{\mathbf{x}}\mathbf{1}^\top + \mathbf{t}_{g'}\mathbf{1}^\top + g \cdot f_\theta(g^{-1} \cdot (\mathbf{Q}_{g'}(\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)))\right] \\ &= \mathbb{E}_{p_\omega(g|g'\cdot(\mathbf{x}-\overline{\mathbf{x}}\mathbf{1}^\top))}\left[g' \cdot \bar{\mathbf{x}}\mathbf{1}^\top + g \cdot f_\theta(g^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))\right] + \mathbf{t}_{g'}\mathbf{1}^\top. \end{aligned}$$

Note that, inside the expectation, we interpret the rotation component of g' as an element of the orthogonal group $\mathrm{O}(n)$. Similar as in the proof of Theorem 9, we introduce transformed random variable $h = g'^{-1}g \in \mathrm{O}(n)$ that $g = g'h$. Since the distribution p_ω is $\mathrm{O}(n)$-equivariant, we can see that $p_\omega(g|g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) = p_\omega(g'^{-1}g|g'^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) = p_\omega(g'^{-1}g|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top) = p_\omega(h|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)$. Thus we can rewrite the above expectation with respect to h as follows:

$$\begin{aligned} \phi_{\theta,\omega}(g' \cdot \mathbf{x}) &= \mathbb{E}_{p_\omega(h|\mathbf{x}-\overline{\mathbf{x}}\mathbf{1}^\top)}\left[g' \cdot \bar{\mathbf{x}}\mathbf{1}^\top + g'h \cdot f_\theta((g'h)^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))\right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbb{E}_{p_\omega(h|\mathbf{x}-\overline{\mathbf{x}}\mathbf{1}^\top)}\left[g' \cdot \bar{\mathbf{x}}\mathbf{1}^\top + g'h \cdot f_\theta(h^{-1} \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))\right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}_{g'}\mathbb{E}_{p_\omega(h|\mathbf{x}-\overline{\mathbf{x}}\mathbf{1}^\top)}\left[\bar{\mathbf{x}}\mathbf{1}^\top + h \cdot f_\theta(h^{-1} \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))\right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}_{g'}\phi_{\theta,\omega}(\mathbf{x}) + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= g' \cdot \phi_{\theta,\omega}(\mathbf{x}), \end{aligned}$$

showing the $\mathrm{E}(n)$-equivariance of $\phi_{\theta,\omega}$. □

Proposition 22. *The proposed symmetrization $\phi_{\theta,\omega}$ for special Euclidean group $\mathrm{SE}(n)$ is equivariant.*

Proof. We prove $\phi_{\theta,\omega}(g' \cdot \mathbf{x}) = g' \cdot \phi_{\theta,\omega}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$ and $g' \in \mathrm{SE}(n)$, in an analogous manner to the proof of Proposition 21. From Equation (A.36), we have:

$$\phi_{\theta,\omega}(g' \cdot \mathbf{x}) = \mathbb{E}_{p_\omega(g|g' \cdot \mathbf{x} - \overline{g' \cdot \mathbf{x}}\mathbf{1}^\top)} \left[\overline{g' \cdot \mathbf{x}}\mathbf{1}^\top + g \cdot f_\theta(g^{-1} \cdot (g' \cdot \mathbf{x} - \overline{g' \cdot \mathbf{x}}\mathbf{1}^\top)) \right].$$

In general, an element of special Euclidean group $g' \in \mathrm{SE}(n)$ acts on data $\mathbf{x} \in \mathbb{R}^{n \times d}$ via $g' \cdot \mathbf{x} = \mathbf{Q}^+_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top$ where $\mathbf{Q}^+_{g'} \in \mathrm{SO}(n)$ is rotation component and $\mathbf{t}_{g'} \in \mathbb{R}^n$ is translation [Puny et al., 2022, Kaba et al., 2023]. With this, the centroid of the transformed data $g' \cdot \mathbf{x}$ is given as follows:

$$\overline{g' \cdot \mathbf{x}} = \overline{\mathbf{Q}^+_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top} = \overline{\mathbf{Q}^+_{g'}\mathbf{x}} + \mathbf{t}_{g'} = \mathbf{Q}^+_{g'}\overline{\mathbf{x}} + \mathbf{t}_{g'},$$

which leads to the following:

$$\begin{aligned} g' \cdot \mathbf{x} - \overline{g' \cdot \mathbf{x}}\mathbf{1}^\top &= \mathbf{Q}^+_{g'}\mathbf{x} + \mathbf{t}_{g'}\mathbf{1}^\top - \mathbf{Q}^+_{g'}\overline{\mathbf{x}}\mathbf{1}^\top - \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}^+_{g'}(\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top). \end{aligned}$$

Similar as in Proposition 21, subtracting centroid only leaves $\mathrm{SO}(n)$ component. We then have:

$$\begin{aligned} \phi_{\theta,\omega}(g' \cdot \mathbf{x}) &= \mathbb{E}_{p_\omega(g|\mathbf{Q}^+_{g'}(\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))} \left[\mathbf{Q}^+_{g'}\overline{\mathbf{x}}\mathbf{1}^\top + \mathbf{t}_{g'}\mathbf{1}^\top + g \cdot f_\theta(g^{-1} \cdot (\mathbf{Q}^+_{g'}(\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))) \right] \\ &= \mathbb{E}_{p_\omega(g|g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top))} \left[g' \cdot \overline{\mathbf{x}}\mathbf{1}^\top + g \cdot f_\theta(g^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) \right] + \mathbf{t}_{g'}\mathbf{1}^\top, \end{aligned}$$

where, inside the expectation, we interpret the rotation component of g' as an element of the special orthogonal group $\mathrm{SO}(n)$. Similar as in Theorem 9, we introduce $h = g'^{-1}g \in \mathrm{SO}(n)$ that $g = g'h$. As the distribution p_ω is $\mathrm{SO}(n)$-equivariant, we have $p_\omega(g|g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) = p_\omega(g'^{-1}g|g'^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) = p_\omega(h|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)$. We then rewrite the expectation with respect to h:

$$\begin{aligned} \phi_{\theta,\omega}(g' \cdot \mathbf{x}) &= \mathbb{E}_{p_\omega(h|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)} \left[g' \cdot \overline{\mathbf{x}}\mathbf{1}^\top + g'h \cdot f_\theta((g'h)^{-1}g' \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) \right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbb{E}_{p_\omega(h|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)} \left[g' \cdot \overline{\mathbf{x}}\mathbf{1}^\top + g'h \cdot f_\theta(h^{-1} \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) \right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}^+_{g'}\mathbb{E}_{p_\omega(h|\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)} \left[\overline{\mathbf{x}}\mathbf{1}^\top + h \cdot f_\theta(h^{-1} \cdot (\mathbf{x} - \overline{\mathbf{x}}\mathbf{1}^\top)) \right] + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= \mathbf{Q}^+_{g'}\phi_{\theta,\omega}(\mathbf{x}) + \mathbf{t}_{g'}\mathbf{1}^\top \\ &= g' \cdot \phi_{\theta,\omega}(\mathbf{x}), \end{aligned}$$

showing the $\mathrm{SE}(n)$-equivariance of $\phi_{\theta,\omega}$. □

Product Group $H \times K$ For the product group $H \times K$, we assume that the base representation for each element $g = (h, k)$ is given as a pair of representations $\rho(g) = (\rho(h), \rho(k))$. Without loss of generality, we further assume that the representation $\rho(g)$ can be expressed as the Kronecker product $\rho(g) = \rho(h) \otimes \rho(k)$ that acts on flattened data $\mathrm{vec}(\mathbf{x})$ as $\mathbf{x} \mapsto \mathrm{vec}^{-1}(\rho(g)\mathrm{vec}(\mathbf{x}))$. This follows the standard approach in equivariant deep learning [Finzi et al., 2021, Maron et al., 2019b] that deals with composite representations using direct sum and tensor products of base group representations.

Above approach applies to many practical product groups, including sets and graphs with Euclidean attributes ($\mathrm{S}_n \times \mathrm{O}(d)/\mathrm{SO}(d)$[6]) and sets of symmetric elements ($\mathrm{S}_n \times H$) in general [Maron et al., 2020]. For example, for the group $\mathrm{S}_n \times \mathrm{O}(d)$ on data $\mathbf{x} \in \mathbb{R}^{n \times d}$, an element $g = (h, k)$ has representation

[6]This is after handling the translation component of the Euclidean group $\mathrm{E}(d)/\mathrm{SE}(d)$ as in Equation (A.36).

$\rho(g) = \rho(h) \otimes \rho(k) \in \mathbb{R}^{nd \times nd}$ combined from permutation $\rho(h) \in \mathbb{R}^{n \times n}$ and rotation $\rho(k) \in \mathbb{R}^{d \times d}$, which acts by $\mathbf{x} \mapsto \mathrm{vec}^{-1}(\rho(g)\mathrm{vec}(\mathbf{x}))$ or more simply $\mathbf{x} \mapsto \rho(h)\mathbf{x}\rho(k)^\top$.

Now we recall that the $p_\omega(g|\mathbf{x})$ for the product group $H \times K$ is implemented as follows:

1. Sample noise $\boldsymbol{\epsilon} \in \mathcal{E}$ from i.i.d. normal $\mathcal{N}(0, \eta^2)$ such that $p(\boldsymbol{\epsilon})$ is invariant under representations of H and K that satisfy $|\det \rho'(h)| = 1$ and $|\det \rho'(k)| = 1$, respectively. For example, for $\mathrm{S}_n \times \mathrm{O}(d)$, the noise $\boldsymbol{\epsilon} \in \mathbb{R}^{n \times d}$ that follows i.i.d. normal $\mathcal{N}(0, \eta^2)$ is invariant under base representations of both S_n and $\mathrm{O}(d)$ which are orthogonal.

2. Use a $H \times K$-equivariant neural network to obtain features $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto (\mathbf{Z}_H, \mathbf{Z}_K)$ where $\mathbf{Z}_H$ is K-invariant and $\mathbf{Z}_K$ is H-invariant. For example, for $\mathrm{S}_n \times \mathrm{O}(d)$, we expect node-level scalar features $\mathbf{Z}_{\mathrm{S}_n} \in \mathbb{R}^n$ to be $\mathrm{O}(d)$-invariant and d global rotary features $\mathbf{Z}_{\mathrm{O}(d)} \in \mathbb{R}^{d \times d}$ to be S_n-invariant.

3. Apply postprocessing for H and K groups onto $\mathbf{Z}_H$ and $\mathbf{Z}_K$ respectively to obtain representations $\mathbf{Z}_H \mapsto \rho(h)$ and $\mathbf{Z}_K \mapsto \rho(k)$ of H and K groups respectively. For example, for $\mathrm{S}_n \times \mathrm{O}(d)$, we use argsort in Equation (A.35) to obtain $\mathbf{Z}_{\mathrm{S}_n} \mapsto \rho(h)$ and Gram-Schmidt process to obtain $\mathbf{Z}_{\mathrm{O}(d)} \mapsto \rho(k)$.

4. Combine the representations $\rho(g) = (\rho(h), \rho(k))$ to obtain a representation for the $H \times K$ group.

We now show the following:

Proposition 23. *The proposed distribution $p_\omega(g|\mathbf{x})$ for the product group $H \times K$ is equivariant.*

Proof. By assumption, $p(\boldsymbol{\epsilon})$ is invariant under representations of H and K that satisfy $|\det \rho'(h)| = 1$ and $|\det \rho'(k)| = 1$, respectively. This implies $H \times K$-invariance as well, since $p(\boldsymbol{\epsilon}) = p(h \cdot \boldsymbol{\epsilon}) = p(k \cdot \boldsymbol{\epsilon})$ for all $\boldsymbol{\epsilon} \in \mathcal{E}, h \in H, k \in K$ gives $p(k \cdot h \cdot \boldsymbol{\epsilon}) = p(k \cdot (h \cdot \boldsymbol{\epsilon})) = p(h \cdot \boldsymbol{\epsilon}) = p(\boldsymbol{\epsilon})$, and Kronecker product of matrices of determinant 1 gives a matrix of determinant 1. Furthermore, the map $(\mathbf{x}, \boldsymbol{\epsilon}) \mapsto (\rho(h), \rho(k)) = \rho(g)$ is overall $H \times K$ equivariant, since an input transformed with $g' = (h', k')$ is first mapped by the equivariant neural network as $(g' \cdot \mathbf{x}, g' \cdot \boldsymbol{\epsilon}) \mapsto (h' \cdot \mathbf{Z}_H, k' \cdot \mathbf{Z}_K)$, then postprocessed as $(h' \cdot \mathbf{Z}_H, k' \cdot \mathbf{Z}_K) \mapsto (\rho(h')\rho(h), \rho(k')\rho(k)) = (\rho(h'), \rho(k')) \cdot (\rho(h), \rho(k)) = \rho(g')\rho(g)$. Combining the above, by Theorem 11, the distribution $p_\omega(g|\mathbf{x})$ is $H \times K$-equivariant. □

A.5 Appendix of orbit distance minimization

A.5.1 Proofs

Proof of Theorem 12

For the proof, we provide a formal definition of a distance metric d on the orbit space.

Definition 11. *A function $d : \mathbb{R}^{n\times n}/G \times \mathbb{R}^{n\times n}/G \to \mathbb{R}^+$ is a distance metric on the orbit space $\mathbb{R}^{n\times n}/G$ if it satisfies the following conditions for all orbits $[\mathbf{h}], [\mathbf{h}'], [\mathbf{h}''] \in \mathbb{R}^{n\times n}/G$:*

1. $d([\mathbf{h}], [\mathbf{h}']) \geq 0$ *(non-negativity),*
2. $d([\mathbf{h}], [\mathbf{h}']) = 0 \iff [\mathbf{h}] = [\mathbf{h}']$ *(identity of indiscernibles),*
3. $d([\mathbf{h}], [\mathbf{h}']) = d([\mathbf{h}'], [\mathbf{h}])$ *(symmetry),*
4. $d([\mathbf{h}], [\mathbf{h}']) \leq d([\mathbf{h}], [\mathbf{h}'']) + d([\mathbf{h}''], [\mathbf{h}'])$ *(triangle inequality).*

We now prove Theorem 12.

Proof. Recall the definition of an orbit $[\mathbf{h}] = \{g \cdot \mathbf{h} : g \in G\}$. ($\Longrightarrow$) If $d([\mathbf{h}], [\mathbf{I}]) = 0$, since d is a distance metric, we have $[\mathbf{h}] = [\mathbf{I}]$ from identity of indiscernibles. This implies $[\mathbf{h}] = \rho(G)$ since $[\mathbf{I}] = \{g \cdot \mathbf{I} : g \in G\} = \{\rho(g) : g \in G\} = \rho(G)$. On the orbit $[\mathbf{h}] = \{g \cdot \mathbf{h} : g \in G\}$, by selecting the identity element $\mathrm{id} \in G$ we get $\mathbf{h} \in [\mathbf{h}]$, thus $\mathbf{h} \in \rho(G)$. ($\Longleftarrow$) If q_ω always outputs valid group representations $\mathbf{h} \in \rho(G)$, we can write $\mathbf{h} = \rho(h)$ for some $h \in G$. Then we have $[\mathbf{h}] = \{g \cdot \mathbf{h} : g \in G\} = \{g \cdot \rho(h) : g \in G\} = \{(gh) \cdot \mathbf{I} : g \in G\} = \{g' \cdot \mathbf{I} : g'h^{-1} \in G\} = \{g' \cdot \mathbf{I} : g' \in G\} = [\mathbf{I}]$. Note that we used the associativity and invertibility of group elements as well as the fact that right operation by $h \in G$ maps a group G to itself. Since $[\mathbf{h}] = [\mathbf{I}]$ and d is a distance metric, we have that $d([\mathbf{h}], [\mathbf{I}]) = 0$ due to identity of indiscernibles, which is a global minimum due to non-negativity. □

Proof of Theorem 13

Proof. We first note that a vector norm $\|\cdot\|$ induces a distance metric $d'(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|$, which is called the induced metric. We now explicitly show that $d([\mathbf{h}], [\mathbf{h}']) = \|f(\mathbf{h}) - f(\mathbf{h}')\|$ satisfies the four conditions in Definition 11. Since $\|\cdot\|$ is a vector norm, non-negativity clearly holds. Since f is an orbit separating invariant $[\mathbf{h}] \neq [\mathbf{h}'] \iff f(\mathbf{h}) \neq f(\mathbf{h}')$, by invoking the identity of indiscernibles of the induced metric we have $[\mathbf{h}] \neq [\mathbf{h}'] \iff \|f(\mathbf{h}) - f(\mathbf{h}')\| \neq 0$, which proves the identity of indiscernibles for $d([\mathbf{h}], [\mathbf{h}'])$. Symmetry and triangle inequality of $d([\mathbf{h}], [\mathbf{h}'])$ are inherited from the symmetry and triangle inequality of the induced metric, as we have $d([\mathbf{h}], [\mathbf{h}']) = \|f(\mathbf{h}) - f(\mathbf{h}')\| = \|f(\mathbf{h}') - f(\mathbf{h})\| = d([\mathbf{h}'], [\mathbf{h}])$ for symmetry and $d([\mathbf{h}], [\mathbf{h}']) = \|f(\mathbf{h}) - f(\mathbf{h}')\| \leq \|f(\mathbf{h}) - f(\mathbf{h}'')\| + \|f(\mathbf{h}'') - f(\mathbf{h}')\| = d([\mathbf{h}], [\mathbf{h}'']) + d([\mathbf{h}''], [\mathbf{h}'])$ for triangle inequality. Therefore, $d([\mathbf{h}], [\mathbf{h}'])$ is a distance metric on the orbit space. □

A.6 Appendix of transformation-inverting energy diffusion

A.6.1 Preliminary

We provide an overview of the mathematical background, and refer the readers to Lee [2012], Tu [2010] for more details.

Lie group A Lie group G is a group that is also a smooth manifold, such that multiplications of elements and taking inverses are smooth. In deep learning, it is useful as an abstraction of a class of continuous data transformations such as perspective transformations of an image. For any $g \in G$, we denote by $L_g : G \to G$ the left multiplication map $h \mapsto gh$ and by $R_g : G \to G$ the right multiplication map $h \mapsto hg$. We denote by μ the left-invariant (unnormalized) Haar measure on G.

Tangent spaces As a manifold, a Lie group G has a tangent space T_gG at each point g, intuitively containing all possible directions of infinitesimal updates at g. We can map between tangent spaces using the notion of differentials. For any smooth function $f : G \to G$ and any $g \in G$, we denote by $\mathrm{d}f_g : T_gG \to T_{f(g)}G$ the differential of f evaluated at g. For left multiplications L_g, the differentials $\mathrm{d}(L_g)_h : T_hG \to T_{gh}G$ are bijective linear maps between tangent spaces called the tangent maps. The tangent maps satisfy $\mathrm{d}(L_g)_{hk} \circ \mathrm{d}(L_h)_k = \mathrm{d}(L_{gh})_k$ and $\mathrm{d}(L_e)_g = \mathrm{id}_{T_gG}$ for all $g, h, k \in G$.

Lie algebra While a Lie group is a curved manifold, a lot of its properties derive from its tangent space at the identity, the Lie algebra, defined by $\mathfrak{g} \equiv T_eG$. For an n-dimensional Lie group, the Lie algebra $\mathfrak{g}$ is an n-dimensional vector space equipped with a binary operation $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called the Lie bracket. We can always construct an inner product $\langle \cdot, \cdot \rangle_{\mathfrak{g}}$ on $\mathfrak{g}$ by choosing any basis $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ and defining $\langle \sum_i u_i \mathbf{e}_i, \sum_i v_i \mathbf{e}_i \rangle_{\mathfrak{g}} \equiv \sum_i u_i v_i$. This is the unique inner product on $\mathfrak{g}$ for which the chosen basis is orthonormal. The exponential map $\exp : \mathfrak{g} \to G$ is a smooth map that specifies group structure from the Lie algebra. In general, this specification is local around the identity $e = \exp(\mathbf{0})$ as exp can be non-surjective for non-compact G. Yet, for connected Lie groups, any element can be reached as a finite product of these local elements [Olver, 1993, Proposition 1.24].

Gradients We can define the notion of gradients on a Lie group by equipping it with a choice of metric. For a finite-dimensional Lie group, one possible choice is a metric $\langle \cdot, \cdot \rangle$ that is left-invariant:

$$\langle \mathrm{d}(L_h)_g \mathbf{u}, \mathrm{d}(L_h)_g \mathbf{v} \rangle_{hg} = \langle \mathbf{u}, \mathbf{v} \rangle_g.$$

Such a metric can be always constructed by extending any inner product $\langle \cdot, \cdot \rangle_{\mathfrak{g}}$ on the Lie algebra:

$$\langle \mathbf{u}, \mathbf{v} \rangle_g \equiv \langle \mathrm{d}(L_{g^{-1}})_g \mathbf{u}, \mathrm{d}(L_{g^{-1}})_g \mathbf{v} \rangle_{\mathfrak{g}}.$$

Then, for any smooth function $f : G \to \mathbb{R}$, the gradient $\nabla f(g)$ at each point $g \in G$ is given as the unique element of the tangent space T_gG that satisfies the following for all $\mathbf{v} \in T_gG$:

$$\langle \nabla f(g), \mathbf{v} \rangle_g = \mathrm{d}f_g(\mathbf{v}), \tag{A.37}$$

where $\mathrm{d}f_g : T_gG \to \mathbb{R}$ is the differential of f evaluated at g [Lee, 2012, Section 13].

Trivialization In deep learning context, the curved geometry of Lie groups can be challenging to deal with. We can narrow this down to two specific difficulties in our problem setup.

1. In energy-based diffusion, one usually employs an expectation form of the score function that averages energy gradients evaluated across the state space [De Bortoli et al., 2024, Akhound-Sadegh et al., 2024]. However, on a Lie group, gradients of a function at different points live in different tangent spaces and hence are not directly compatible, e.g., cannot be added or averaged.

2. In general, during sampling on a Lie group, the update directions live in tangent spaces that change over time simultaneously as a sample is updated, requiring a careful handling.

We can sidestep both difficulties by always working in the Lie algebra instead of arbitrary tangent spaces. This technique is called (left-)trivialization [Lezcano Casado, 2019, Tao and Ohsawa, 2020, Kong and Tao, 2024, Zhu et al., 2025]. We use two related tools that together address the difficulties.

Our first tool is **trivialized gradient** that can be understood as gradient evaluated in the Lie algebra. For any smooth function $f : G \to \mathbb{R}$, we define the trivialized gradient $\nabla_{\mathfrak{g}} f(g) \in \mathfrak{g}$ at each point $g \in G$ by mapping the standard gradient $\nabla f(g) \in T_g G$ through the tangent map $\mathrm{d}(L_{g^{-1}})_g : T_g G \to \mathfrak{g}$:

$$\nabla_{\mathfrak{g}} f(g) \equiv \mathrm{d}(L_{g^{-1}})_g \nabla f(g).$$

Since trivialized gradients always live in the same vector space $\mathfrak{g}$, we can add or average them without restrictions. This underlies our novel score identity on general Lie groups.

Another nice property of trivialized gradients is that, despite their definition involving a tangent map, they can be computed directly using automatic differentiation without any explicit handling of tangent maps. This knowledge is not new [Kobilarov, 2014a,b], but rarely used in deep learning context.

Proposition 24 (Trivialized gradient)**.** *Let G be a finite-dimensional Lie group. Under any choice of a left-invariant metric, for any smooth function $f : G \to \mathbb{R}$, the following holds for all $g \in G$:*

$$\nabla_{\mathfrak{g}} f(g) = \nabla_{\mathbf{v}} f(g \exp(\mathbf{v}))|_{\mathbf{v}=\mathbf{0}} \,. \tag{A.38}$$

Proof. The proof can be found in Appendix A.6.2. □

Also, as a useful property, we can use the standard log-derivative trick on trivialized gradients:

Proposition 25 (Trivialized score)**.** *Let G be a finite-dimensional Lie group. Under any choice of a left-invariant metric, for any positive and smooth function $p : G \to \mathbb{R}$, it holds that $\nabla_{\mathfrak{g}} p = p \nabla_{\mathfrak{g}} \log p$.*

Proof. The proof can be found in Appendix A.6.2. □

Our second tool is **trivialized SDE** that specifies diffusion on Lie groups using components in the Lie algebra. Specifically, for an n-dimensional Lie group G and any choice of an orthonormal basis of the Lie algebra $\mathfrak{g}$, we consider Itô SDEs on the group having the following form:

$$\mathrm{d}g_t = \mathrm{d}(L_{g_t})_e \left[\phi(g_t, t)\,\mathrm{d}t + \gamma(t)\,\mathrm{d}\mathbf{w}_t^{\mathfrak{g}}\right], \tag{A.39}$$

with $\mathrm{d}\mathbf{w}_t^{\mathfrak{g}}$ the Brownian motion on Lie algebra given as $\mathrm{d}\mathbf{w}_t^{\mathfrak{g}} = \sum_i \mathbf{e}_i \mathrm{d}w_t^i$ where $\mathrm{d}w_t^1, ..., \mathrm{d}w_t^n$ are independent standard Brownian motions on $\mathbb{R}$ and $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ is the chosen orthonormal basis of $\mathfrak{g}$. In addition, $\phi(\cdot, t) : G \to \mathfrak{g}$ is the drift coefficient, and $\gamma(t) \in \mathbb{R}$ is the diffusion coefficient.

The trivialized nature of the SDE motivates a natural Euler time-discretization scheme for sampling that eliminates the need to handle moving tangent spaces. This is based on the fact that, for any fixed $\mathbf{v} \in \mathfrak{g}$, the trivialized dynamics $\mathrm{d}g = \mathrm{d}(L_g)_e \mathbf{v}\, \mathrm{d}t$ has a closed-form solution $g(t) = g(0)\exp(t\mathbf{v})$ that can be computed using the exponential map [Lee, 2012, (8.13) and Proposition 20.8]. Specifically, for step size $\Delta t > 0$, a Euler time-discretization can be written as follows, where $m = 0, ..., 1/\Delta t$:

$$\hat{g}_{m+1} = \hat{g}_m \exp\left(\phi(\hat{g}_m, m\Delta t)\,\Delta t + \gamma(m\Delta t)\,\mathbf{z}_m\right), \quad \mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \Delta t \mathbf{I}), \quad \hat{g}_0 \overset{d}{=} g_0. \tag{A.40}$$

A.6.2 Proofs

Proof of Proposition 7

Proof. For the first statement, using independence of $\mathbf{x}$ and g and the relation $\tilde{\mathbf{x}} = g \cdot \mathbf{x}$, we have

$$\begin{aligned} p\left(\mathbf{x}, \tilde{\mathbf{x}} \mid g\right) &= p_{\mathbf{x}}\left(\mathbf{x}\right) p\left(\tilde{\mathbf{x}} \mid \mathbf{x}, g\right) \\ p\left(\mathbf{x}, \tilde{\mathbf{x}} \mid g\right) &= p_{\mathbf{x}}\left(\mathbf{x}\right) \delta\left(\tilde{\mathbf{x}} - g \cdot \mathbf{x}\right) \end{aligned}$$

Marginalizing with respect to $\mathbf{x}$ yields

$$p\left(\tilde{\mathbf{x}} \mid g\right) = \int p_{\mathbf{x}}\left(\mathbf{x}\right) \delta\left(\tilde{\mathbf{x}} - g \cdot \mathbf{x}\right) \mathrm{d}\mathbf{x}$$

Since the group action is diffeomorphic, we have by the change of variable formula

$$\begin{aligned} p\left(\tilde{\mathbf{x}} \mid g\right) &= \int p_{\mathbf{x}}\left(\mathbf{x}\right) \delta\left(g^{-1} \cdot \tilde{\mathbf{x}} - \mathbf{x}\right) \left|\det J_{g^{-1}}\left(\tilde{\mathbf{x}}\right)\right| \mathrm{d}\mathbf{x} \\ p\left(\tilde{\mathbf{x}} \mid g\right) &= p_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) \left|\det J_{g^{-1}}\left(\tilde{\mathbf{x}}\right)\right| \end{aligned}$$

With a uniform prior on g, we obtain via Bayes rule

$$p\left(g \mid \tilde{\mathbf{x}}\right) \propto p_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) \left|\det J_{g^{-1}}\left(\tilde{\mathbf{x}}\right)\right|$$

For the second statement, for $\mathbf{x}' = g^{-1} \cdot \tilde{\mathbf{x}}$ and measurable $B \subseteq G \cdot \tilde{\mathbf{x}}$, we have

$$\begin{aligned} \mathbb{P}\left(\mathbf{x}' \in B \mid \tilde{\mathbf{x}}\right) &= \int_{g \in G} p\left(g \mid \tilde{\mathbf{x}}\right) \mathbb{1}_B\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) \mathrm{d}\mu(g) \\ \mathbb{P}\left(\mathbf{x}' \in B \mid \tilde{\mathbf{x}}\right) &\propto \int_{g \in G} p_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) \left|\det J_{g^{-1}}\left(\tilde{\mathbf{x}}\right)\right| \mathbb{1}_B\left(g^{-1} \cdot \tilde{\mathbf{x}}\right) \mathrm{d}\mu(g) \end{aligned}$$

Making the change of variable $\mathbf{x}' = g^{-1} \cdot \tilde{\mathbf{x}}$ and using the fact that the group action is diffeomorphic, we have by the change of variable formula

$$\mathbb{P}\left(\mathbf{x}' \in B \mid \tilde{\mathbf{x}}\right) \propto \int_{\mathbf{x}' \in G \cdot \tilde{\mathbf{x}}} p_{\mathbf{x}}\left(\mathbf{x}'\right) \mathbb{1}_B\left(\mathbf{x}'\right) \mathrm{d}\mu_{G \cdot \tilde{\mathbf{x}}}(\mathbf{x}')$$

Therefore we have the following density

$$p\left(\mathbf{x}' \mid \tilde{\mathbf{x}}\right) \propto p_{\mathbf{x}}\left(\mathbf{x}'\right) \mathbb{1}_{G \cdot \tilde{\mathbf{x}}}\left(\mathbf{x}'\right)$$

with respect to $\mu_{G \cdot \tilde{\mathbf{x}}}$. □

Proof of Proposition 8

Proof. By left-invariance of the Haar measure, the density $p(g \mid \tilde{\mathbf{x}})$ is equivariant if

$$p(h \cdot g \mid h \cdot \tilde{\mathbf{x}}) = p(g \mid \tilde{\mathbf{x}}), \quad \forall g, h \in G, \tilde{\mathbf{x}} \in \mathcal{X}$$

We can verify this explicitly for the posterior given by

$$p(g \mid \tilde{\mathbf{x}}) = \frac{e^{-E_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right)} \left|\det J_{g^{-1}}(\tilde{\mathbf{x}})\right|}{Z(\tilde{\mathbf{x}})}$$

We have

$$p(h \cdot g \mid h \cdot \tilde{\mathbf{x}}) = \frac{e^{-E_{\mathbf{x}}\left(g^{-1} \cdot h^{-1} \cdot h \cdot \tilde{\mathbf{x}}\right)} \left|\det J_{(h \cdot g)^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}{Z(h \cdot \tilde{\mathbf{x}})}$$
$$p(h \cdot g \mid h \cdot \tilde{\mathbf{x}}) = \frac{e^{-E_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right)} \left|\det J_{(h \cdot g)^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}{Z(h \cdot \tilde{\mathbf{x}})}$$

Using the chain rule for the Jacobian, we obtain

$$p(h \cdot g \mid h \cdot \tilde{\mathbf{x}}) = \frac{e^{-E_{\mathbf{x}}\left(g^{-1} \cdot \tilde{\mathbf{x}}\right)} \left|\det J_{g^{-1}}(\tilde{\mathbf{x}})\right| \left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}{Z(h \cdot \tilde{\mathbf{x}})}$$

We can absorb the factor $\left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|$ into the normalization constant by using the chain rule for Jacobians, then a change of variable $g' = h^{-1}g$ followed by left-invariance of the Haar measure

$$\begin{aligned}
\frac{\left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}{Z(h \cdot \tilde{\mathbf{x}})} &= \frac{\left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}{\int_G e^{-E_{\mathbf{x}}(g^{-1}h \cdot \tilde{\mathbf{x}})} \left|\det J_{g^{-1}}(h \cdot \tilde{\mathbf{x}})\right| \mathrm{d}\mu(g)} \\
&= \frac{\cancel{\left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|}}{\int_G e^{-E_{\mathbf{x}}(g^{-1}h \cdot \tilde{\mathbf{x}})} \left|\det J_{g^{-1}h}(\tilde{\mathbf{x}})\right| \cancel{\left|\det J_{h^{-1}}(h \cdot \tilde{\mathbf{x}})\right|} \mathrm{d}\mu(g)} \\
&= \frac{1}{\int_G e^{-E_{\mathbf{x}}(g'^{-1} \cdot \tilde{\mathbf{x}})} \left|\det J_{g'^{-1}}(\tilde{\mathbf{x}})\right| \mathrm{d}\mu(hg')} \\
&= \frac{1}{Z(\tilde{\mathbf{x}})}
\end{aligned}$$

This gives us $p(h \cdot g \mid h \cdot \tilde{\mathbf{x}}) = p(g \mid \tilde{\mathbf{x}})$. □

Proof of Proposition 24

Proof. Let $\mathbf{u} \in \mathfrak{g}$ be an arbitrary unit vector and let $\gamma(t) \equiv g \exp(t\mathbf{u})$ be an associated smooth curve on the group parameterized by $t \in \mathbb{R}$. Then from Lee [2012, Corollary 3.25] the following holds

$$\mathrm{d}f_g(\dot{\gamma}(0)) = \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma(t)) \Big|_{t=0}$$

Starting from the left-hand side, we get

$$\begin{aligned}
\mathrm{d}f_g(\dot{\gamma}(0)) &= \langle \nabla f(g), \dot{\gamma}(0) \rangle_g \\
&= \langle \nabla f(g), \mathrm{d}(L_g)_e \mathbf{u} \rangle_g \\
&= \langle \mathrm{d}(L_{g^{-1}})_g \nabla f(g), \mathrm{d}(L_{g^{-1}})_g \circ \mathrm{d}(L_g)_e \mathbf{u} \rangle_e \\
&= \langle \nabla_{\mathfrak{g}} f(g), \mathbf{u} \rangle_e
\end{aligned}$$

where we use $\dot{\gamma}(0) = \mathrm{d}(L_{\gamma(0)})_e \mathbf{u} = \mathrm{d}(L_g)_e \mathbf{u}$ [Lee, 2012, Sections 8, 9, 20] for the second equality.

Starting from the right-hand side, we get

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} f(\gamma(t))\bigg|_{t=0} &= \frac{\mathrm{d}}{\mathrm{d}t} f(g \exp(\mathbf{0} + t\mathbf{u}))\bigg|_{t=0} \\
&= \nabla_{\mathbf{u}} (f \circ L_g \circ \exp)(\mathbf{0}) \\
&= \langle \nabla_{\mathbf{v}} f(g \exp(\mathbf{v}))|_{\mathbf{v}=\mathbf{0}}, \mathbf{u} \rangle_e
\end{aligned}$$

where, for the second equality, we use the definition of directional derivative of the composite function $f \circ L_g \circ \exp : \mathfrak{g} \to \mathbb{R}$ along the direction $\mathbf{u}$ evaluated at $\mathbf{0}$.

Therefore, for arbitrary unit vector $\mathbf{u} \in \mathfrak{g}$ it holds that

$$\langle \nabla_{\mathfrak{g}} f(g), \mathbf{u} \rangle_e = \langle \nabla_{\mathbf{v}} f(g \exp(\mathbf{v}))|_{\mathbf{v}=\mathbf{0}}, \mathbf{u} \rangle_e$$

and we have Equation (A.38). □

Proof of Proposition 25

Proof. For any smooth function $f : G \to \mathbb{R}$, we can turn its differential $\mathrm{d}f_g$ evaluated at g into a linear functional on $\mathfrak{g}$ by pre-composing with $\mathrm{d}(L_{g^{-1}})_g^{-1}$. Concretely, for any $g \in G$ and $\mathbf{u} \in \mathfrak{g}$,

$$\langle \nabla_{\mathfrak{g}} f(g), \mathbf{u} \rangle_{\mathfrak{g}} = \mathrm{d}f_g(\mathrm{d}(L_{g^{-1}})_g^{-1} \mathbf{u}) = \frac{\mathrm{d}}{\mathrm{d}t} f(g \exp(t\mathbf{u}))\bigg|_{t=0}$$

This is exactly the directional derivative of f along the left-invariant vector field generated by $\mathbf{u}$.

We can apply the above equation to the left-trivialized score since we assumed the smoothness of each p. With $f = \log p$, we get

$$(\mathrm{d}(L_{g^{-1}})_g \nabla \log p(g))(\mathbf{u}) = \frac{\mathrm{d}}{\mathrm{d}t} \log p(g \exp(t\mathbf{u}))\bigg|_{t=0}$$

By the chain rule on $\mathbb{R}$,

$$\frac{\mathrm{d}}{\mathrm{d}t} \log p(g \exp(t\mathbf{u})) = \frac{1}{p(g \exp(t\mathbf{u}))} \frac{\mathrm{d}}{\mathrm{d}t} p(g \exp(t\mathbf{u}))$$

Evaluating at $t = 0$ gives

$$\langle \nabla_{\mathfrak{g}} \log p(g), \mathbf{u} \rangle_{\mathfrak{g}} = \frac{1}{p(g)} \frac{\mathrm{d}}{\mathrm{d}t} p(g \exp(t\mathbf{u}))\bigg|_{t=0} = \frac{\langle \nabla_{\mathfrak{g}} p(g), \mathbf{u} \rangle_{\mathfrak{g}}}{p(g)}$$

Since this holds for arbitrary $\mathbf{u} \in \mathfrak{g}$, we get $\nabla_{\mathfrak{g}} p(g) = p(g) \nabla_{\mathfrak{g}} \log p(g)$ for all $g \in G$. □

Proof of Proposition 9

The proof is inspired by the generator-based approach of Zhu et al. [2025], Holderrieth et al. [2024].

For an n-dimensional Lie group G with any choice of orthonormal basis $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$ of Lie algebra $\mathfrak{g}$, we recall the SDE given in Equation (A.39), with Brownian motion in the Lie algebra $\mathrm{d}\mathbf{w}_t^{\mathfrak{g}} = \sum_i \mathbf{e}_i \mathrm{d}w_t^i$, drift coefficient $\phi(\cdot, t) : G \to \mathfrak{g}$, and diffusion coefficient $\gamma(t) \in \mathbb{R}$:

$$\mathrm{d}g_t = \mathrm{d}(L_{g_t})_e \left[\phi(g_t, t)\,\mathrm{d}t + \gamma(t)\,\mathrm{d}\mathbf{w}_t^{\mathfrak{g}}\right].$$

We show some useful lemmas related to the SDE. We first derive its infinitesimal generator $\mathcal{L}_t$, which is a linear operator defined for every smooth function $f : G \to \mathbb{R}$ as the following for each $g \in G$:

$$(\mathcal{L}_t f)(g) \equiv \lim_{h \to 0^+} \frac{\mathbb{E}[f(g_{t+h}) - f(g_t) | g_t = g]}{h}.$$

Lemma 11. *Assume the SDE in Equation* (A.39) *has a smooth drift. Then its infinitesimal generator* $\mathcal{L}_t$ *satisfies the following for any smooth* $f : G \to \mathbb{R}$*, where* Δ *is the Laplace-Beltrami operator:*

$$\mathcal{L}_t f = \langle \nabla_{\mathfrak{g}} f, \phi(\cdot, t) \rangle_{\mathfrak{g}} + \frac{\gamma(t)^2}{2} \Delta f. \tag{A.41}$$

Proof. Let us denote $E_i \equiv \mathrm{d}(L_{[\cdot]})_e \mathbf{e}_i$ and rewrite Equation (A.39) as follows

$$\mathrm{d}g_t = \Phi(g_t, t)\,\mathrm{d}t + \sum_{i=1}^{n} \Gamma_i(g_t, t)\,\mathrm{d}w_t^i, \quad \Phi(g, t) \equiv \mathrm{d}(L_g)_e \phi(g, t), \quad \Gamma_i(g, t) \equiv \gamma(t) E_i$$

Since $\Phi(\cdot, t), \Gamma_1(\cdot, t), ..., \Gamma_n(\cdot, t)$ are smooth vector fields on G, this expression makes it explicit that Equation (A.39) is an Itô SDE taking G as the state space. Then, by applying Lee and Chirikjian [2025, Theorem 2 and Proposition 1][7] we get

$$\begin{aligned}
\mathcal{L}_t f &= \mathrm{d}f_{[\cdot]}(\Phi(\cdot, t)) + \frac{1}{2} \sum_{i=1}^{n} \mathrm{Hess}_f(\Gamma_i(\cdot, t), \Gamma_i(\cdot, t)) \\
&= \mathrm{d}f_{[\cdot]}(\Phi(\cdot, t)) + \frac{\gamma(t)^2}{2} \sum_{i=1}^{n} \mathrm{Hess}_f(E_i, E_i) \\
&= \mathrm{d}f_{[\cdot]}(\Phi(\cdot, t)) + \frac{\gamma(t)^2}{2} \Delta f \\
&= \langle \nabla f, \Phi(\cdot, t) \rangle + \frac{\gamma(t)^2}{2} \Delta f
\end{aligned}$$

where we use bilinearity of Hessian for the second equality and use Equation (A.37) for the last equality. With left-invariance of the metric we obtain Equation (A.41). □

We now derive the adjoint operator $\mathcal{L}_t^*$ of the infinitesimal generator, which is defined through the following relationship for any smooth, compactly supported test function $f : G \to \mathbb{R}$ and density ρ with respect to the Haar measure μ [Holderrieth et al., 2024, Appendix A.3]:

$$\int_G (\mathcal{L}_t f)\, \rho\, \mathrm{d}\mu = \int_G f\, (\mathcal{L}_t^* \rho)\, \mathrm{d}\mu. \tag{A.42}$$

[7] Here we consider an extension of the results in Lee and Chirikjian [2025] to time-dependent drift and diffusion coefficients. Such an extension can be obtained since the results are based on applying the chain rule to a standard form of Stratonovich SDE with respect to only the state variable and hence we can regard t as a constant.

Lemma 12. *Assume the SDE in Equation* (A.39) *has a smooth drift. The adjoint $\mathcal{L}_t^*$ of its infinitesimal generator satisfies the following for any bounded and smooth density ρ, where* div *is the divergence:*

$$\mathcal{L}_t^* \rho = -\mathrm{div}(\rho\, \Phi(\cdot, t)) + \frac{\gamma(t)^2}{2} \Delta \rho. \tag{A.43}$$

Proof. From Equation (A.42), Lemma 11, and bilinearity of the metric, we get

$$\int_G f\, (\mathcal{L}_t^* \rho)\, \mathrm{d}\mu = \int_G \langle \nabla f, \rho\, \Phi(\cdot, t) \rangle\, \mathrm{d}\mu + \frac{\gamma(t)^2}{2} \int_G \rho\, \Delta f\, \mathrm{d}\mu \tag{A.44}$$

For the first term, we use the following relationship

$$\begin{aligned} 0 &= \int_G \mathrm{div}(f\, \rho\, \Phi(\cdot, t))\, \mathrm{d}\mu \\ &= \int_G f\, \mathrm{div}(\rho\, \Phi(\cdot, t))\, \mathrm{d}\mu + \int_G \langle \nabla f, \rho\, \Phi(\cdot, t) \rangle\, \mathrm{d}\mu \end{aligned} \tag{A.45}$$

where the first equality follows from the divergence theorem for the compactly supported vector field $f\, \rho\, \Phi(\cdot, t)$, and the second equality follows from the identity $\mathrm{div}(f\, X) = f\, \mathrm{div} X + \langle \nabla f, X \rangle$ with the vector field $X = \rho\, \Phi(\cdot, t)$ [Chavel, 2006, Theorem III.7.3 and p.150].

For the second term, we use the fact that Δ is self-adjoint [Chavel, 2006, Theorem III.7.4], that is

$$\int_G \rho\, \Delta f\, \mathrm{d}\mu = \int_G f\, \Delta \rho\, \mathrm{d}\mu \tag{A.46}$$

Plugging Equation (A.45) and Equation (A.46) into Equation (A.44), we get

$$\int_G f\, (\mathcal{L}_t^* \rho)\, \mathrm{d}\mu = -\int_G f\, \mathrm{div}(\rho\, \Phi(\cdot, t))\, \mathrm{d}\mu + \frac{\gamma(t)^2}{2} \int_G f\, \Delta \rho\, \mathrm{d}\mu$$

Since this holds for every test function f, we get Equation (A.43). □

We are now ready to prove Proposition 9. We recall the forward trivialized SDE in Equation (3.11):

$$\mathrm{d}g_t = \mathrm{d}(L_{g_t})_e \left[\gamma(t)\, \mathrm{d}\mathbf{w}_t^{\mathfrak{g}} \right].$$

The proof uses adjoint Kolmogorov forward equation that describes density evolution ρ_t of a stochastic process using the adjoint of its infinitesimal generator [Zhu et al., 2025, Holderrieth et al., 2024]:

$$\frac{\partial}{\partial t} \rho_t = \mathcal{L}_t^* \rho_t.$$

Proof. Denoting the density of the forward process by p_t and of the reverse by $\bar{p}_t$, we prove $p_t = \bar{p}_{1-t}$ for all $t \in [0, 1]$ given the initial condition $p_1 = \bar{p}_0$.

From Lemma 12, the adjoint Kolmogorov forward equation of the forward process is

$$\frac{\partial}{\partial t} p_t = \frac{\gamma(t)^2}{2} \Delta p_t$$

Likewise, the adjoint Kolmogorov forward equation of the proposed reverse process is

$$\frac{\partial}{\partial t} \bar{p}_t = -\gamma(1-t)^2 \mathrm{div}(\bar{p}_t\, \nabla \log \bar{p}_t) + \frac{\gamma(1-t)^2}{2} \Delta \bar{p}_t = -\frac{\gamma(1-t)^2}{2} \Delta \bar{p}_t$$

where we used $\operatorname{div}(\bar{p}_t \nabla \log \bar{p}_t) = \operatorname{div}(\nabla \bar{p}_t) = \Delta \bar{p}_t$ for the second equality.

Then, the partial derivative of $\bar{p}_{1-t}$ with respect to t is

$$\frac{\partial}{\partial t}\bar{p}_{1-t} = \frac{\gamma(t)^2}{2}\Delta \bar{p}_{1-t}$$

Since this precisely matches the partial derivative of p_t with respect to t, given the initial condition $p_1 = \bar{p}_0$ we have that $p_t = \bar{p}_{1-t}$ for all $t \in [0, 1]$, completing the proof. □

Proof of Proposition 10

Before proving our main score identity, we state two lemmas regarding conditional random variables on general Lie groups. We distinguish between random variables and their realizations for clarity.

Lemma 13. *Let G be a finite-dimensional Lie group, and let X, W be independent random variables with densities p_X, p_W with respect to the left-invariant measure μ. Let $Y \equiv XW$. Then we have:*

$$p_{Y|X}(y|x) = p_W(x^{-1}y). \tag{A.47}$$

Proof. For any measurable set $A \subseteq G$, we have

$$\mathbb{P}(Y \in A | X = x) = \mathbb{P}(xW \in A) = \mathbb{P}(W \in x^{-1}A) = \int_{x^{-1}A} p_W(w)\,\mathrm{d}\mu(w)$$

where the first equality is due to independence of X and W.

Let us apply change of variable $y \equiv xw$. Since $\mathrm{d}\mu(w) = \mathrm{d}\mu(x^{-1}y) = \mathrm{d}\mu(y)$, we have

$$\mathbb{P}(Y \in A | X = x) = \int_{x^{-1}A} p_W(w)\,\mathrm{d}\mu(w) = \int_A p_W(x^{-1}y)\,\mathrm{d}\mu(y)$$

Since this holds for every measurable set $A \subseteq G$, we get Equation (A.47). □

For the next lemma, we introduce the concept of modular function $\lambda : G \to \mathbb{R}_{>0}$ of a Lie group G that specifies how the left-invariant measure μ changes under right multiplication. Specifically, for every measurable set $A \subseteq G$ and every $g \in G$, it holds that $\mu(Ag) = \lambda(g)\mu(A)$, and especially we have that $\mathrm{d}\mu(g^{-1}) = \lambda(g^{-1})\,\mathrm{d}\mu(g) = \lambda(g)^{-1}\,\mathrm{d}\mu(g)$ [Folland, 1994, Section 2.4].

Lemma 14. *Let G be a finite-dimensional Lie group, and let X, W be independent random variables with densities p_X, p_W with respect to μ. Let $Y \equiv XW$. Then we have, with λ the modular function:*

$$p_Y(y) = \int_G p_W(w)\, p_X(yw^{-1})\, \lambda(w)^{-1}\,\mathrm{d}\mu(w). \tag{A.48}$$

Proof. From Lemma 13, we obtain

$$p_Y(y) = \int_G p_X(x)\, p_{Y|X}(y|x)\,\mathrm{d}\mu(x) = \int_G p_X(x)\, p_W(x^{-1}y)\,\mathrm{d}\mu(x)$$

By applying change of variable $w = x^{-1}y$ with $\mathrm{d}\mu(x) = \mathrm{d}\mu(yw^{-1}) = \mathrm{d}\mu(w^{-1}) = \lambda(w)^{-1}\mathrm{d}\mu(w)$, we get Equation (A.48). □

We are now ready to prove the main score identity in Proposition 10.

Proof. We note that g_0 and g_t as random variables follow a relationship $g_t = g_0 w_t$ where $g_0 \sim p_0$ and $w_t \sim k_t$ are independent. By applying Lemma 14, we get

$$p_t(g_t) = \int_G k_t(w_t)\, p_0(g_t w_t^{-1})\, \lambda(w_t)^{-1}\, \mathrm{d}\mu(w_t) \tag{A.49}$$

By taking trivialized gradient $\nabla_{\mathfrak{g}}$ with respect to the variable g_t and noting that it is a linear operator,

$$\nabla_{\mathfrak{g}} p_t(g_t) = \int_G k_t(w_t)\, \nabla_{\mathfrak{g}} p_0(g_t w_t^{-1})\, \lambda(w_t)^{-1}\, \mathrm{d}\mu(w_t) \tag{A.50}$$

Then, using the identity $\nabla_{\mathfrak{g}} p = p\, \nabla_{\mathfrak{g}} \log p$ from Proposition 25 we get

$$\nabla_{\mathfrak{g}} \log p_t(g_t) = \int_G \nabla_{\mathfrak{g}} \log p_0(g_t w_t^{-1})\, \frac{k_t(w_t)\, p_0(g_t w_t^{-1})}{p_t(g_t)}\, \lambda(w_t)^{-1}\, \mathrm{d}\mu(w_t)$$

By change of variable $g_0 = g_t w_t^{-1}$ with $\lambda(w_t)^{-1}\, \mathrm{d}\mu(w_t) = \mathrm{d}\mu(w_t^{-1}) = \mathrm{d}\mu(g_t w_t^{-1}) = \mathrm{d}\mu(g_0)$, and using the relationship $p_{t|0}(g_t|g_0) = k_t(g_0^{-1} g_t) = k_t(w_t)$ obtained by Lemma 13, we get

$$\nabla_{\mathfrak{g}} \log p_t(g_t) = \int_G \nabla_{\mathfrak{g}} \log p_0(g_0)\, \frac{p_{t|0}(g_t|g_0)\, p_0(g_0)}{p_t(g_t)}\, \mathrm{d}\mu(g_0)$$

where we remark that $\nabla_{\mathfrak{g}}$ is taken with respect to g_t, so $\nabla_{\mathfrak{g}} \log p_0(g_0)$ is understood as $\nabla_{\mathfrak{g}} \log p_0(g_t b)$ for b having the value of $g_t^{-1} g_0$. Using Bayes' rule we get Equation (3.13). □

Proof of Proposition 11

The proof is inspired by the convolution argument of Akhound-Sadegh et al. [2024], or equivalently an importance sampling estimation applied to target score identity [De Bortoli et al., 2024].

Proof. We start at Equation (A.50) and use $\nabla_{\mathfrak{g}} p_t = p_t\, \nabla_{\mathfrak{g}} \log p_t$ with Equation (A.49) to get

$$\nabla_{\mathfrak{g}} \log p_t(g) = \frac{\int_G k_t(w)\, \nabla_{\mathfrak{g}} p_0(g w^{-1})\, \lambda(w)^{-1}\, \mathrm{d}\mu(w)}{\int_G k_t(w)\, p_0(g w^{-1})\, \lambda(w)^{-1}\, \mathrm{d}\mu(w)}$$

Then using $p_0(\cdot) = e^{-E(\cdot)}/Z$ with the normalization constant Z canceling out, we get

$$\nabla_{\mathfrak{g}} \log p_t(g) = \frac{\int_G k_t(w)\, \nabla_{\mathfrak{g}} e^{-E(g_t w^{-1}) - \log \lambda(w)}\, \mathrm{d}\mu(w)}{\int_G k_t(w)\, e^{-E(g_t w^{-1}) - \log \lambda(w)}\, \mathrm{d}\mu(w)}$$

Using linearity to move $\nabla_{\mathfrak{g}}$ with respect to g out of the integration, and then applying the identity $\nabla_{\mathfrak{g}} f = f \nabla_{\mathfrak{g}} \log f$ with f the denominator, we get Equation (3.14) (by writing g_t instead of g). □

Monte Carlo estimation and proof of consistency

We discuss sampling estimation of trivialized score based on Proposition 11 and prove its consistency. Starting at Equation (3.14) and using Monte Carlo estimation for the integral with N samples, we get:

$$\nabla_{\mathfrak{g}} \log p_t(g) \approx \nabla_{\mathfrak{g}} \log \frac{1}{N} \sum_{i=1}^{N} \exp\big(-E(g w^{(i)-1}) - \log \lambda(w^{(i)})\big), \quad w^{(i)} \sim k_t. \tag{A.51}$$

The $1/N$ factor does not affect the gradient, and hence can be removed. We then obtain an expression of the form $\log \sum_i \exp$ inside the gradient, which allows for a numerically stable evaluation using the LogSumExp trick [Akhound-Sadegh et al., 2024].

We note that the estimator is not unbiased because it has sample mean inside a concave function log. The bias can be understood as the Jensen gap. Nevertheless, we show that it is a consistent estimator.

Proposition 26. *For the forward SDE in Equation* (3.11) *where p_0 is a Boltzmann density specified by a smooth energy E, under proper integrability and positivity conditions, Equation* (A.51) *is a consistent estimator of $\nabla_{\mathfrak{g}} \log p_t$.*

Proof. For each $g \in G$, let us denote $h(g, w) \equiv e^{-E(gw^{-1}) - \log \lambda(w)}$ and define

$$J(g) \equiv \int_G k_t(w)\, \nabla_{\mathfrak{g}} h(g, w)\, \mathrm{d}\mu(w), \quad Z(g) \equiv \int_G k_t(w)\, h(g, w)\, \mathrm{d}\mu(w)$$

and define the corresponding empirical means

$$\hat{J}_N(g) \equiv \frac{1}{N} \sum_{i=1}^{N} \nabla_{\mathfrak{g}} h(g, w^{(i)}), \quad \hat{Z}_N(g) \equiv \frac{1}{N} \sum_{i=1}^{N} h(g, w^{(i)}), \quad w^{(i)} \sim k_t$$

Then Equations (3.14) and (A.51) can be written respectively as

$$\nabla_{\mathfrak{g}} \log p_t(g) = \nabla_{\mathfrak{g}} \log Z(g) = \frac{J(g)}{Z(g)}, \quad \nabla_{\mathfrak{g}} \log \hat{Z}_N(g) = \frac{\hat{J}_N(g)}{\hat{Z}_N(g)}$$

Assume integrability and positivity conditions

$$\int_G k_t(w)\, \|\nabla_{\mathfrak{g}} h(g, w)\|\, \mathrm{d}\mu(w) < \infty, \quad 0 < \int_G k_t(w)\, h(g, w)\, \mathrm{d}\mu(w) < \infty$$

By the strong law of large numbers

$$\hat{J}_N(g) \xrightarrow{\text{a.s.}} J(g), \quad \hat{Z}_N(g) \xrightarrow{\text{a.s.}} Z(g)$$

Then, by continuous mapping theorem [Shao, 2003, Theorem 1.10 and Example 1.30]

$$\frac{\hat{J}_N(g)}{\hat{Z}_N(g)} \xrightarrow{\text{a.s.}} \frac{J(g)}{Z(g)}$$

thus we have $\nabla_{\mathfrak{g}} \log \hat{Z}_N(g) \xrightarrow{\text{a.s.}} \nabla_{\mathfrak{g}} \log Z(g) = \nabla_{\mathfrak{g}} \log p_t(g)$. □

A.6.3 Practical implementation details

We discuss numerical computation of diffusion sampling $\hat{g}_0 \sim p_0 \propto e^{-E(g)}$ given $E : G \to \mathbb{R}$. The sampling follows a Euler time-discretization (Equation (A.40)) of the reverse diffusion (Equation (3.12)). For step size $\Delta t > 0$ we use the following for decreasing step indices $m = M, ..., 1$ with $M = 1/\Delta t$:

$$\begin{aligned} &\hat{g}_{m-1} = \hat{g}_m \exp\left(\gamma(m\Delta t)^2 \nabla_{\mathfrak{g}} \log p_{m\Delta t}(\hat{g}_m)\, \Delta t + \gamma(m\Delta t)\, \bar{\mathbf{z}}_m\right), \quad \bar{\mathbf{z}}_m \sim \mathcal{N}(\mathbf{0}, \Delta t \mathbf{I}), \\ &\hat{g}_M \overset{d}{=} \hat{w}_M \sim k_1, \end{aligned}$$

where we approximate $k_1 \approx p_1$ for initialization as usually done for variance-exploding diffusion [Song et al., 2020]. To run the sampling, we first need to sample from k_1 and at each step compute the MC estimator of the score $\nabla_{\mathfrak{g}} \log p_t$ (Equation (A.51)). This translates to the following requirements:

1. A method to sample $w \sim k_t$,
2. A method to calculate $\log \lambda(w)$,
3. A method to compute trivialized gradient $\nabla_{\mathfrak{g}}$ of general $f : G \to \mathbb{R}$.

For the first item, we recall the relationship $g_t = g_0 w_t$ with $g_0 \sim p_0$ and $w_t \sim k_t$ independent (proof of Proposition 10), and recall the forward process (Equation (3.11)). Together, these imply that $w_t \sim k_t$ is described by the following SDE which is identical to the forward SDE but starts at the identity:

$$\mathrm{d}w_t = \mathrm{d}(L_{w_t})_e \left[\gamma(t)\, \mathrm{d}\mathbf{w}_t^{\mathfrak{g}}\right], \quad w_0 = e.$$

This suggests a natural Euler time-discretization for sampling. For step indices $m = 1, ..., M$ we use:

$$\hat{w}_{m+1} = \hat{w}_m \exp\left(\gamma(m\Delta t)\, \mathbf{z}_m\right), \quad \mathbf{z}_m \sim \mathcal{N}(\mathbf{0}, \Delta t \mathbf{I}), \quad \hat{w}_0 = e, \tag{A.52}$$

and treat $\hat{w}_m \sim k_{m\Delta t}$.

For the second item, to evaluate the modular function $\lambda(\hat{w}_m)$ for $\hat{w}_m \sim k_{m\Delta t}$ we use the fact that $\hat{w}_m$ can be always written as the following, thanks to the structure of time-discretization (Equation (A.52)):

$$\hat{w}_m = \exp(\mathbf{v}_1) \exp(\mathbf{v}_2) \cdots \exp(\mathbf{v}_{m-1}),$$

where each $\mathbf{v}_i = \gamma(i\Delta t)\mathbf{z}_i$ is a Lie algebra element obtained at each sampling step. This allows us to evaluate the modular function only using the knowledge of the Lie bracket $[\cdot, \cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$:

Proposition 27. *Let G be connected n-dimensional Lie group. Under a choice of orthonormal basis $\{\mathbf{e}_1, ..., \mathbf{e}_n\}$, with structure constants c_{ij}^k satisfying $[\mathbf{e}_i, \mathbf{e}_j] = \sum_{k=1}^n c_{ij}^k \mathbf{e}_k$, let us denote:*

$$\mathbf{a} \equiv \begin{bmatrix} \sum_{j=1}^n c_{1j}^j \\ \cdots \\ \sum_{j=1}^n c_{nj}^j \end{bmatrix} \in \mathbb{R}^n, \quad \mathbf{b}_1 \equiv \begin{bmatrix} \langle \mathbf{e}_1, \mathbf{v}_1 \rangle_{\mathfrak{g}} \\ \cdots \\ \langle \mathbf{e}_n, \mathbf{v}_1 \rangle_{\mathfrak{g}} \end{bmatrix} \quad \cdots \quad \mathbf{b}_{m-1} \equiv \begin{bmatrix} \langle \mathbf{e}_1, \mathbf{v}_{m-1} \rangle_{\mathfrak{g}} \\ \cdots \\ \langle \mathbf{e}_n, \mathbf{v}_{m-1} \rangle_{\mathfrak{g}} \end{bmatrix} \in \mathbb{R}^n.$$

Then the following holds:

$$\lambda(\hat{w}_m) = e^{-\mathbf{a}^\top \mathbf{b}_1 - \mathbf{a}^\top \mathbf{b}_2 - ... - \mathbf{a}^\top \mathbf{b}_{m-1}}. \tag{A.53}$$

Proof. For every connected Lie group G, we have [Folland, 1994, Proposition 2.30]

$$\lambda(g) = \det \mathrm{Ad}(g^{-1}) \tag{A.54}$$

where $\mathrm{Ad}(\cdot) : \mathfrak{g} \to \mathfrak{g}$ is the adjoint action of G on the Lie algebra. Using its properties as a linear group

representation [Kirillov, 2008, Example 4.8], we get

$$\begin{aligned}\lambda(\hat{w}_m) &= \frac{1}{\det \mathrm{Ad}(\hat{w}_m)} = \frac{1}{\det \mathrm{Ad}(\exp(\mathbf{v}_1)\cdots\exp(\mathbf{v}_{m-1}))} \\ &= \frac{1}{\det \mathrm{Ad}(\exp(\mathbf{v}_1))} \times \cdots \times \frac{1}{\det \mathrm{Ad}(\exp(\mathbf{v}_{m-1}))}\end{aligned}$$

Then, using that $\det \mathrm{Ad}(\exp(\mathbf{v})) = \det e^{\mathrm{ad}(\mathbf{v})} = e^{\mathrm{tr}(\mathrm{ad}(\mathbf{v}))}$, where $\mathrm{ad}(\mathbf{v}) : \mathbf{u} \mapsto [\mathbf{v}, \mathbf{u}]$ is a linear map on $\mathfrak{g}$ [Kirillov, 2008, Lemma 3.14] with properties following from bilinearity of the Lie bracket

$$\begin{aligned}\mathrm{ad}(\mathbf{v})(\mathbf{e}_i) &= [\mathbf{v}, \mathbf{e}_i] = \sum_j \langle \mathbf{e}_j, \mathbf{v}\rangle_{\mathfrak{g}} [\mathbf{e}_j, \mathbf{e}_i] = \sum_j \langle \mathbf{e}_j, \mathbf{v}\rangle_{\mathfrak{g}} \sum_k c_{ji}^k \mathbf{e}_k \\ \mathrm{tr}(\mathrm{ad}(\mathbf{v})) &= \sum_i \langle \mathbf{e}_i, \mathrm{ad}(\mathbf{v})(\mathbf{e}_i)\rangle_{\mathfrak{g}} = \sum_i \langle \mathbf{e}_i, \mathbf{v}\rangle_{\mathfrak{g}} \sum_j c_{ij}^j\end{aligned}$$

we obtain Equation (A.53). □

Finally, for the last item, we use Proposition 24 to directly evaluate the trivialized gradient using automatic differentiation.

A.7 Appendix of random walk neural networks

A.7.1 Second-order random walks

In second-order random walks, the probability of choosing v_{t+1} depends not only on v_t but also on v_{t-1}. We consider non-backtracking [Alon et al., 2007, Fitzner and van der Hofstad, 2013] and node2vec [Grover and Leskovec, 2016] random walks as representatives.

A non-backtracking random walk is defined upon a first-order random walk algorithm, e.g. one in Equation (4.3), by enforcing $v_{t+1} \neq v_{t-1}$:

$$\text{Prob}[v_{t+1} = x | v_t = j, v_{t-1} = i] \coloneqq \begin{cases} \dfrac{\text{Prob}[v_{t+1} = x | v_t = j]}{\sum_{y \in N(j) \setminus \{i\}} \text{Prob}[v_{t+1} = y | v_t = j]} & \text{for } x \neq i, \\ 0 & \text{for } x = i, \end{cases} \tag{A.55}$$

where $\text{Prob}[v_{t+1} = x | v_t = j]$ is the probability of walking $j \to x$ given by the underlying first-order random walk. Notice that the probabilities are renormalized over $N(j) \setminus \{i\}$. This is ill-defined in the case the walk traverses $i \to j$ and reaches a dangling vertex j which has i as its only neighbor, since $N(j) \setminus \{i\} = \emptyset$. In such cases, we allow the random walk to "begrudgingly" backtrack [Rappaport et al., 2017], $i \to j$ and then $j \to i$, given that it is the only possible choice due to the dangling of j.

In case of node2vec random walk [Grover and Leskovec, 2016], a weighting term $\alpha(v_{t-1}, v_{t+1})$ with two (hyper)parameters, return p and in-out q, modifies the behavior of a first-order random walk:

$$\text{Prob}[v_{t+1} = x | v_t = j, v_{t-1} = i] \coloneqq \frac{\alpha(i, x) \, \text{Prob}[v_{t+1} = x | v_t = j]}{\sum_{y \in N(j)} \alpha(i, y) \, \text{Prob}[v_{t+1} = y | v_t = j]}, \tag{A.56}$$

where the probability $\text{Prob}[v_{t+1} = x | v_t = j]$ is given by the underlying first-order random walk and $\alpha(v_{t-1}, v_{t+1})$ is defined as follows using the shortest path distance $d(v_{t-1}, v_{t+1})$:

$$\alpha(v_{t-1}, v_{t+1}) \coloneqq \begin{cases} 1/p & \text{for } d(v_{t-1}, v_{t+1}) = 0, \\ 1 & \text{for } d(v_{t-1}, v_{t+1}) = 1, \\ 1/q & \text{for } d(v_{t-1}, v_{t+1}) = 2. \end{cases} \tag{A.57}$$

Choosing a large return parameter p reduces backtracking since it decreases the probability of walking from v_t to v_{t-1}. Choosing a small in-out parameter q has a similar effect of avoiding v_{t-1}, with a slight difference that it avoids the neighbors of v_{t-1} as well.

We now show an extension of Proposition 13 to the above second-order random walks:

Proposition 28. *The non-backtracking random walk in Equation* (A.55) *and the node2vec random walk in Equation* (A.56) *are invariant if their underlying first-order random walk algorithm is invariant.*

Proof. The proof can be found in Appendix A.7.4. □

A.7.2 Main algorithm

We outline the main algorithms for the random walk and the recording function described in Section 4.1.1 and used in Section 4.1.4. Algorithm 2 shows the random walk algorithm, and Algorithms 3, 4, and 5 show the recording functions. For the random walk algorithm, we use non-backtracking described

in Appendix A.7.1 and the minimum degree local rule conductance $c_G(\cdot)$ given in Equation (4.5) by default.

Based on the algorithms, in Figures A.6, A.7, and A.8 we illustrate the task formats used for graph separation experiments in Table 4.3 by providing examples of input graphs, text records of random walks on them, and prediction targets for the language model that processes the records. Likewise, in Figures A.9, A.10, and A.11 we show task formats for transductive classification on arXiv citation network in Section 4.1.4. Finally, in Figure A.12 we show task format used for Peptides-func graph classification experiment in Table 4.7.

Algorithm 2: Random walk algorithm

Data: Input graph G, optional input vertex v, conductance function $c_G : E(G) \to \mathbb{R}_+$, walk length l, optional restart probability $\alpha \in (0, 1)$ or period $k > 1$
Result: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$
`/* transition probabilities` $p : E(G) \to \mathbb{R}_+$ `*/`
for $(u, x) \in E(G)$ **do**
  $p(u, x) \leftarrow c_G(u, x) / \sum_{y \in N(u)} c_G(u, y)$ `// Equation (4.3)`
`/* starting vertex` v_0 `*/`
if v *is given* **then**
  `/* query starting vertex */`
  $v_0 \leftarrow v$
else
  `/* sample starting vertex */`
  $v_0 \sim \text{Uniform}(V(G))$
`/* random walk` $v_1 \to \cdots \to v_l$`, restart flags` $r_1, ..., r_l$ `*/`
$v_1 \sim \text{Categorical}(\{p(v_0, x) : x \in N(v_0)\})$
$r_1 \leftarrow 0$
for $t \leftarrow 2$ **to** l **do**
  `/* restart flag` r_t `*/`
  $r_t \leftarrow 0$
  if α *is given* **then**
    `/* if restarted at` $t - 1$`, do not restart */`
    if $r_{t-1} = 0$ **then**
      $r_t \sim \text{Bernoulli}(\alpha)$
  else if k *is given* **then**
    if $t \equiv 0 \pmod{k}$ **then**
      $r_t \leftarrow 1$
  `/* random walk` v_t `*/`
  if $r_t = 0$ **then**
    $S \leftarrow N(v_{t-1}) \setminus \{v_{t-2}\}$
    if $S \neq \emptyset$ **then**
      `/* non-backtracking */`
      $v_t \sim \text{Categorical}(\{p(v_{t-1}, x) / \sum_{y \in S} p(v_{t-1}, y) : x \in S\})$ `// Equation (A.55)`
    else
      `/* begrudgingly backtracking */`
      $v_t \leftarrow v_{t-2}$
  else
    `/* restart */`
    $v_t \leftarrow v_0$

Algorithm 3: Recording function, using anonymization

Data: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$
Result: Text record $\mathbf{z}$
`/* named vertices` S`, namespace` $\mathrm{id}(\cdot)$ `*/`
$S \leftarrow \{v_0\}$
$\mathrm{id}(v_0) \leftarrow 1$
`/* text record` $\mathbf{z}$ `*/`
$\mathbf{z} \leftarrow \mathrm{id}(v_0)$
for $t \leftarrow 1$ **to** l **do**
  `/* anonymization` $v_t \mapsto \mathrm{id}(v_t)$ `*/`
  if $v_t \notin S$ **then**
    $S \leftarrow S \cup \{v_t\}$
    $\mathrm{id}(v_t) \leftarrow |S|$
  `/* text record` $\mathbf{z}$ `*/`
  if $r_t = 0$ **then**
    `/* record walk` $v_{t-1} \to v_t$ `*/`
    $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"-"} + \mathrm{id}(v_t)$
  else
    `/* record restart` $v_t = v_0$ `*/`
    $\mathbf{z} \leftarrow \mathbf{z} + \texttt{";"} + \mathrm{id}(v_t)$

Algorithm 4: Recording function, using anonymization and named neighbors

Data: Random walk $v_0 \to \cdots \to v_l$, restart flags $r_1, ..., r_l$, input graph G
Result: Text record $\mathbf{z}$
`/* named vertices` S`, namespace` $\mathrm{id}(\cdot)$`, recorded edges` T `*/`
$S \leftarrow \{v_0\}$
$\mathrm{id}(v_0) \leftarrow 1$
$T \leftarrow \emptyset$
`/* text record` $\mathbf{z}$ `*/`
$\mathbf{z} \leftarrow \mathrm{id}(v_0)$
for $t \leftarrow 1$ **to** l **do**
  `/* anonymization` $v_t \mapsto \mathrm{id}(v_t)$ `*/`
  if $v_t \notin S$ **then**
    $S \leftarrow S \cup \{v_t\}$
    $\mathrm{id}(v_t) \leftarrow |S|$
  `/* text record` $\mathbf{z}$ `*/`
  if $r_t = 0$ **then**
    `/* record walk` $v_{t-1} \to v_t$ `*/`
    $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"-"} + \mathrm{id}(v_t)$
    $T \leftarrow T \cup \{(v_{t-1}, v_t), (v_t, v_{t-1})\}$
    `/* named neighbors` U `*/`
    $U \leftarrow N(v_t) \cap S$
    if $U \neq \emptyset$ **then**
      `/* sort in ascending order` $\mathrm{id}(u_1) < \cdots < \mathrm{id}(u_{|U|})$ `*/`
      $[u_1, ..., u_{|U|}] \leftarrow \mathrm{SortByKey}(U, \mathrm{id})$
      for $u \leftarrow u_1$ **to** $u_{|U|}$ **do**
        `/* record named neighbors` $v_t \to u$ `*/`
        if $(v_t, u) \notin T$ **then**
          $\mathbf{z} \leftarrow \mathbf{z} + \texttt{"\#"} + \mathrm{id}(u)$
          $T \leftarrow T \cup \{(v_t, u), (u, v_t)\}$
  else
    `/* record restart` $v_t = v_0$ `*/`
    $\mathbf{z} \leftarrow \mathbf{z} + \texttt{";"} + \mathrm{id}(v_t)$

Algorithm 5: Recording function for transductive classification on arXiv network

Data: Directed input graph G, random walk $v_0 \to \cdots \to v_l$ and restart flags $r_1, ..., r_l$ on the undirected copy of G, paper titles $\{\mathbf{t}_v\}_{v \in V(G)}$ and abstracts $\{\mathbf{a}_v\}_{v \in V(G)}$, target category labels $\{\mathbf{y}_v\}_{v \in L}$ for labeled vertices $L \subset V(G)$

Result: Text record $\mathbf{z}$

`/* named vertices` S`, namespace` $\mathrm{id}(\cdot)$`, recorded edges` T `*/`

$S \leftarrow \{v_0\}$

$\mathrm{id}(v_0) \leftarrow 1$

$T \leftarrow \emptyset$

`/* text record` $\mathbf{z}$ `*/`

`/* record starting vertex */`

$\mathbf{z} \leftarrow$ `"Paper 1 - Title: {`$\mathbf{t}_{v_0}$`}", Abstract: {`$\mathbf{a}_{v_0}$`}"`

for $t \leftarrow 1$ **to** l **do**

  `/* anonymization` $v_t \mapsto \mathrm{id}(v_t)$ `*/`

  if $v_t \notin S$ **then**

    $S \leftarrow S \cup \{v_t\}$

    $\mathrm{id}(v_t) \leftarrow |S|$

  `/* text record` $\mathbf{z}$ `*/`

  if $r_t = 0$ **then**

    `/* record walk` $v_{t-1} \to v_t$ `with direction */`

    if $(v_{t-1}, v_t) \in E(G)$ **then**

      $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\mathrm{id}(v_{t-1})$`} cites Paper {`$\mathrm{id}(v_t)$`}"`

    else

      $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\mathrm{id}(v_{t-1})$`} is cited by Paper {`$\mathrm{id}(v_t)$`}"`

    $T \leftarrow T \cup \{(v_{t-1}, v_t), (v_t, v_{t-1})\}$

    `/* record title` $\mathbf{t}_v$`, abstract` $\mathbf{a}_v$`, and label` $\mathbf{y}_v$ `if` v `is labeled */`

    if $\mathrm{id}(v_t) = |S|$ **then**

      $\mathbf{z} \leftarrow \mathbf{z} +$ `" - {`$\mathbf{t}_v$`}"`

      if $v \in L$ **then**

        $\mathbf{z} \leftarrow \mathbf{z} +$ `", Category: {`$\mathbf{y}_v$`}"`

      $\mathbf{z} \leftarrow \mathbf{z} +$ `", Abstract: {`$\mathbf{a}_v$`}"`

    else

      $\mathbf{z} \leftarrow \mathbf{z} +$ `"."`

    `/* named neighbors` U `*/`

    $U \leftarrow N(v_t) \cap S$

    if $U \neq \emptyset$ **then**

      `/* sort in ascending order` $\mathrm{id}(u_1) < \cdots < \mathrm{id}(u_{|U|})$ `*/`

      $[u_1, ..., u_{|U|}] \leftarrow \mathrm{SortByKey}(U, \mathrm{id})$

      for $u \leftarrow u_1$ **to** $u_{|U|}$ **do**

        `/* record named neighbors` $v_t \to u$ `with directions */`

        if $(v_t, u) \notin T$ **then**

          if $(v_t, u) \in E(G)$ **then**

            $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\mathrm{id}(v_t)$`} cites Paper {`$\mathrm{id}(u)$`}."`

          else

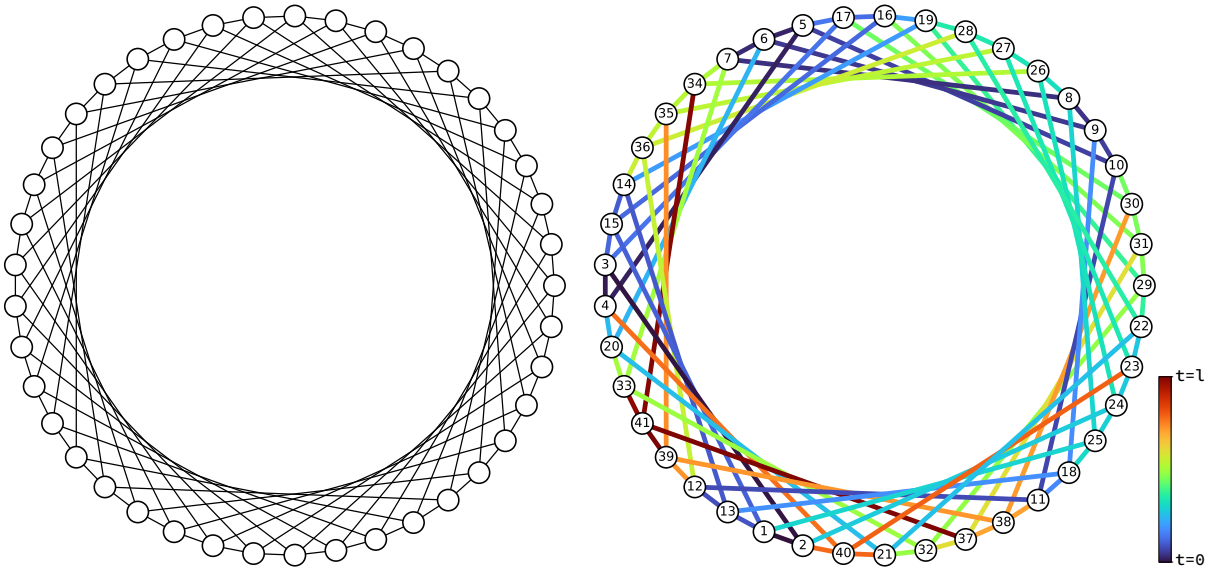
            $\mathbf{z} \leftarrow \mathbf{z} +$ `" Paper {`$\mathrm{id}(v_t)$`} is cited by Paper {`$\mathrm{id}(u)$`}."`

          $T \leftarrow T \cup \{(v_t, u), (u, v_t)\}$

  else

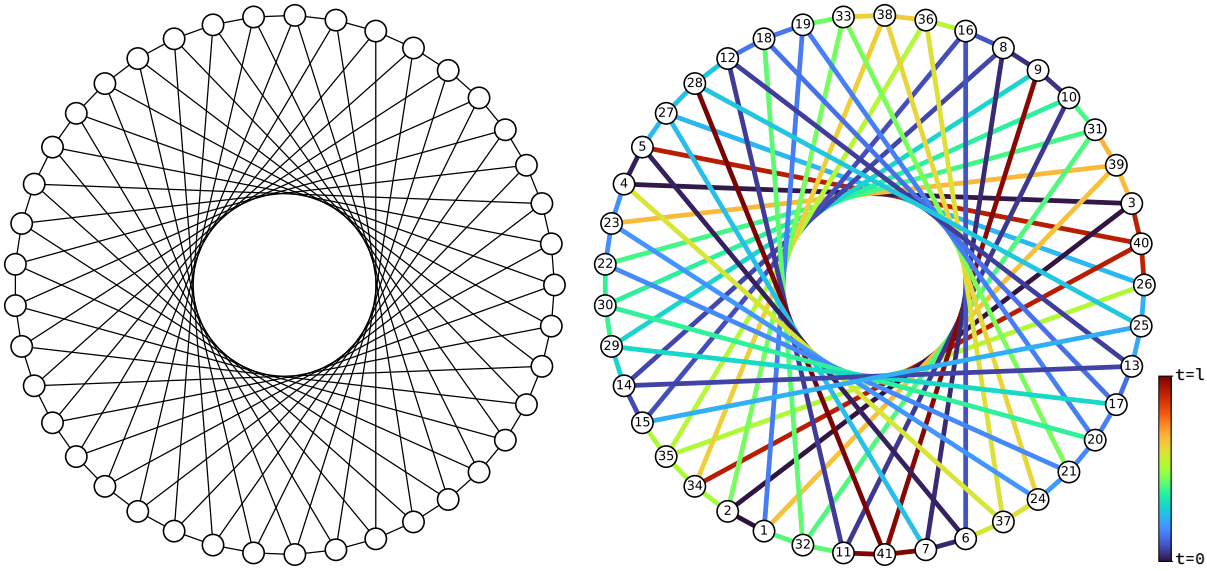
    `/* record restart */`

    $\mathbf{z} \leftarrow \mathbf{z} +$ `" Restart at Paper 1."`



Graph | Random Walk

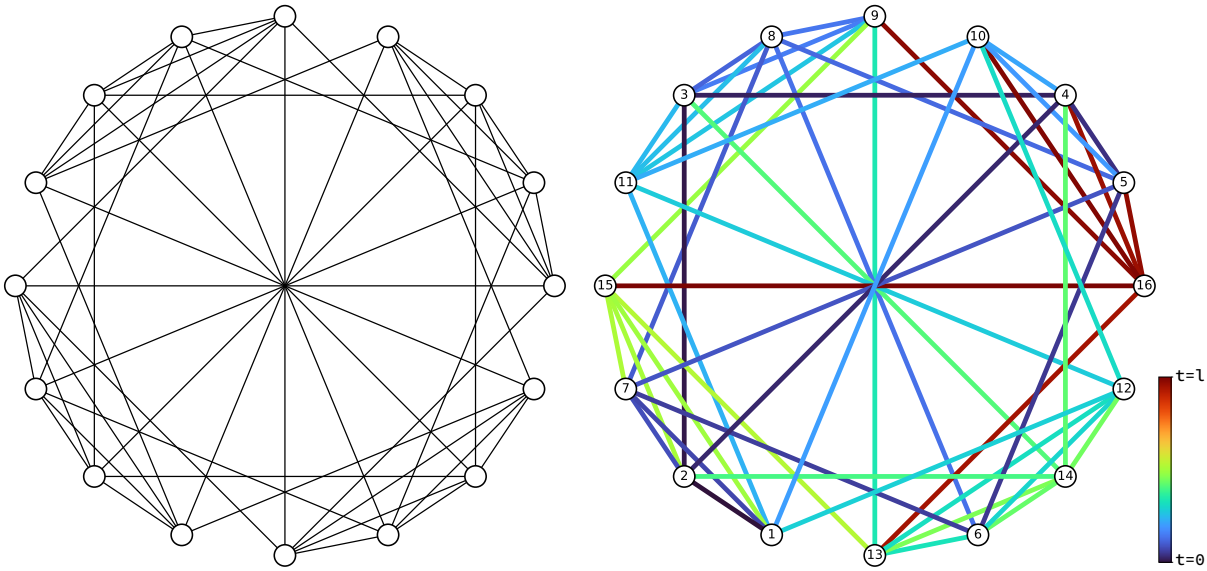
Record	[CLS]1-2-3-4-5-6-7-8-9#6-10#5-11-12-13#1-14-15#1#3-16-17#3#5-5-10-11-18#9#13-13-14-19#16-16-17-5-6-20#4-21-22-23-24#2-25#1#8#18-8-26#24-27#23-28#19#22-22-29#19-19-14-13-18-11-10-5-17-30#10-31#16#29-29-32#21-33#7#20-7-34#26-35#27-36#12#14#28-28-22-29-31-37#32-32-33-7-6-5-4-20-6-5-4-20-33-32-37-38#11#30-39#12#35-12-11-10-5-4-40#2#21#23-2-24-23-27-28-19-16-17-30-10-5-4-40-2-3-4-5-17-3-2-40-4-20-33-32-21-20-33-41#34#37#39-39-38-37-31-29-32-33-41-34-35-36-28-27-35-39-12-13-1-15-3-4-20-6-7-33-41-34-35-39-41-37-32-33-20-21-40-23-22-29-31-37-41-34-35-39-12-13-18-25-24-26-34-7-33-20-4-40-23-27-35-36-12-11-10-30-31-16-15-3-4-40-23-22-28-36-14-19-29-31-37-41-33-32-29-22-21-32-33-20-4-40-21-22-29-19-14-15-16-31-29-[SEP]
Answer	csl(41, 9)



Graph | Random Walk

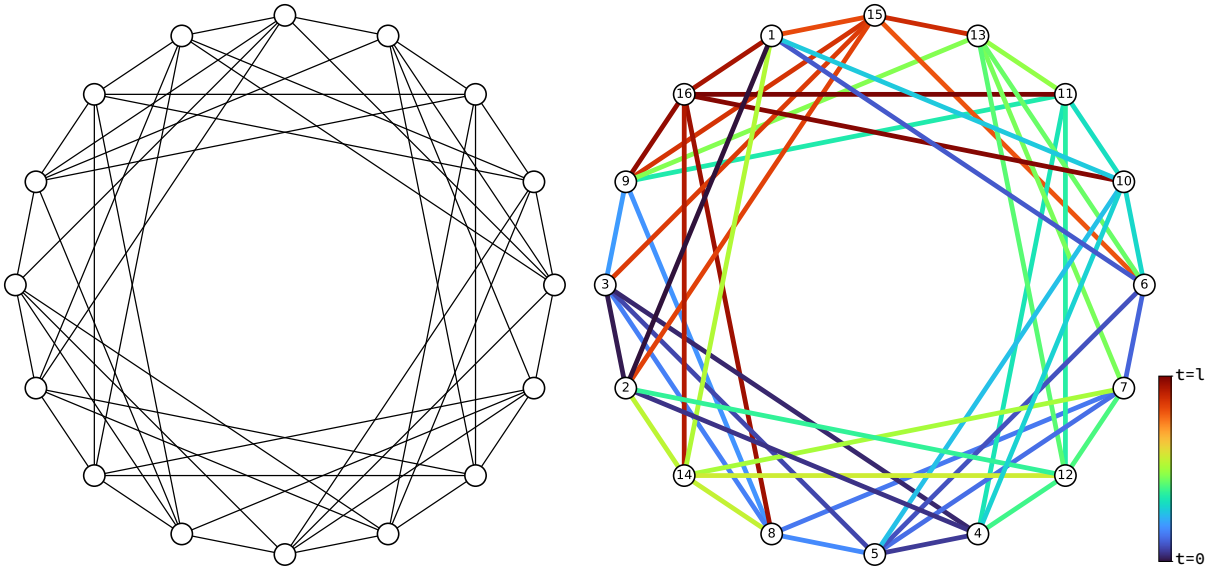
Record	[CLS]1-2-3-4-5-6-7-8-9-10-11-12-13-14#8-15-16#6#8-6-7-8-14-13-17-18#12-19#1-20#17-21-22-23#4-24#21-21-20-17-13-25#15-26-27#5#7-28#12#25-12-13-14-29#9#17-17-13-25-28-12-13-14-8-9-10-30#20#22#29-22-31#10-32#1#11#18-1-19-33#2#21-21-22-31-32-1-19-33-2-34-35#15#26-36#16-16-15-25-28-27-7-6-37#4#24#36-4-23-24-38#33#34#36-34-2-1-39#3#23#31-23-24-38-33-19-1-2-3-39-23-24-21-33-38-24-23-4-37-24-23-39-1-2-34-38-36-37-6-5-4-23-39-3-4-5-6-16-15-35-26-40#3#5#34-5-6-7-27-28-12-13-17-29-9-41#7#11#28-7-27-26-25-13-12-11-32-18-12-13-14-15-25-26-40-34-38-36-37-24-23-4-37-24-21-33-2-3-4-5-27-7-6-16-15-35-34-2-3-40-26-35-34-38-33-21-20-17-29-14-15-16-8-9-41-28-27-7-8-14-15-25-28-27-5-6-7-41-9-8-7-41-9-10-31-[SEP]
Answer	csl(41, 16)

Figure A.6: Two CSL graphs and text records of random walks from Algorithms 2 and 4. Task is graph classification into 10 isomorphism types csl$(41, s)$ for skip length $s \in \{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$. In random walks, we label vertices by anonymization and color edges by their time of discovery.



Graph | Random Walk

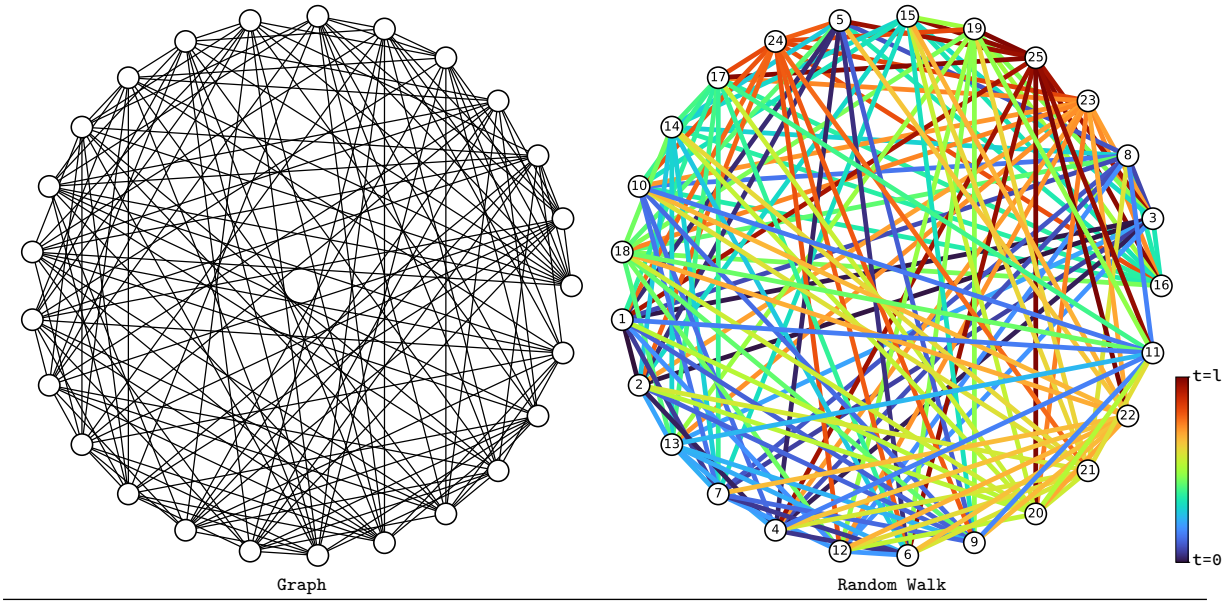
Record	[CLS]1-2-3-4#2-5-6-7#1#2#5-8#3#5#6-9#3-3-4-5-10#1#4-11#1#3#8#9-12#1#6#10-13#6#9-6-5-8-7-2-14#3#4#6#12#13-13-9-15#1#2#7#13-7-6-13-12-1-2-4-10-12-11-1-7-15-1-11-12-10-4-14-13-15-2-7-8-9-3-2-7-8-5-7-2-3-14-13-15-1-2-14-13-16#4#5#9#10#15-5-10-1-11-3-14-4-16-10-1-12-13-6-12-13-6-12-10-11-12-10-11-1-7-6-13-15-16-5-4-14-3-9-13-16-15-1-12-13-9-15-7-2-14-13-6-5-7-2-4-3-9-15-13-14-6-5-7-8-9-11-10-5-16-4-3-14-2-15-13-12-10-5-16-4-14-13-12-14-13-16-10-5-6-14-3-9-16-13-9-15-1-2-15-7-1-12-13-6-8-5-7-6-12-1-11-12-14-4-2-15-9-8-7-1-15-16-4-14-12-1-10-5-16-9-8-5-16-13-12-6-14-2-1-7-15-16-4-2-1-15-16-10-4-16-15-9-16-15-13-6-8-9-16-5-6-[SEP]
Answer	4×4 rook's graph



Graph | Random Walk

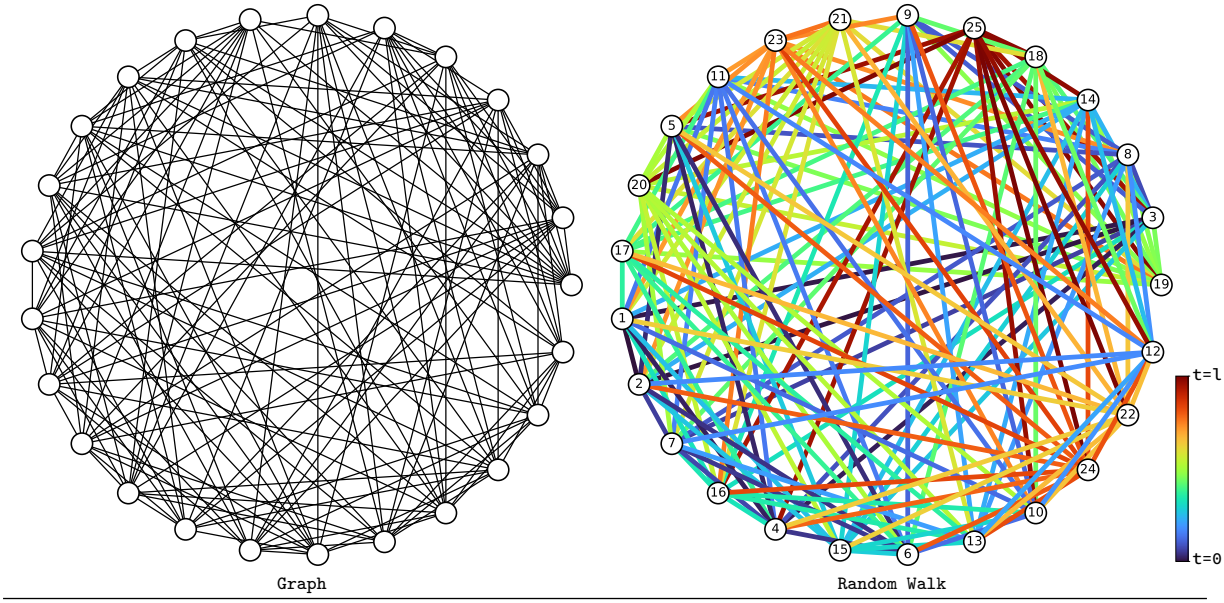
Record	[CLS]1-2-3-4#2-5#3-6#1-7#5-8#3#5-9#3-3-4-5-10#1#4#6-11#4#9-12#2#4#7-13#6#7#9#11-7-14#1#2#8#12-8-5-6-7-13-9-11-4-5-3-4-12-14-1-10-6-15#1#2#3#9#13-2-14-16#1#8#9#10#11-10-11-4-10-6-1-16-8-7-5-8-9-3-5-7-8-14-7-6-15-2-3-8-14-2-1-16-11-4-10-6-15-13-6-1-16-10-6-1-16-14-12-13-7-12-4-3-2-12-11-4-5-7-13-15-3-5-6-1-15-9-13-11-16-10-6-13-9-11-4-12-7-13-6-7-5-6-1-15-9-11-13-6-5-4-2-3-9-15-1-14-16-1-15-6-7-13-15-1-10-5-3-2-1-16-10-4-11-16-10-5-4-12-11-9-16-11-9-13-12-14-16-1-10-4-11-12-13-6-5-7-12-2-3-5-4-10-6-13-9-8-5-10-16-11-12-2-1-6-10-5-3-9-8-14-16-11-10-5-6-7-12-14-16-11-12-14-1-16-11-10-4-11-13-9-16-8-3-5-8-9-16-14-2-[SEP]
Answer	Shrikhande graph

Figure A.7: SR16 graphs and text records of random walks from Algorithms 2 and 4. The task is graph classification into two isomorphism types $\{4 \times 4$ rook's graph, Shrikhande graph$\}$. In random walks, we label vertices by anonymization and color edges by their time of discovery.



Graph | Random Walk

Record	[CLS]1-2-3#1-4#2-5#1#2-6#4-7#1#3-3-8#1#4#5#7-5-9#1#2#7-10#4#7#8-11#1#4#8#9-8-12#1#3#6#7-3-13#2#4#6#7#9#11-2-3-7-6-14#2#5#7#8#13-15#2#3#7#9#10-16#3#5#8#10#14-17#1#6#7#10#11#13#14-13-18#5#8#9#11#12#14#16-19#3#6#9#11#12#13#15#16-9-7-13-11-20#1#2#12#14#15#17#18-21#2#4#6#8#10#11#12#14#15#19-8-7-14-5-22#4#6#7#9#10#12#15#18#20-23#2#3#4#10#12#13#16#17#18#20-13-9-2-21-24#1#2#5#6#9#10#12#16#17#19#23-19-6-24-23-16-3-1-5-8-21-14-2-3-7-8-5-6-7-1-25#3#4#5#6#11#15#16#17#19#20#22-17-1-11-4-3-15-20-2-5-22-12-21-14-7-9-5-4-22-20-25-11-21-8-1-24-17-10-23-18-13-6-4-8-3-7-17-14-20-12-24-16-17-10-21-14-5-4-13-14-7-13-3-8-1-3-7-14-13-2-3-8-18-22-5-[SEP]
Answer	Graph 1



Graph | Random Walk

Record	[CLS]1-2-3#1-4#1-5#2-6#2#4-7#2#3#4-3-8#4#5#7-9#3#6-10#1#6-11#1#4#6#7#8-12#2#7#8#10-8-13#7#9#12-14#1#2#3#11#12-15#1#2#3#5#6#8#9#10#13-1-3-16#1#4#7#9#10#13-13-17#1#4#6#9#11#14-18#2#3#6#7#9#10#12-19#3#5#8#9#11#14#17-11-7-20#5#6#10#11#13#14#15#16#19-21#1#2#5#7#13#16#17#18#19-16-13-15-22#1#4#5#8#10#12#13#17#18#21-10-6-18-7-11-23#1#2#5#8#9#10#12#16#19#21-12-8-4-24#2#5#6#9#12#13#14#16#17#23-16-10-23-9-6-5-8-19-3-1-23-8-22-5-2-3-16-25#3#4#5#10#12#14#18#19#20#22#24-14-13-16-1-21-17-1-22-15-3-9-13-16-21-22-15-10-11-4-24-9-6-10-12-14-24-12-8-7-11-20-5-25-20-7-6-4-8-3-16-21-20-13-15-5-19-23-5-6-17-9-6-7-11-4-7-3-8-7-3-16-23-2-1-3-8-5-24-23-[SEP]
Answer	Graph 8

Figure A.8: Two SR25 graphs and text records of random walks from Algorithms 2 and 4. The task is graph classification into 15 isomorphism types. In random walks, we label vertices by anonymization and color edges by their time of discovery.

System	A walk on the arXiv citation network will be given. Predict the arXiv CS sub-category Paper 1 belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM). Only respond with the answer, do not say any word or explain.
Record	A walk on arXiv citation network is as follows: Paper 1 - Title: Safety guided deep reinforcement learning via online gaussian process estimation, Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Restart at 1. Paper 1 is cited by 2 - Learning to walk in the real world with minimal human effort, Abstract: Reliable and stable locomotion has been one of the most fundamental challenges for legged robots. deep reinforcement learning (deep rl) has emerged as a promising method for developing such control policies... Restart at 1. Paper 1 cites 3 - Deep q learning from demonstrations, Category: Artificial Intelligence (cs.AI), Abstract: Deep reinforcement learning (rl) has achieved several high profile successes in difficult decision-making problems. however, these algorithms typically require a huge amount of data before they reach... Restart at 1. Paper 1 cites 4 - Constrained policy optimization, Category: Machine Learning (cs.LG), Abstract: For many applications of reinforcement learning it can be more convenient to specify both a reward function and constraints, rather than trying to design behavior through the reward function. for example,... Paper 4 is cited by 2. Paper 4 is cited by 5 - Artificial intelligence values and alignment, Abstract: This paper looks at philosophical questions that arise in the context of ai alignment. it defends three propositions. first, normative and technical aspects of the ai alignment problem are interrelated,... Paper 5 cites 6 - Agi safety literature review, Category: Artificial Intelligence (cs.AI), Abstract: The development of artificial general intelligence (agi) promises to be a major event. along with its many potential benefits, it also raises serious safety concerns (bostrom, 2014). the intention of... Paper 6 is cited by 7 - Modeling agi safety frameworks with causal influence diagrams, Abstract: Proposals for safe agi systems are typically made at the level of frameworks, specifying how the components of the proposed system should be trained and interact with each other. in this paper, we model... Restart at 1. Paper 1 cites 8 - Safe learning of regions of attraction for uncertain nonlinear systems with gaussian processes, Category: Systems and Control (cs.SY), Abstract: Control theory can provide useful insights into the properties of controlled, dynamic systems. one important property of nonlinear systems is the region of attraction (roa), a safe subset of the state... Paper 8 is cited by 9 - Control theory meets pomdps a hybrid systems approach, Abstract: Partially observable markov decision processes (pomdps) provide a modeling framework for a variety of sequential decision making under uncertainty scenarios in artificial intelligence (ai). since the... Restart at 1. ... Which arXiv CS sub-category does Paper 1 belong to? Only respond with the answer, do not say any word or explain.
Answer	Machine Learning (cs.LG)

Figure A.9: Transductive classification on arXiv citation network using text record of random walk from Algorithms 2 and 5. Colors indicate task instruction, walk information, and label information.

System	Title and abstract of an arXiv paper will be given. Predict the arXiv CS sub-category the paper belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM). Only respond with the answer, do not say any word or explain.
Context	Title: Safety guided deep reinforcement learning via online gaussian process estimation Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Which arXiv CS sub-category does this paper belong to?
Answer	Machine Learning (cs.LG)

Figure A.10: Zero-shot format for vertex classification on arXiv citation network. Task instruction and label information are colored.

System	Title and abstract of an arXiv paper will be given. Predict the arXiv CS sub-category the paper belongs to. It is one of the following: Numerical Analysis (cs.NA), Multimedia (cs.MM), Logic in Computer Science (cs.LO), ... Discrete Mathematics (cs.DM).
Context	Title: Deeptrack learning discriminative feature representations online for robust visual tracking Abstract: Deep neural networks, albeit their great success on feature learning in various computer vision tasks, are usually considered as impractical for online visual tracking, because they require very long... Category: cs.CV Title: Perceived audiovisual quality modelling based on decison trees genetic programming and neural networks... Abstract: Our objective is to build machine learning based models that predict audiovisual quality directly from a set of correlated parameters that are extracted from a target quality dataset. we have used the... Category: cs.MM ... Title: Safety guided deep reinforcement learning via online gaussian process estimation Abstract: An important facet of reinforcement learning (rl) has to do with how the agent goes about exploring the environment. traditional exploration strategies typically focus on efficiency and ignore safety. however, for practical applications, ensuring safety of the agent during exploration is crucial since performing an unsafe action or reaching an unsafe state could result in irreversible damage to the agent. the main challenge of safe exploration is that characterizing the unsafe states and actions... Which arXiv CS sub-category does this paper belong to?
Answer	cs.LG

Figure A.11: One-shot format for transductive classification on arXiv citation network. 40 labeled examples are given in form of multi-turn dialogue. Task instruction and label information are colored.

Record	1[O,sp2]=π2[C,d:3,sp2,r]-3[C,cw,d:4,H:1,sp3,r]-4[N,d:3,H:1,sp2,r]-π5[C,d:3,sp2,r]=π6[O,sp2]=π5-π4-3-2=π1=π2-3-7[C,d:4,H:2,sp3,r]-8[S,d:2,sp3,r]-9[S,d:2,sp3,r]-10[C,d:4,H:2,sp3,r]-11[C,ccw,d:4,H:1,sp3,r]-12[N,d:3,H:1,sp2,r]-π13[C,d:3,sp2,r]=π14[O,sp2]=π13-π12-11-15[C,d:3,sp2,r]-π16[N,d:3,H:1,sp2,r]-17[C,cw,d:4,H:1,sp3,r]-18[C,d:3,sp2,r]=π19[O,sp2]=π18-17-20[C,d:4,H:2,sp3]-21[C,d:4,H:2,sp3]-22[C,d:3,sp2]=π23[O,sp2]=π22-π24[O,d:2,H:1,sp2]-π22=π23=π22-π24-π22=π23=π22-21-20-17-16-π15=π25[O,sp2]=π15-π16-17-20-21-22-π24-π22=π23=π22-21-20-17-16-π15=π25=π15-π16-17-20-21-22-π24-π22=π23=π22-π24-π22-21-20-17-18=π19=π18-π26[N,d:3,H:1,sp2,r]-27[C,ccw,d:4,H:1,sp3,r]-28[C,d:3,sp2]=π29[O,sp2]=π28-π30[N,d:3,H:1,sp2]-31[C,ccw,d:4,H:1,sp3]-32[C,d:4,H:1,sp3]-33[C,d:4,H:3,sp3]-32-34[C,d:4,H:3,sp3]-32-33-32-34-32-33-32-34-32-33-32-31-30-π28=π29=π28-π30-31-32-34-32-31-30-π28=π29=π28-π30-31-32-33-32-34-32-33-32-34-32-31-30-π28=π29=π28-π30-31-35[C,d:3,sp2]=π36[O,sp2]=π35-π37[N,d:3,H:1,sp2]-38[C,cw,d:4,H:1,sp3]-39[C,d:3,sp2]-π40[N,d:3,H:1,sp2]-41[C,cw,d:4,H:1,sp3]-42[C,d:4,H:2,sp3]-43[C,d:3,sp2,*,r]∗π44[C,d:3,H:1,sp2,*,r]∗π45[N,d:3,H:1,sp2,*,r]∗π46[C,d:3,H:1,sp2,*,r]∗π47[N,d:2,sp2,*,r]#∗π43∗π43∗π44∗π45∗π46∗π47∗π43-42-41-40-π39=π48[O,sp2]=π39-38-37-π35=π36=π35-π37-38-49[C,d:4,H:2,sp3]-50[C

Figure A.12: An example text record for Peptides-func graph classification.

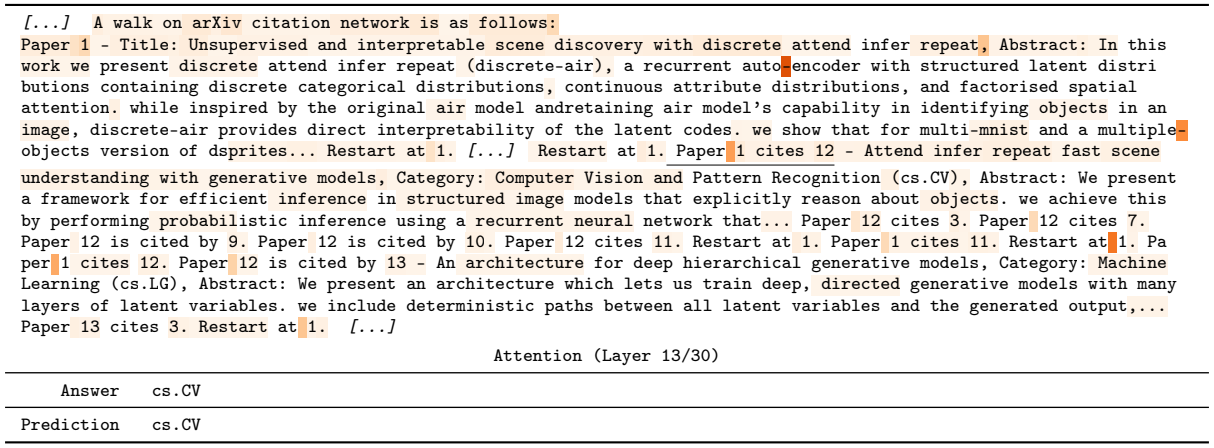


Figure A.13: Example of attention in a frozen Llama 3 8B applied on arXiv transductive classification. Attention weights are averaged over all 30 heads.

A.7.3 Attention visualizations

In Figure A.13, we show an example of self-attention in the frozen Llama 3 8B model applied to arXiv transductive classification (Section 4.1.4). We show attention weights on text record of random walk from the generated `cs` token as query, which is just before finishing the prediction, e.g., `cs.CV`. We color the strongest activation with orange, and do not color values below 1% of it. The model invests a nontrivial amount of attention on walk information such as `Paper 1 cites 12`, while also making use of titles and labels of labeled vertices. This indicates the model is utilizing the graph structure recorded by the random walk in conjunction with other information to make predictions.

In Figures A.14, A.15, and A.16, we show examples of self-attention in DeBERTa models trained for graph separation (Table 4.3). We first show attention weights on text record of random walk from the `[CLS]` query token. Then, we show an alternative visualization where the attention weights are mapped onto the original input graph. For example, for each attention on v_{t+1} where the walk has traversed $v_t \to v_{t+1}$, we regard it as an attention on the edge (v_t, v_{t+1}) in the input graph. We ignore `[CLS]` and `[SEP]` tokens since they do not appear on the input graph. We observe that the models often focus on sparse, connected substructures, which presumably provide discriminative information on the isomorphism types. In particular, for CSL graphs (Figure A.14) we observe an interpretable pattern of approximate cycles composed of skip links. This is presumably related to measuring the lengths of skip links, which provides sufficient information of isomorphism types of CSL graphs.

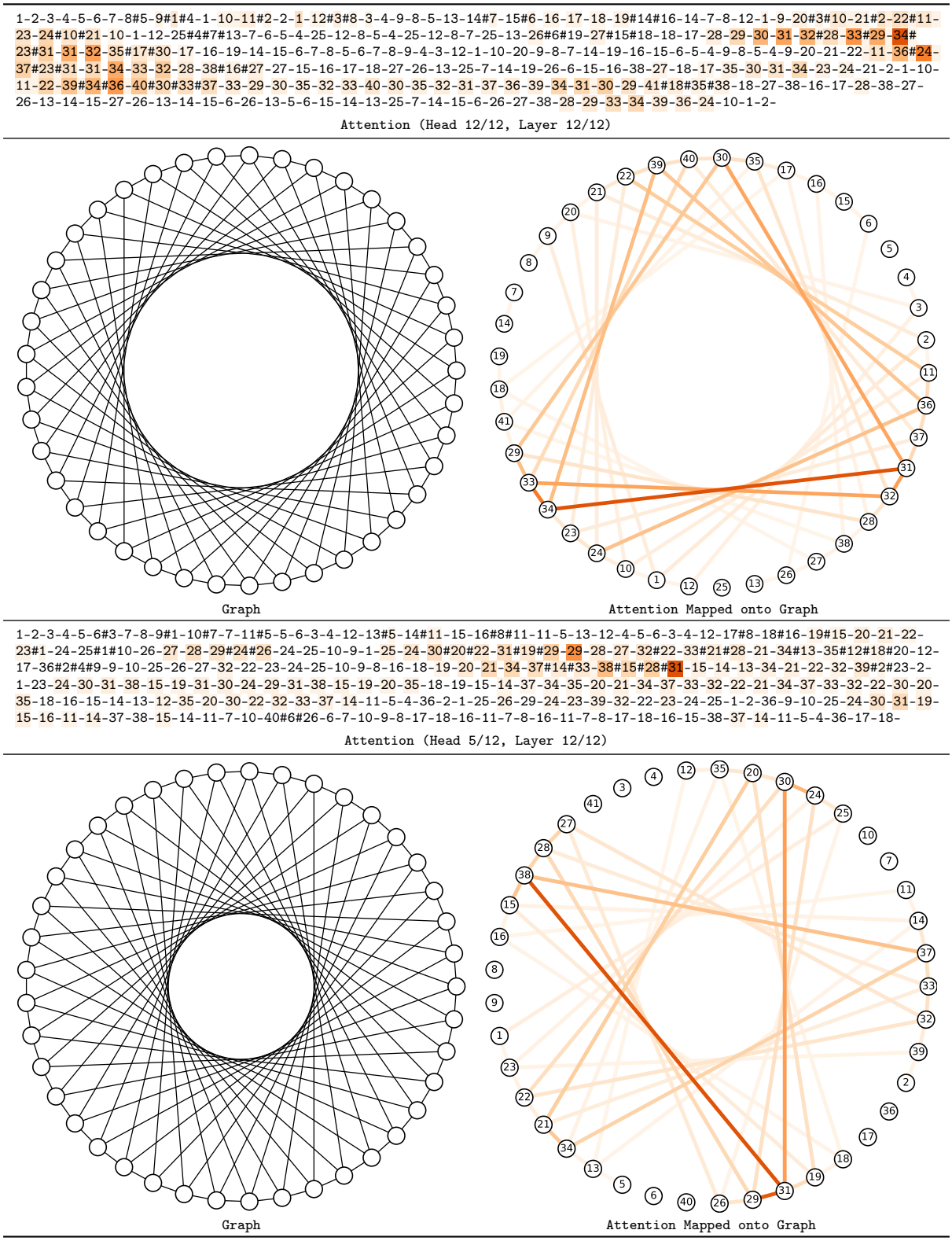


Figure A.14: Examples of attention from [CLS] token in a DeBERTa trained on CSL graph separation, forming approximate cycles using skip links. This is presumably related to measuring the lengths of skip links, which provides sufficient information to infer isomorphism types of CSL graphs.

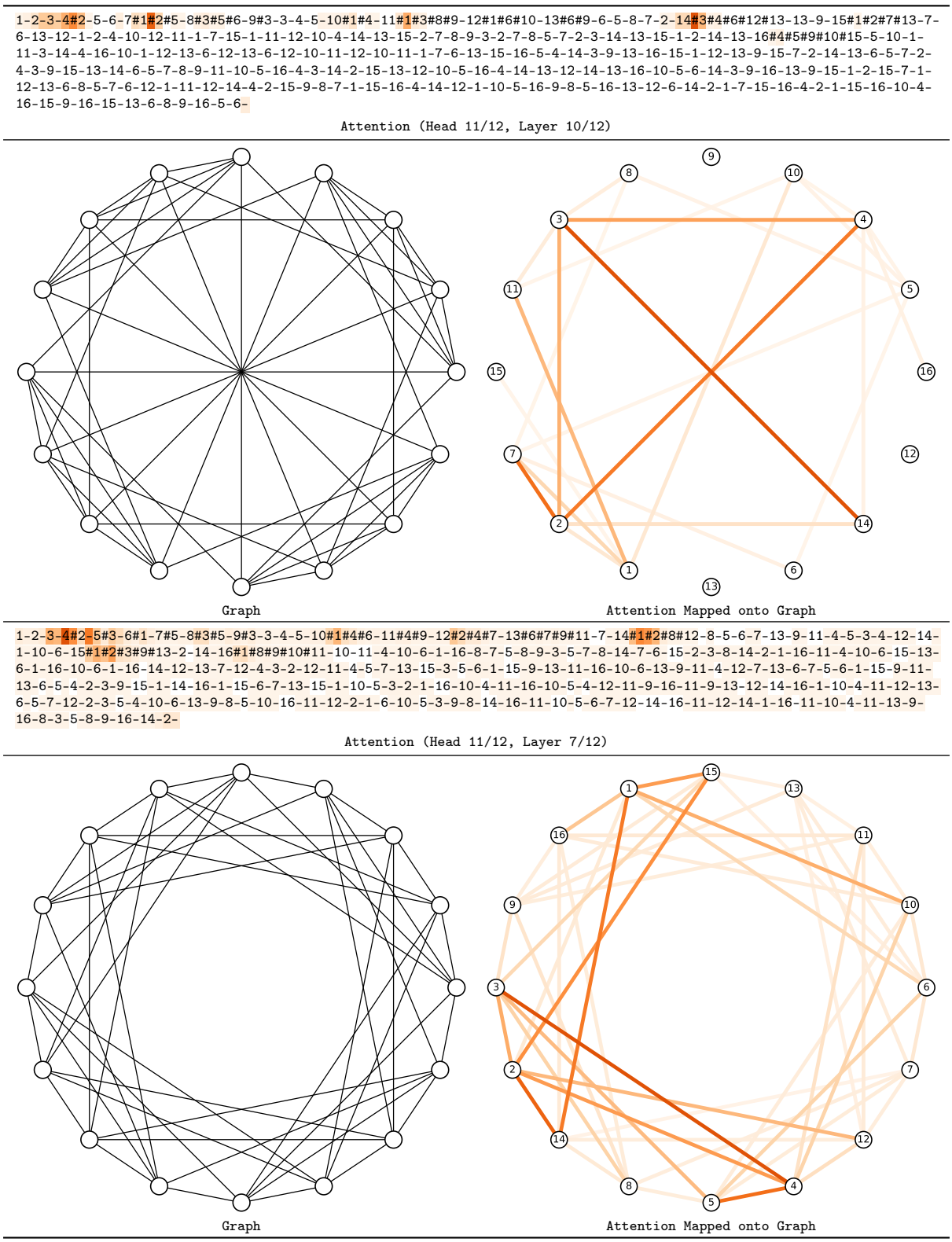


Figure A.15: Examples of attention from [CLS] token in a DeBERTa trained on SR16 graph separation. The model tend to focus on named neighbor records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR16 graphs.

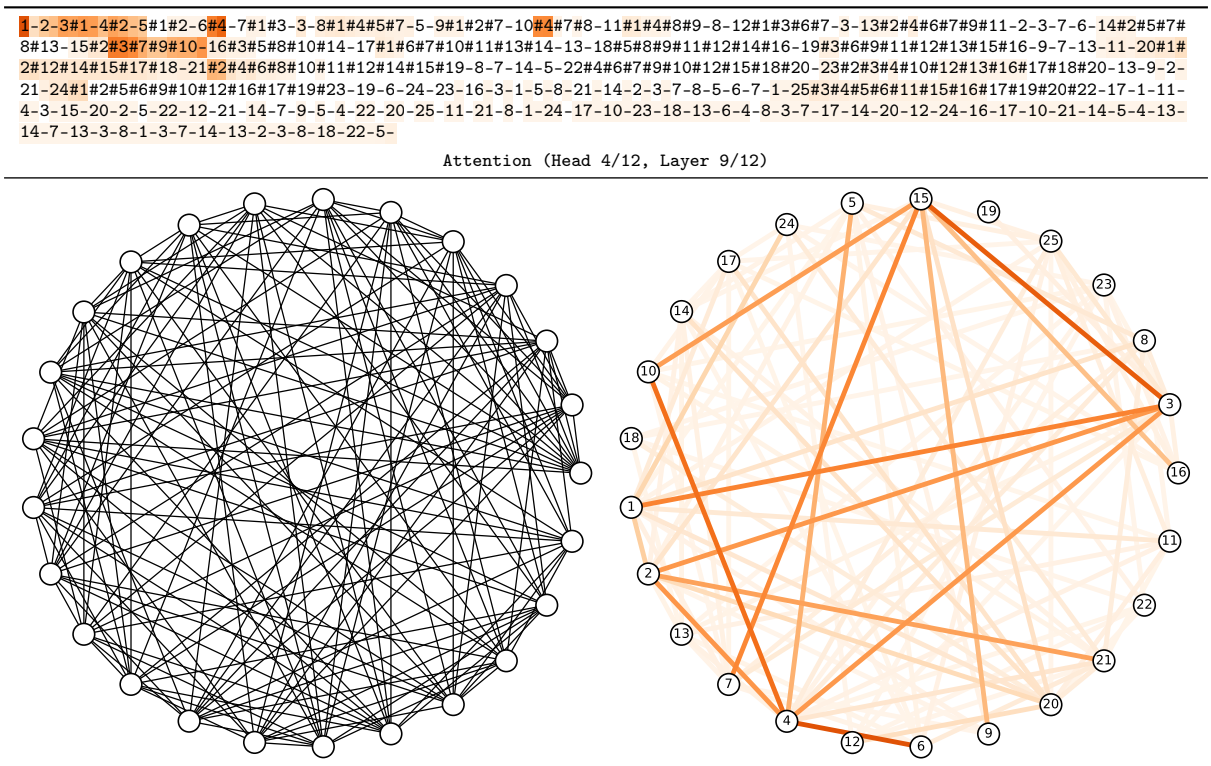


Figure A.16: Examples of attention from [CLS] token in a DeBERTa trained on SR25 graph separation. The model tend to focus on named neighbor records that form a sparse connected substructure, which we conjecture to provide discriminative information on the isomorphism type of SR25 graphs.

A.7.4 Proofs

We recall that an isomorphism between two graphs G and H is a bijection $\pi : V(G) \to V(H)$ such that any two vertices u and v are adjacent in G if and only if $\pi(u)$ and $\pi(v)$ are adjacent in H. If an isomorphism π exists between G and H, the graphs are isomorphic and written $G \simeq H$ or $G \overset{\pi}{\simeq} H$. By $X \overset{d}{=} Y$ we denote the equality of two random variables X and Y in distributions.

In the proofs, we write by $\mathrm{id}(\cdot)$ the namespace obtained by anonymization of a walk $v_0 \to \cdots \to v_l$ (Section 4.1.1) that maps each vertex v_t to its unique integer name $\mathrm{id}(v_t)$ starting from $\mathrm{id}(v_0) = 1$.

Let us define some notations related to cover times. These will be used to prove Theorem 14. We denote by $H(u, v)$ the expected number of steps a random walk starting from u takes until reaching v. This is called the hitting time between u and v. For a graph G, let $C_V(G, u)$ be the expected number of steps a random walk starting from u takes until visiting every vertices of G. The vertex cover time $C_V(G)$, or the cover time, is this quantity given by the worst possible u:

$$C_V(G) \coloneqq \max_{u \in V(G)} C_V(G, u). \tag{A.58}$$

Likewise, let $C_E(G, u)$ be the expected number of steps a random walk starting from u takes until traversing every edge of G. The edge cover time $C_E(G)$ is given by the worst possible u:

$$C_E(G) \coloneqq \max_{u \in V(G)} C_E(G, u). \tag{A.59}$$

For local balls $B_r(u)$, we always set the starting vertex of the random walk to u (Section 4.1.1). Thus, we define their cover times as follows:

$$C_V(B_r(u)) \coloneqq C_V(B_r(u), u), \tag{A.60}$$

$$C_E(B_r(u)) \coloneqq C_E(B_r(u), u). \tag{A.61}$$

Proof of Proposition 12

Proof. We recall the probabilistic invariance of random walk algorithm in Equation (4.2):

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \qquad \forall G \overset{\pi}{\simeq} H,$$

where $v_{[\cdot]}$ is a random walk on G and $u_{[\cdot]}$ is a random walk on H. We further recall the invariance of recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$ in Equation (4.6):

$$q(v_0 \to \cdots \to v_l, G) = q(\pi(v_0) \to \cdots \to \pi(v_l), H), \qquad \forall G \overset{\pi}{\simeq} H,$$

for any given random walk $v_{[\cdot]}$ on G. Combining Equation (4.2) and Equation (4.6), we have:

$$q(v_0 \to \cdots \to v_l, G) \overset{d}{=} q(u_0 \to \cdots \to u_l, H), \qquad \forall G \simeq H.$$

Then, since the reader neural network f_θ is a deterministic map, we have:

$$f_\theta(q(v_0 \to \cdots \to v_l, G)) \overset{d}{=} f_\theta(q(u_0 \to \cdots \to u_l, H)), \qquad \forall G \simeq H,$$

which leads to:

$$X_\theta(G) \stackrel{d}{=} X_\theta(H), \qquad \forall G \simeq H.$$

This shows Equation (4.1) and completes the proof. □

Proof of Proposition 13

We first show a useful lemma.

Lemma 15. *The random walk in Equation* (4.3) *is invariant in probability, if the probability distributions of the starting vertex* v_0 *and each transition* $v_{t-1} \to v_t$ *are invariant.*

Proof. We prove by induction. Let us write probabilistic invariance of the starting vertex v_0 as:

$$\pi(v_0) \stackrel{d}{=} u_0, \qquad \forall G \stackrel{\pi}{\simeq} H, \tag{A.62}$$

and probabilistic invariance of each transition $v_{t-1} \to v_t$ as follows, by fixing v_{t-1} and u_{t-1}:

$$\pi(v_{t-1}) \to \pi(v_t) \stackrel{d}{=} u_{t-1} \to u_t, \qquad \forall G \stackrel{\pi}{\simeq} H, v_{t-1} \in V(G), u_{t-1} \coloneqq \pi(v_{t-1}). \tag{A.63}$$

Let us assume the following for some $t \geq 1$:

$$\pi(v_0) \to \cdots \to \pi(v_{t-1}) \stackrel{d}{=} u_0 \to \cdots \to u_{t-1}, \qquad G \stackrel{\pi}{\simeq} H.$$

Then, from the chain rule of joint distributions, the first-order Markov property of random walk in Equation (4.3), and probabilistic invariance in Equation (A.63), we obtain the following:

$$\pi(v_0) \to \cdots \to \pi(v_t) \stackrel{d}{=} u_0 \to \cdots \to u_t, \qquad G \stackrel{\pi}{\simeq} H.$$

By induction from the initial condition $\pi(v_0) \stackrel{d}{=} u_0$ in Equation (A.62), the following holds $\forall l > 0$:

$$\pi(v_0) \to \cdots \to \pi(v_l) \stackrel{d}{=} u_0 \to \cdots \to u_l, \qquad \forall G \stackrel{\pi}{\simeq} H.$$

This shows Equation (4.2) and completes the proof. □

We now prove Proposition 13.

Proof. We recall Equation (4.3) for a given conductance function $c_G : E(G) \to \mathbb{R}_+$:

$$\text{Prob}[v_t = x | v_{t-1} = u] \coloneqq \frac{c_G(u,x)}{\sum_{y \in N(u)} c_G(u,y)}.$$

From Lemma 15, it is sufficient to show the probabilistic invariance of each transition $v_{t-1} \to v_t$ as we sample v_0 from invariant distribution Uniform($V(G)$) (Algorithm 2). We rewrite Equation (A.63) as:

$$\text{Prob}[v_t = x | v_{t-1} = u] = \text{Prob}[u_t = \pi(x) | u_{t-1} = \pi(u)], \qquad \forall G \stackrel{\pi}{\simeq} H. \tag{A.64}$$

If the conductance function $c_{[\cdot]}(\cdot)$ is invariant (Equation (4.4)), we can show Equation (A.64) by:

$$\begin{aligned}
\mathrm{Prob}[v_t = x | v_{t-1} = u] &\coloneqq \frac{c_G(u,x)}{\sum_{y \in N(u)} c_G(u,y)}, \\
&= \frac{c_H(\pi(u),\pi(x))}{\sum_{y \in N(u)} c_H(\pi(u),\pi(y))}, \\
&= \frac{c_H(\pi(u),\pi(x))}{\sum_{\pi(y) \in N(\pi(u))} c_H(\pi(u),\pi(y))}, \\
&= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(u)], \qquad \forall G \overset{\pi}{\simeq} H.
\end{aligned}$$

In the third equality, we used the fact that isomorphism $\pi(\cdot)$ preserves adjacency. It is clear that any conductance function $c_{[\cdot]}(\cdot)$ with a constant conductance such as $c_G(\cdot) = 1$ is invariant. Furthermore, any conductance function that only uses degrees of endpoints $\deg(u)$, $\deg(v)$ is invariant as isomorphism $\pi(\cdot)$ preserves the degree of each vertex. □

Proof of Proposition 28

We extend the proof of Proposition 13 to second-order random walks discussed in Appendix A.7.1. We first show a useful lemma:

Lemma 16. *A second-order random walk algorithm is invariant in probability, if the probability distributions of the starting vertex v_0, the first transition $v_0 \to v_1$, and each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ are invariant.*

Proof. We prove by induction. The probabilistic invariance of starting vertex v_0 and first transition $v_0 \to v_1$ are:

$$\pi(v_0) \overset{d}{=} u_0, \qquad \forall G \overset{\pi}{\simeq} H, \tag{A.65}$$

$$\pi(v_0) \to \pi(v_1) \overset{d}{=} u_0 \to u_1, \qquad \forall G \overset{\pi}{\simeq} H, v_0 \in V(G), u_0 \coloneqq \pi(v_0), \tag{A.66}$$

and probabilistic invariance of each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ can be written as follows by fixing (v_{t-2}, v_{t-1}) and (u_{t-2}, u_{t-1}):

$$\begin{aligned}
\pi(v_{t-2}) \to \pi(v_{t-1}) \to \pi(v_t) \overset{d}{=} u_{t-2} \to u_{t-1} \to u_t, \qquad & \forall G \overset{\pi}{\simeq} H, \\
& (v_{t-2}, v_{t-1}) \in E(G), \\
& (u_{t-2}, u_{t-1}) \coloneqq (\pi(v_{t-2}), \pi(v_{t-1})).
\end{aligned} \tag{A.67}$$

Let us assume the following for some $t \geq 2$:

$$\pi(v_0) \to \cdots \to \pi(v_{t-2}) \to \pi(v_{t-1}) \overset{d}{=} u_0 \to \cdots \to u_{t-2} \to u_{t-1}, \qquad G \overset{\pi}{\simeq} H.$$

Then, from the chain rule of joint distributions, the second-order Markov property of second-order random walks, and probabilistic invariance in Equation (A.67), we obtain the following:

$$\pi(v_0) \to \cdots \to \pi(v_{t-1}) \to \pi(v_t) \overset{d}{=} u_0 \to \cdots \to u_{t-1} \to u_t, \qquad G \overset{\pi}{\simeq} H.$$

By induction from the initial condition $\pi(v_0) \to \pi(v_1) \overset{d}{=} u_0 \to u_1$ given from Equations (A.65) and

(A.66), the following holds $\forall l > 0$:

$$\pi(v_0) \to \cdots \to \pi(v_l) \overset{d}{=} u_0 \to \cdots \to u_l, \qquad \forall G \overset{\pi}{\simeq} H.$$

This shows Equation (4.2) and completes the proof. □

We now prove Proposition 28.

Proof. We assume that the first transition $v_0 \to v_1$ is given by the underlying first-order random walk algorithm. This is true for non-backtracking since there is no v_{t-2} to avoid, and true for the official implementation of node2vec [Grover and Leskovec, 2016]. Then from Lemma 16, it is sufficient to show probabilistic invariance of each transition $(v_{t-2}, v_{t-1}) \to v_t$ for $t > 1$ as we sample v_0 from the invariant distribution Uniform($V(G)$) (Algorithm 2) and the first transition $v_0 \to v_1$ is given by the first-order random walk which is assumed to be invariant in probability. Rewrite Equation (A.67) as:

$$\text{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] = \text{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \qquad \forall G \overset{\pi}{\simeq} H. \tag{A.68}$$

For non-backtracking, we first handle the case where $i \to j$ reaches a dangling vertex j which has i as its only neighbor. In this case the walk begrudgingly backtracks $i \to j \to i$ (Appendix A.7.1). Since isomorphism $\pi(\cdot)$ preserves adjacency, $\pi(i) \to \pi(j)$ also reaches a dangling vertex $\pi(j)$ and must begrudgingly backtrack $\pi(i) \to \pi(j) \to \pi(i)$. By interpreting the distributions on v_t and u_t as one-hot at i and $\pi(i)$, respectively, we can see that Equation (A.68) holds. If $i \to j$ does not reach a dangling vertex, the walk $j \to x$ follows the invariant probability of the underlying first-order random walk $\text{Prob}[v_{t+1} = x | v_t = j]$ renormalized over $N(j) \setminus \{i\}$. Then we can show Equation (A.68) by:

$$\begin{aligned}
&\text{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] \\
&\qquad := \begin{cases} \dfrac{\text{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j) \setminus \{i\}} \text{Prob}[v_t = y | v_{t-1} = j]} & \text{for } x \neq i, \\ 0 & \text{for } x = i, \end{cases} \\
&\qquad = \begin{cases} \dfrac{\text{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j)) \setminus \{\pi(i)\}} \text{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]} & \text{for } \pi(x) \neq \pi(i), \\ 0 & \text{for } \pi(x) = \pi(i), \end{cases} \\
&\qquad = \text{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \qquad \forall G \overset{\pi}{\simeq} H.
\end{aligned}$$

In the second equality, we have used the fact that isomorphism $\pi(\cdot)$ is a bijection and preserves adjacency, and the assumption that the first-order random walk algorithm is invariant in probability. For node2vec walk, we first show the invariance of the weighting term $\alpha(i, x)$ in Equation (A.57) using the fact that isomorphism preserves shortest path distances between vertices $d(i, x) = d(\pi(i), \pi(x))$:

$$\begin{aligned}
\alpha(i, x) &:= \begin{cases} 1/p & \text{for } d(i, x) = 0, \\ 1 & \text{for } d(i, x) = 1, \\ 1/q & \text{for } d(i, x) = 2. \end{cases} = \begin{cases} 1/p & \text{for } d(\pi(i), \pi(x)) = 0, \\ 1 & \text{for } d(\pi(i), \pi(x)) = 1, \\ 1/q & \text{for } d(\pi(i), \pi(x)) = 2. \end{cases} \\
&= \alpha(\pi(i), \pi(x)).
\end{aligned}$$

Then we can show Equation (A.68) by:

$$\begin{aligned}\mathrm{Prob}[v_t = x | v_{t-1} = j, v_{t-2} = i] &\coloneqq \frac{\alpha(i,x)\,\mathrm{Prob}[v_t = x | v_{t-1} = j]}{\sum_{y \in N(j)} \alpha(i,y)\,\mathrm{Prob}[v_t = y | v_{t-1} = j]}, \\ &= \frac{\alpha(\pi(i),\pi(x))\,\mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j)]}{\sum_{\pi(y) \in N(\pi(j))} \alpha(\pi(i),\pi(y))\,\mathrm{Prob}[u_t = \pi(y) | u_{t-1} = \pi(j)]}, \\ &= \mathrm{Prob}[u_t = \pi(x) | u_{t-1} = \pi(j), u_{t-2} = \pi(i)], \qquad \forall G \overset{\pi}{\simeq} H.\end{aligned}$$

In the second equality, we have used the fact that isomorphism $\pi(\cdot)$ preserves adjacency, and the assumption that the first-order walk algorithm is invariant in probability. □

Proof of Proposition 14

Proof. Given a walk $v_0 \to \cdots \to v_l$ on G, we can write the anonymized namespace $\mathrm{id}(\cdot)$ as follows:

$$\mathrm{id}(v_t) = 1 + \arg\min_i [v_i = v_t].$$

Consider any walk $\pi(v_0) \to \cdots \to \pi(v_l)$ on some H which is isomorphic to G via π. Let us denote the anonymization of this walk as $\mathrm{id}'(\cdot)$. Then we have:

$$\begin{aligned}\mathrm{id}'(\pi(v_t)) &= 1 + \arg\min_i [\pi(v_i) = \pi(v_t)], \\ &= 1 + \arg\min_i [v_i = v_t], \\ &= \mathrm{id}(v_t), \qquad \forall G \overset{\pi}{\simeq} H. \end{aligned} \tag{A.69}$$

The second equality is due to the fact that π is a bijection. This completes the proof for anonymization. For the named neighbors, we prove by induction. Let us fix some $t \geq 1$ and assume:

$$q(v_0 \to \cdots \to v_{t-1}, G) = q(\pi(v_0) \to \cdots \to \pi(v_{t-1}), H), \qquad \forall G \overset{\pi}{\simeq} H. \tag{A.70}$$

Let $T \subseteq E(G)$ be the set of edges recorded by $\mathbf{z} \coloneqq q(v_0 \to \cdots \to v_{t-1}, G)$, and $T' \subseteq E(H)$ be the set of edges recorded by $\mathbf{z}' \coloneqq q(\pi(v_0) \to \cdots \to \pi(v_{t-1}), H)$. Then, T and T' are isomorphic via π:

$$T' = \{(\pi(u), \pi(v)) \forall (u,v) \in T\}.$$

The equality is shown as follows. ($\supseteq$) Any $(u,v) \in T$ is recorded in $\mathbf{z}$ as $(\mathrm{id}(u), \mathrm{id}(v))$. From Equations (A.70) and (A.69), we have $\mathbf{z} = \mathbf{z}'$ and $(\mathrm{id}(u), \mathrm{id}(v)) = (\mathrm{id}(\pi(u)), \mathrm{id}(\pi(v)))$. Thus, $(\pi(u), \pi(v))$ is recorded in $\mathbf{z}'$ as $(\mathrm{id}(\pi(u)), \mathrm{id}(\pi(v)))$, i.e., $(\pi(u), \pi(v)) \in T'$. ($\subseteq$) This follows from symmetry. Now, we choose some $v_t \in N(v_{t-1})$ and consider the set $F \subseteq E(G)$ of all unrecorded edges from v_t:

$$F \coloneqq \{(v_t, u) : u \in N(v_t)\} \setminus T.$$

Then, the set D of unrecorded neighbors of v_t is given as:

$$D \coloneqq \{u : (v_t, u) \in F\}.$$

Since $\pi(\cdot)$ preserves adjacency, the set $F' \subseteq E(H)$ of all unrecorded edges from $\pi(v_t)$ is given by:

$$\begin{aligned}
F' &\coloneqq \{(\pi(v_t), \pi(u)) : \pi(u) \in N(\pi(v_t))\} \setminus T', \\
&= \{(\pi(v_t), \pi(u)) : u \in N(v_t)\} \setminus \{(\pi(u'), \pi(v'))\forall(u', v') \in T\}, \\
&= \{(\pi(v_t), \pi(u)) : u \in N(v_t), (v_t, u) \notin T\}, \\
&= \{(\pi(v_t), \pi(u)) : (v_t, u) \in F\},
\end{aligned}$$

and consequently:

$$D' \coloneqq \{\pi(u) : (\pi(v_t), \pi(u)) \in F'\} = \{\pi(u) : u \in D\}.$$

While D and D' may contain vertices not yet visited by the walks and hence not named by $\mathrm{id}(\cdot)$, what we record are named neighbors. Let S be the set of all named vertices in G. It is clear that the set S' of named vertices in H is given by $S' = \{\pi(v) : v \in S\}$. Then, the named neighbors in G to be newly recorded for $v_{t-1} \to v_t$ is given by $U = D \cap S$, and for $\pi(v_{t-1}) \to \pi(v_t)$ in H the set is given by $U' = D' \cap S'$. Since $\pi(\cdot)$ is a bijection, we can see that $U' = \{\pi(u) : u \in U\}$. From the invariance of anonymization in Equation (A.69), U and U' are named identically $\{\mathrm{id}(u) : u \in U\} = \{\mathrm{id}(\pi(u)) : \pi(u) \in U'\}$, and therefore will be recorded identically. As a result, the information to be added to the records at time t are identical, and we have:

$$q(v_0 \to \cdots \to v_t, G) = q(\pi(v_0) \to \cdots \to \pi(v_t), H), \qquad \forall G \overset{\pi}{\simeq} H.$$

Then, by induction from the initial condition:

$$q(v_0, G) = q(\pi(v_0), H) = \mathrm{id}(v_0) = \mathrm{id}(\pi(v_0)) = 1,$$

the recording function using anonymization and named neighbors is invariant. □

Proof of Theorem 14

Proof. We revisit the known fact that hitting times on the infinite line $\mathbb{Z}$ is infinite [Dumitriu et al., 2003, Janson and Peres, 2012, carnage, 2012, McNew, 2013]. Consider a local ball $B_1(0)$ of radius 1 from the origin 0 such that $V(B_1(0)) = \{-1, 0, 1\}$. Let us assume that the expected number of steps for a random walk starting at 0 to visit 1 for the first time is finite, and denote it by T. In the first step of the walk, we have to go either to -1 or 1. If we go to -1, we have to get back to 0 and then to 1 which would take $2T$ in expectation (by translation symmetry). This leads to $T = 1/2 \cdot 1 + 1/2 \cdot (1 + 2T) = 1 + T$, which is a contradiction. Since visiting all vertices or traversing all edges in in $B_1(0)$ clearly involves visiting 1, the vertex and edge cover times cannot be finitely bounded. □

Proof of Theorem 15

Let us recall that our graph G is locally finite (Section 4.1.1), and denote its maximum degree as Δ. We further denote by $d(u, v)$ the shortest path distance between u and v. We show some useful lemmas. We first show that $H(u, x)$, the expected number of steps of a random walk starting at u takes until reaching x, is finite for any u, x in $B_r(v)$ for any nonzero restart probability α or restart period $k \geq r$.

Lemma 17. *For any u and x in $B_r(v)$, $H(u,x)$ is bounded by the following:*

$$H(u,x) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^r + \frac{1}{\alpha}\left(\frac{1}{\alpha}-1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r}, \tag{A.71}$$

if the random walk restarts at v with any probability $\alpha \in (0,1)$, and:

$$H(u,x) \leq k + k\Delta^r, \tag{A.72}$$

if the random walk restarts at v with any period $k \geq r$.

Proof. We first prove for random restarts. The proof is inspired by Theorem 1.1 of McNew [2013] and Lemma 2 of Zuckerman [1991]. We clarify some measure-theoretic details. Let W be the set of all (infinite) random walks on G. We denote by P the probability measure defined on the space W, equipped with its cylinder σ-algebra, by the random walk restarting at v with probability $\alpha \in (0,1)$. For a vertex x in G, we define the hitting time function $|\cdot|_x : W \to \mathbb{N} \cup \{\infty\}$ as follows:

$$|w|_x \coloneqq \arg\min_i [(w)_i = x], \qquad \forall w \in W. \tag{A.73}$$

Let $W(a)$ be the set of all random walks that start at a, $W(a,b)$ be the set of random walks that start at a and visit b, and $W(a,b^-)$ be the set of random walks that start at a and do not visit b. Then, the measurability of $|\cdot|_x$ follows from the measurability of the sets $W(a)$, $W(a,b)$, and $W(a,b^-)$, which we show as follows. $\forall a,b \in V(G)$, $W(a)$ is clearly measurable, $W(a,b)$ is measurable as it equals $\bigcup_{n\in\mathbb{N}}\{w \in W | (w)_1 = a, (w)_n = b\}$, and $W(a,b^-)$ is measurable as it is $W(a) \setminus W(a,b)$.

Consider $W(u,x^-)$, the set of random walks that start at u and never visit x. We start by showing that this set is of measure zero:

$$P(W(u,x^-)) = 0. \tag{A.74}$$

For this, we use the fact that $W(u,x^-) = W(u,v^-,x^-) \cup W(u,v,x^-)$, where $W(u,v^-,x^-)$ is the set of random walks starting at u that do not visit v nor x, and $W(u,v,x^-)$ is the set of walks starting at u that visit v and do not visit x. We have the following:

$$\begin{aligned} P(W(u,x^-)) &= P(W(u,v^-,x^-)) + P(W(u,v,x^-)), \\ &\leq P(W(u,v^-)) + P(W(u,v,x^-)). \end{aligned} \tag{A.75}$$

We show Equation (A.74) by showing $P(W(u,v^-)) = 0$ and $P(W(u,v,x^-)) = 0$ in Equation (A.75).

1. ($P(W(u,v^-)) = 0$) Consider $W(u,v^-)$, the set of all random walks that start at u and never visit v. Since restart sends a walk to v, every walk in this set never restarts. The probability of a walk not restarting until step t is $(1-\alpha)^t$. Denote this probability by p_t, and let p be the probability of a random walk to never restart. Then $p_t \downarrow p = 0$ and $P(W(u,v^-)) \leq p = 0$.

2. ($P(W(u,v,x^-)) = 0$) Assume $P(W(u,v,x^-)) > 0$. Then $P(W(v,x^-)) > 0$ since each walk step is independent. Let $W_N(v,x^-)$ be the set of walks that start at v and do not reach x within N restarts. Then we have $W_N(v,x^-) \downarrow W(v,x^-)$. If a walk restarts at v, the probability of reaching x before the next restart is at least the probability of exactly walking the shortest path from v

to x, which is $\geq (\frac{1-\alpha}{\Delta})^{d(v,x)} \in (0,1)$. Then $P(W_N(v,x^-)) \leq (1-(\frac{1-\alpha}{\Delta})^{d(v,x)})^N \downarrow 0$, leading to $P(W(v,x^-)) = 0$. This is a contradiction, so we have $P(W(u,v,x^-)) = 0$.

We are now ready to bound $H(u,x)$. Using the hitting time function $|\cdot|_x$ in Equation (A.73):

$$H(u,x) = \mathbb{E}\left[|w|_x | w \in W(u)\right].$$

Since $W(u) = W(u,x) \cup W(u,x^-)$, and $P(W(u,x^-)) = 0$ from Equation (A.74), we have:

$$\begin{aligned} H(u,x) &= \mathbb{E}[|w|_x | w \in W(u,x)] \\ &= \int_{W(u,x)} |w|_x \, dP(w | w \in W(u,x)). \end{aligned}$$

Each walk in $W(u,x)$ starts at u and, when it reaches x, it either has or has not visited v. We treat the two cases separately. Let $W(a,b,c)$ be the set of walks that start at a and reach c after b, and let $W(a,b^-,c)$ be the set of walks that start at a and reach c before b. Then we have:

$$\begin{aligned} H(u,x) &= \int_{W(u,v,x) \cup W(u,v^-,x)} |w|_x \, dP(w|W(u,x)) \\ &= \int_{W(u,v,x)} |w|_x \, dP(w|W(u,x)) + \int_{W(u,v^-,x)} |w|_x \, dP(w|W(u,x)). \end{aligned}$$

Let $\hat{H}(a,b,c)$ (or $\hat{H}(a,b^-,c)$) be the expected number of steps for a walk starting at a to reach c after reaching b (or before b), given that it is in $W(a,b,c)$ (or in $W(a,b^-,c)$). If a walk from u reaches v before x, the expected steps is given by $\hat{H}(u,x^-,v) + H(v,x)$. If the walk reaches x before v, the expected steps is $\hat{H}(u,v^-,x)$. Then, if $W(u,v^-,x) \neq \emptyset$, we can write $H(u,x)$ as follows:

$$\begin{aligned} H(u,x) = &\left[\hat{H}(u,x^-,v) + H(v,x)\right] P(W(u,v,x)|W(u,x)) \\ &+ \hat{H}(u,v^-,x)\, P(W(u,v^-,x)|W(u,x)). \end{aligned} \tag{A.76}$$

On the other hand, if $W(u,v^-,x) = \emptyset$, we simply have:

$$H(u,x) = \hat{H}(u,x^-,v) + H(v,x). \tag{A.77}$$

We first consider $\hat{H}(u,x^-,v)$, the expected number of steps from u to reach v before x. We show:

$$\hat{H}(u,x^-,v) \leq \frac{1}{\alpha}. \tag{A.78}$$

To see this, note that $\hat{H}(u,x^-,v)$ is equivalent to the expectation $\mathbb{E}[T]$ of the number of steps T to reach v from u, on the graph G with the vertex x deleted and transition probabilities renormalized. If u is isolated on this modified graph, the only walk in $W(u,x^-,v)$ is the one that immediately restarts at v, giving $\mathbb{E}[T] = 1$. Otherwise, we have $\mathbb{E}[T] \leq 1/\alpha$ due to the following. Let T' be the number of steps until the first restart at v. Since restart can be treated as a Bernoulli trial with probability of success α, T' follows geometric distribution with expectation $1/\alpha$. Since it is possible for the walk to reach v before the first restart, we have $\mathbb{E}[T] \leq \mathbb{E}[T'] = 1/\alpha$, which gives Equation (A.78).

We now consider $H(v,x)$, the expectation $\mathbb{E}[T]$ of the number of steps T to reach x from v. Let T' be the steps until the walk restarts at v, then exactly walks the shortest path from v to x for the first

time, and then restarts at v. Since T' walks until restart after walking the shortest path to x, and it is possible for a walk to reach x before walking the shortest path to it, we have $\mathbb{E}[T] \leq \mathbb{E}[T']$. Then, we split the walk of length T' into N trials, where each trial consists of restarting at v and walking until the next restart. A trial is successful if it immediately walks the shortest path from v to x. Then N is the number of trials until we succeed, and it follows geometric distribution with probability of success at least $(\frac{1-\alpha}{\Delta})^{d(v,x)}$ due to bounded degrees. Its expectation is then bounded as:

$$\mathbb{E}[N] \leq \left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)}.$$

Let S_i be the length of the i-th trial. Since each S_i is i.i.d. with finite mean $\mathbb{E}[S_i] = 1/\alpha$, and N is stopping time, we can apply Wald's identity [Hein] to compute the expectation of T':

$$\mathbb{E}[T'] = \mathbb{E}[S_1 + \cdots + S_N] = \mathbb{E}[N]\mathbb{E}[S_1] \leq \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)}.$$

We remark that $H(v,x) \leq \mathbb{E}[T']$. Combining this result with Equation (A.78), we have:

$$\hat{H}(u, x^-, v) + H(v,x) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)}. \tag{A.79}$$

If $W(u, v^-, x) = \emptyset$, we have $H(u,x) = \hat{H}(u, x^-, v) + H(v,x)$ from Equation (A.77) and it is finitely bounded for any $\alpha \in (0,1)$ by the above. If $W(u, v^-, x) \neq \emptyset$, Equations (A.76) and (A.79) lead to:

$$\begin{aligned}
H(u,x) &\leq \hat{H}(u, x^-, v) + H(v,x) + \hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u,x)), \\
&\leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)} + \hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u,x)),
\end{aligned}$$

and it suffices to bound $\hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u,x))$. We show the following:

$$\begin{aligned}
\hat{H}(u, v^-, x) &= \int_{W(u,v^-,x)} |w|_x \, dP(w|W(u, v^-, x)), \\
&= \sum_{k=1}^{\infty} \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} |w|_x \, dP(w|W(u, v^-, x)), \\
&= \sum_{k=1}^{\infty} k \int_{\{w \in W(u,v^-,x) \wedge |w|_x = k\}} dP(w|W(u, v^-, x)), \\
&= \sum_{k=1}^{\infty} k\, P(\{w \in W(u, v^-, x) \wedge |w|_x = k\}|W(u, v^-, x)), \\
&\leq \sum_{k=1}^{\infty} k\, P(\{w : |w|_x = k\}|W(u, v^-, x)), \\
&= \sum_{k=1}^{\infty} k\, \frac{P(\{w : |w|_x = k \wedge w \in W(u, v^-, x)\})}{P(W(u, v^-, x))}, \\
&\leq \sum_{k=1}^{\infty} k(1-\alpha)^k \frac{1}{P(W(u, v^-, x))}, \\
&= \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{1}{P(W(u, v^-, x))}.
\end{aligned}$$

We have used Fubini's theorem for the second equality. Then we have:

$$\begin{aligned}\hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u, x)) &\leq \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{P(W(u, v^-, x)|W(u, x))}{P(W(u, v^-, x))}, \\ &= \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\frac{1}{P(W(u, x))}. \end{aligned} \tag{A.80}$$

$P(W(u, x))$ is at least the probability of precisely walking the shortest path from u to x, which has length $d(u, x) \leq 2r$ since u and x are both in $B_r(v)$. This gives us the following:

$$\begin{aligned} P(W(u, x)) &\geq \left(\frac{1-\alpha}{\Delta}\right)^{d(u,x)}, \\ &\geq \left(\frac{1-\alpha}{\Delta}\right)^{2r}. \end{aligned}$$

Combining this with Equation (A.80), we have:

$$\hat{H}(u, v^-, x)\, P(W(u, v^-, x)|W(u, x)) \leq \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r}.$$

Combining with Equations (A.76) and (A.79), we have:

$$H(u, x) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{d(v,x)} + \frac{1}{\alpha}\left(\frac{1}{\alpha} - 1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r},$$

for any $\alpha \in (0, 1)$. Notice that, while this bound is for the case $W(u, v^-, x) \neq \emptyset$, it subsumes the bound for the case $W(u, v^-, x) = \emptyset$ (Equation (A.79)). Then, using $d(v, x) \leq r$, we get Equation (A.71). This completes the proof for random restarts.

We now prove for periodic restarts. If the walk starting at u reaches x before restarting at v, the steps taken is clearly less than k. If the walk starting at u restarts at v at step k before reaching x, it now needs to reach x from v, while restarting at v at every k steps. Let T be the steps taken to reach x from v. Let T' be the number of steps until the walk restarts at v, then exactly follows the shortest path from v to x for the first time, and then restarts at v. It is clear that $\mathbb{E}[T] \leq \mathbb{E}[T']$. Then, we split the walk of length T' into N trials, where each trial consists of restarting at v and walking k steps until the next restart. A trial is successful if it immediately walks the shortest path from v to x. Then N is the number of trials until we get a success, and it follows geometric distribution with probability of success at least $(1/\Delta)^{d(v,x)}$ only for $k \geq d(v, x)$, and zero for $k < d(v, x)$ since the walk cannot reach x before restart. Hence, its expectation is at most $\Delta^{d(v,x)}$ for $k \geq d(v, x)$, and we have $\mathbb{E}[T] \leq \mathbb{E}[T'] = k\mathbb{E}[N] \leq k\Delta^{d(v,x)}$. Adding the k steps until the first restart at v, we have:

$$H(u, x) \leq k + k\Delta^{d(v,x)},$$

for any $k \geq d(v, x)$. Using $d(v, x) \leq r$, we get Equation (A.72). This completes the proof. □

We now extend Lemma 17 to edges. Let $H(u, (x, y))$ be the expected number of steps of a random walk starting at u takes until traversing an edge (x, y) by $x \to y$. We show that $H(u, (x, y))$ is finite for any u and adjacent x, y in $B_r(v)$ for any nonzero restart probability α or restart period $k \geq r + 1$.

Lemma 18. *For any u and adjacent x, y in $B_r(v)$, $H(u,(x,y))$ is bounded by the following:*

$$H(u,(x,y)) \leq \frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{r+1} + \frac{1}{\alpha}\left(\frac{1}{\alpha}-1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r+1},$$

if the random walk restarts at v with any probability $\alpha \in (0,1)$, and:

$$H(u,(x,y)) \leq k + k\Delta^{r+1},$$

if the random walk restarts at v with any period $k \geq r+1$.

Proof. The proof is almost identical to Lemma 17, except the target x of reaching is substituted by (x,y) in the direction of $x \to y$, and all arguments that use the shortest path from u or v to x instead use the shortest path to x postfixed by $x \to y$, which adds $+1$ to several terms in the bounds. □

We are now ready to prove Theorem 15.

Proof. The proof is inspired by the spanning tree argument of Aleliunas et al. [1979]. Let us consider a depth first search of $B_r(v)$ starting from v. We denote by T the resulting spanning tree with vertices $V(T) = V(B_r(v))$. We consider the expected time for a random walk starting at v to visit every vertex in the precise order visited by the depth first search by traversing each edge twice. It is clear that this upper-bounds the vertex cover time of $B_r(v)$ starting at v (Equation (A.60)):

$$C_V(B_r(v)) \leq \sum_{(x,y)\in E(T)} [H(x,y) + H(y,x)].$$

Then, using the bounds from Lemma 17, the property of spanning trees $|E(T)| = |V(T)| - 1$, and the fact that $|V(T)| = |V(B_r(v))| \leq \Delta^r$ from bounded degree, we obtain:

$$C_V(B_r(v)) \leq 2(\Delta^r - 1)\left(\frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{r} + \frac{1}{\alpha}\left(\frac{1}{\alpha}-1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r}\right),$$

if the random walk restarts at v with any probability $\alpha \in (0,1)$, and:

$$C_V(B_r(v)) \leq 2(\Delta^r - 1)(k + k\Delta^r),$$

if the random walk restarts at v with any period $k \geq r$. This completes the proof for the vertex cover time. For the edge cover time, we consider the expected time for a random walk starting at v to visit every edge in the precise order discovered[8] by the depth first search by traversing each edge twice. It is clear that this upper-bounds the edge cover time of $B_r(v)$ starting at v (Equation (A.61)):

$$C_E(B_r(v)) \leq \sum_{(x,y)\in E(B_r(v))} [H(x,(x,y)) + H(y,(y,x))].$$

Then, using Lemma 17 and the fact that $|E(B_r(v))| \leq \Delta^{2r} - 1$ from bounded degree, we obtain:

$$C_E(B_r(v)) \leq 2(\Delta^{2r} - 1)\left(\frac{1}{\alpha} + \frac{1}{\alpha}\left(\frac{\Delta}{1-\alpha}\right)^{r+1} + \frac{1}{\alpha}\left(\frac{1}{\alpha}-1\right)\left(\frac{\Delta}{1-\alpha}\right)^{2r+1}\right),$$

[8]We remark that depth first search discovers all edges of a graph, while not necessarily visiting all of them.

if the random walk restarts at v with any probability $\alpha \in (0, 1)$, and:

$$C_E(B_r(v)) \leq 2(\Delta^{2r} - 1) \cdot (k + k\Delta^{r+1}),$$

if the random walk restarts at v with any period $k \geq r + 1$. This completes the proof. □

While our proof shows finite bounds for the cover times, it is possible that they can be made tighter, for instance based on Zuckerman [1991]. We leave improving the bounds as a future work.

Proof of Theorem 16

Recall universal approximation of graph-level functions in probability in Definition 6. We remark that an RWNN $X_\theta(\cdot)$ is composed of a random walk algorithm, a recording function $q : (v_0 \to \cdots \to v_l, G) \mapsto \mathbf{z}$, and a reader neural network $f_\theta : \mathbf{z} \mapsto \hat{\mathbf{y}} \in \mathbb{R}$.

Intuitively, if the record $\mathbf{z}$ of the random walk always provides complete information of the input graph G, we may invoke universal approximation of f_θ to always obtain $|\phi(G) - f_\theta(\mathbf{z})| < \epsilon$, and thus $|\phi(G) - X_\theta(G)| < \epsilon$. However, this is not always true as the random walk may e.g. fail to visit some vertices of G, in which case the record $\mathbf{z}$ would be incomplete. As we show below, this uncertainty leads to the probabilistic bound $> 1 - \delta$ of the approximation.

Let us denote the collection of all possible random walk records as $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$, and consider a decoding function $\psi : \{\mathbf{z}\} \to \mathbb{G}_n$ that takes the record $\mathbf{z} \coloneqq q(v_0 \to \cdots \to v_l, G)$ of a given random walk $v_{[\cdot]}$ and outputs the graph $\psi(\mathbf{z}) \in \mathbb{G}_n$ composed of all recorded vertices $V(H) \coloneqq \{\mathrm{id}(v_t) : v_t \in \{v_0, ..., v_l\}\}$ and all recorded edges $E(H) \subset V(H) \times V(H)$. We show the following lemma:

Lemma 19. *Let $G_\mathbf{z}$ be the subgraph of G whose vertices and edges are recorded by $\mathbf{z}$. Then the graph $\psi(\mathbf{z})$ decoded from the record $\mathbf{z}$ is isomorphic to $G_\mathbf{z}$ through the namespace* $\mathrm{id}(\cdot)$*:*

$$G_\mathbf{z} \overset{\mathrm{id}}{\simeq} \psi(\mathbf{z}). \tag{A.81}$$

Furthermore, the decoded graph $\psi(\mathbf{z})$ reconstructs G up to isomorphism, that is,

$$G \overset{\mathrm{id}}{\simeq} \psi(\mathbf{z}), \tag{A.82}$$

if the recording function $q(\cdot)$ and the random walk $v_{[\cdot]}$ satisfies either of the following:

- *$q(\cdot)$ uses anonymization, and $v_{[\cdot]}$ has traversed all edges of G.*
- *$q(\cdot)$ uses anonymization and named neighbors, and $v_{[\cdot]}$ has visited all vertices of G.*

Proof. Equation (A.81) is straightforward from the fact that the namespace $\mathrm{id}(\cdot)$ defines a bijection from $V(G_\mathbf{z})$ to $[|V(G_\mathbf{z})|]$, and the recording function uses names $\mathrm{id}(v_t)$ to record vertices and edges. Equation (A.82) is satisfied when all vertices and edges of G have been recorded, i.e., $G_\mathbf{z} = G$, which is possible either when the random walk has traversed all edges of G, or when it has traversed all vertices of G and named neighbors are used to record the induced subgraph $G[V(G)] = G$. □

We further remark Markov's inequality for any nonnegative random variable T and $a > 0$:

$$\mathrm{Prob}[T \geq a] \leq \frac{\mathbb{E}[T]}{a}. \tag{A.83}$$

We are now ready to prove Theorem 16.

Proof. Instead of directly approximating the target function $\phi : \mathbb{G}_n \to \mathbb{R}$, it is convenient to define a proxy target function on random walk records $\phi' : \{\mathbf{z}\} \to \mathbb{R}$ where $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$ as follows:

$$\phi' \coloneqq \phi \circ \psi,$$

where $\psi : \{\mathbf{z}\} \to \mathbb{G}_n$ is the decoding function of walk records. Then, for a given $\mathbf{z}$, we have:

$$G \simeq \psi(\mathbf{z}) \qquad \Longrightarrow \qquad \phi(G) = \phi'(\mathbf{z}),$$

which is because ϕ is an invariant function, so $\phi(G) = \phi(\psi(\mathbf{z})) = \phi \circ \psi(\mathbf{z}) = \phi'(\mathbf{z})$. Then we have:

$$\mathrm{Prob}[G \simeq \psi(\mathbf{z})] \leq \mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| = 0]. \tag{A.84}$$

We now invoke universality of f_θ to approximate ϕ'. If f_θ is a universal approximator of functions on its domain $\{\mathbf{z}\} \coloneqq \mathrm{Range}(q)$, for any $\epsilon > 0$ there exists a choice of θ such that the below holds:

$$|\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon, \qquad \forall \mathbf{z} \in \mathrm{Range}(q). \tag{A.85}$$

Combining Equations (A.84) and (A.85), we have:

$$\mathrm{Prob}[G \simeq \psi(\mathbf{z})] \leq \mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon]. \tag{A.86}$$

We remark triangle inequality of distances on $\mathbb{R}$, for a given $\mathbf{z}$:

$$|\phi(G) - f_\theta(\mathbf{z})| \leq |\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})|,$$

which implies, for a given $\mathbf{z}$:

$$|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon \qquad \Longrightarrow \qquad |\phi(G) - f_\theta(\mathbf{z})| < \epsilon,$$

and hence:

$$\mathrm{Prob}[|\phi(G) - \phi'(\mathbf{z})| + |\phi'(\mathbf{z}) - f_\theta(\mathbf{z})| < \epsilon] \leq \mathrm{Prob}[|\phi(G) - f_\theta(\mathbf{z})| < \epsilon]. \tag{A.87}$$

Combining Equations (A.86) and (A.87), we have:

$$\mathrm{Prob}[|\phi(G) - f_\theta(\mathbf{z})| < \epsilon] \geq \mathrm{Prob}[G \simeq \psi(\mathbf{z})],$$

which can be written as follows:

$$\mathrm{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] \geq \mathrm{Prob}[G \simeq \psi(\mathbf{z})]. \tag{A.88}$$

We now consider the probability of the event $G \simeq \psi(\mathbf{z})$ based on Lemma 19. We first consider the case where the recording function $q(\cdot)$ uses anonymization. In this case, $G \simeq \psi(\mathbf{z})$ is achieved if the random walk of length l has traversed all edges of G. Let $T_E(G, v_0)$ be the number of steps that a random walk starting at v_0 takes until traversing all edges of G. Since the edge cover time $C_E(G)$ is its

expectation taken at the worst possible starting vertex (Equation (A.59)), we have the following:

$$\mathbb{E}[T_E(G, v_0)] \leq C_E(G), \qquad \forall v_0 \in V(G),$$

which leads to the following from Markov's inequality (Equation (A.83)):

$$\text{Prob}[T_E(G, v_0) < l] \geq 1 - \frac{\mathbb{E}[T_E(G)]}{l} \geq 1 - \frac{C_E(G)}{l}. \tag{A.89}$$

For a given random walk $v_0 \to \cdots \to v_l$ and its record $\mathbf{z}$, the following holds:

$$T_E(G, v_0) < l \qquad \Longrightarrow \qquad G \simeq \psi(\mathbf{z}),$$

which implies the following:

$$\text{Prob}[T_E(G, v_0) < l] \leq \text{Prob}[G \simeq \psi(\mathbf{z})]. \tag{A.90}$$

Combining Equations (A.88), (A.89), and (A.90), we have:

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] \geq 1 - \frac{C_E(G)}{l}.$$

Therefore, for any $\delta > 0$, if we choose $l > C_E(G)/\delta$ we would have the following:

$$\text{Prob}[|\phi(G) - X_\theta(G)| < \epsilon] > 1 - \delta.$$

This completes the proof for anonymization. The proof is identical for the recording function that uses anonymization and named neighbors, except that the edge cover time is changed to the vertex cover time $C_V(G)$ (Equation (A.58)). This is because named neighbor recording automatically records the induced subgraph of visited vertices, thus visiting all vertices implies recording all edges, $G[V(G)] = G$. □

Proof of Theorem 17

Proof. The proof is almost identical to Theorem 16, except $G \in \mathbb{G}_n$ are substituted by $B_r(v) \in \mathbb{B}_r$, and the decoding function $\psi : \{\mathbf{z}\} \to \mathbb{B}_r$ is defined to ignore all recorded vertices $\text{id}(x)$ whose shortest path distance from the starting vertex $\text{id}(v) = \text{id}(v_0) = 1$ exceeds r. The latter is necessary to restrict the range of the decoding function ψ to $\mathbb{B}_r$. In addition, any nonzero restart probability α or restart period $k \geq r + 1$ is sufficient to make the cover times $C_E(B_r(v))$ and $C_V(B_r(v))$ finite (Theorem 15), thereby guaranteeing the existence of a finite choice of l. □

Proof of Theorem 18

Proof. Since G is connected and non-bipartite, the uniform random walk on it defines an ergodic Markov chain with a unique stationary distribution $\boldsymbol{\pi}$. The limiting frequency of visits on each vertex v is precisely the stationary probability $\boldsymbol{\pi}_v$. Since the model reads $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by average pooling, the output is given by weighted mean $\sum_v \boldsymbol{\pi}_v \mathbf{x}_v$ which is $\mathbf{x}^\top \boldsymbol{\pi}$. □

Proof of Theorem 19

Proof. Since the model reads $\mathbf{x}_{v_0} \to \cdots \to \mathbf{x}_{v_l}$ by average pooling, the feature Jacobian $|\partial \mathbf{h}_u^{(l)} / \partial \mathbf{x}_v|$ is given as number of visits to the vertex v in the random walk $v_0 \to \cdots \to v_l$ starting at $v_0 = u$, divided by length $l+1$. Let us denote the expected number of these visits by $J(u, v, l)$. Let $\mathbb{1}_{v_t=v}$ be the indicator function that equals 1 if $v_t = v$ and 0 otherwise. Then we can write $J(u, v, l)$ as:

$$J(u, v, l) = \mathbb{E}\left[\sum_{t=0}^{l} \mathbb{1}_{v_t=v} | v_0 = u\right] = \sum_{t=0}^{l} \mathbb{E}[\mathbb{1}_{v_t=v} | v_0 = u].$$

We have used linearity of expectations for the second equality. $\mathbb{E}[\mathbb{1}_{v_t=v} | v_0 = u]$ is the probability of being at v at step t given that the walk started at u. This probability is precisely $[P^t]_{uv}$. Therefore:

$$J(u, v, l) = \sum_{t=0}^{l} \left[P^t\right]_{uv} = \left[\sum_{t=0}^{l} P^t\right]_{uv},$$

which gives the equality in Equation (4.7). Furthermore, since G is connected and non-bipartite, the uniform random walk on it defines an ergodic Markov chain with a unique stationary distribution $\boldsymbol{\pi}$. The limiting frequency of visits on vertex v is precisely the stationary probability $\boldsymbol{\pi}_v$, which gives the convergence in Equation (4.7). □

A.8 Appendix of Flock

A.8.1 Proofs

Proof of expressivity (Proposition 15)

The main proposition of this section formalizes the fact that Flock can approximate any link-invariant function over fixed-size knowledge graphs in probability. Intuitively, when the length of the sampled walks ℓ becomes higher, the probability of a single walk witnessing all the edges grows to 1. Once a walk visits all the edges, a sufficiently powerful sequence processor can derive the whole graph structure from its anonymized representation, recreating the graph in its entirety, up to isomorphism. Then, the processor can return the value of the approximated function for that graph.

We start by showing that the edge cover time $C_E(\cdot)$ of graphs in $K_{n,m}$ is bounded:

Lemma 20. *Let $G \in \mathbb{K}_{n,m}$ for some n, m. The edge cover time $C_E(G)$ of G, using the algorithm from Appendix A.8.2, is finite.*

Proof. Let $G = (V, E, R) \in \mathbb{K}_{n,m}$ be a graph. For any edge $e \in E$ and any vertex $v \in V$, let $H_v(e)$ denote the expected number of steps of the random walk algorithm η described in Appendix A.8.2. Then, the edge cover time $C_E(G)$ of G with η, i.e. the expected number of steps that η needs to take before visiting every edge in G, is bounded above by:

$$C_E(G) \leq \sum_{e \in E} \max_{v \in V} H_v(e) \leq m \cdot \max_{\substack{e \in E \\ v \in V}} H_v(e)$$

Indeed, consider the event of visiting all these edges in order $e_1, \ldots, e_m$:

$$\begin{aligned}
C_E(G) &= \mathbb{E}[\#\text{steps to visit all } e_1, \ldots, e_m] \\
&\leq \mathbb{E}[\#\text{steps to visit } e_1, \text{ then } e_2, \ldots, \text{ then } e_m] \\
&\leq \mathbb{E}[\#\text{steps to visit } e_1] + \sum_{i=1}^{m-1} \mathbb{E}[\#\text{steps to visit } e_{i+1} \text{ starting from } h_i \text{ or } t_i] \\
&\leq \max_{v \in V} H_v(e_1) + \sum_{i=1}^{m-1} \max(H_{h_i}(e_{i+1}), H_{t_i}(e_{i+1})) \\
&\leq \max_{v \in V} H_v(e_1) + \sum_{i=1}^{m-1} \max_{v \in V} H_v(e_{i+1}) \\
&= \sum_{i=1}^{m} \max_{v \in V} H_v(e_i)
\end{aligned}$$

where h_i and t_i are the head and tail of the edge e_i, respectively. Therefore, to show that $C_E(G)$ is finite, it suffices to prove that $H_v(e)$ is bounded for all $v \in V, e \in E$.

Fix $v \in V$ and $e \in E$. Consider an infinite random walk generated with η over G, starting at v:

$$v = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} v_2 \xrightarrow{r_3} \ldots$$

We want to bound the expected first index t, such that e is the edge traversed in step $v_{t-1} \xrightarrow{r_t} v_t$. Denote by Δ a maximum degree of a vertex in G (counted as the number of connected vertices $\mathcal{N}(v)$), by ρ the maximum number of edges between any single pair of nodes and by d – the diameter of the graph, i.e.

the length of the longest shortest path between two vertices (in the undirected version of G). Consider the series of events $A_0, A_1, \dots$ where A_i is characterized as:

$$A_i := \text{the event that starting from } v_{i(d+2)} \text{ the walk will follow a shortest path to one of the endpoints of } e \text{ and then go through } e$$

Let $e = (h_e, r_e, t_e)$. For all values of i, by definition, the length of the shortest path from $v_{i(d+2)}$ to h_e or t_e is at most d. Therefore, the whole part of the walk described in A_i is at most $d+1$ steps long. By the definition of the used random walk algorithm, which only looks at the previously taken edge, we can deduce that the events A_i are all mutually independent.

Moreover, let $v_{i(d+2)} = u_0 \xrightarrow{s_1} u_1 \xrightarrow{s_2} \dots \xrightarrow{s_\ell} u_\ell \in \{h_e, t_e\}$ be a shortest path from $v_{i(d+2)}$ to one of h_e, t_e. Note that by minimality, there cannot be any backtracking while following this path. Therefore, the probability of the next visited node is dependent only on the value of the previous one, and we can bound the probability $P(A_i)$ of A_i from below by:

$$\mathbb{P}(A_i) \geq \mathbb{P}(\text{pass through } e \text{ after reaching } h_e \text{ or } t_e) \cdot \prod_{j=0}^{\ell-1} \mathbb{P}(v_{i(d+2)+j+1} = u_{j+1} \mid v_{i(d+2)+j} = u_j)$$

The first term on the right hand side is the probability of selecting e while being at h_e or t_e, which is the probability of first selecting the other endpoint (out of at most Δ neighbors) and then picking e over other edges between h_e and t_e (of which there is at most ρ). Hence:

$$\mathbb{P}(\text{pass through } e \text{ after reaching } h_e \text{ or } t_e) \geq \frac{1}{\Delta} \cdot \frac{1}{\rho} = \frac{1}{\Delta \cdot \rho}$$

As we never reach a backtracking situation by minimality of the shortest path, we can also write:

$$\mathbb{P}(v_{i(d+2)+j+1} = u_{j+1} \mid v_{i(d+2)+j} = u_j) = \frac{1}{|\mathcal{N}(v_{i(d+2)+j})|} \geq \frac{1}{\Delta}$$

Combining these observations, we can derive a bound for $\mathbb{P}(A_i)$ in terms of Δ, ρ and d:

$$\begin{aligned}\mathbb{P}(A_i) &\geq \mathbb{P}(\text{pass through } e \text{ after reaching } h_e \text{ or } t_e) \cdot \prod_{j=0}^{\ell-1} \mathbb{P}(v_{i(d+2)+j+1} = u_{j+1} \mid v_{i(d+2)+j} = u_j) \\ &\geq \frac{1}{\Delta \cdot \rho} \cdot \prod_{j=0}^{\ell-1} \frac{1}{\Delta} \quad \geq \quad \frac{1}{\Delta \cdot \rho} \left(\frac{1}{\Delta}\right)^{\ell} \quad \geq \quad \frac{1}{\Delta \cdot \rho} \left(\frac{1}{\Delta}\right)^{d} \quad = \quad \frac{1}{\rho \Delta^{d+1}}\end{aligned}$$

Finally, note that if A_i is true, then the first index t such that $v_{t-1} \xrightarrow{r_t} v_t$ traverses e is at most $(i+1)(d+2)$. We can therefore bound the expectation of such t, being $H_v(e) = H_{v_0}(e)$ by:

$$\begin{aligned}H_v(e) &\leq (d+2) \cdot \mathbb{P}(A_0) + 2(d+2) \cdot \mathbb{P}(\neg A_0 \wedge A_1) + 3(d+2) \cdot \mathbb{P}(\neg A_0 \wedge \neg A_1 \wedge A_2) + \dots \\ &= (d+2) \cdot \mathbb{P}(A_0) + 2(d+2) \cdot \mathbb{P}(\neg A_0) \cdot P(A_1) + 3(d+2) \cdot \mathbb{P}(\neg A_0) \cdot \mathbb{P}(\neg A_1) \cdot \mathbb{P}(\wedge A_2) + \dots \\ &= (d+2) + \mathbb{P}(\neg A_0) \cdot (d+2 + \mathbb{P}(\neg A_1) \cdot (d+2 + \mathbb{P}(\neg A_2) \cdot (\dots))) \\ &\leq (d+2) + \left(1 - \frac{1}{\rho \Delta^{d+1}}\right) \cdot \left(d+2 + \left(1 - \frac{1}{\rho \Delta^{d+1}}\right) \cdot \left(d+2 + \left(1 - \frac{1}{\rho \Delta^{d+1}}\right) \cdot (\dots)\right)\right) \\ &= \rho(d+2)\Delta^{d+1}\end{aligned}$$

Since $\rho \leq m, d+2 \leq n$ and $\Delta \leq n$, we have $H_v(e) \leq m(n+2)n^n$, which completes the proof. □

Remark 1. *The bound obtained in the proof of Lemma 20 is very crude. In fact, we could transform the given knowledge graph into a simple graph (undirected, with no multi-edges) by substituting each edge* $u \xrightarrow{r} v$ *with two undirected edges* $u \leftrightarrow v_{(u,r,v)} \leftrightarrow v$*. The augmented graph will then have* $n + m$ *vertices, and our random walk algorithm naturally translates to a weighted random walk on the transformed graph. This hints at an assumption that in practice, the edge cover time of the used random walk algorithm is of the magnitude* $O((n+m)^3) = O(n^3 + m^3)$*.*

Let us now prove a fact about the number of distinct, up to isomorphism, graphs in $\mathbb{K}_{n,m}$.

Lemma 21. *For any* n, m*, the number of isomorphism classes in* $\mathbb{K}_{n,m}$ *is finite.*

Proof. Since the number of distinct relation types that a graph in $\mathbb{K}_{n,m}$ is at most m, it suffices to show that the number of isomorphism classes of graphs in $\mathbb{K}_{n,m}$ with exactly k relation types is bounded, for all $k \in \{1, 2, \ldots, m\}$.

Fix the number $k \in \{1, 2, \ldots, m\}$ and consider $G = (V, E, R) \in \mathbb{K}_{n,m}$ with $|R| = k$. We will show that, up to isomorphism, there are finitely many such choices of G. Firstly, as renaming does not change the graph structure, without loss of generality, we can assume that:

$$V = \{v_1, v_2, \ldots, v_n\} \qquad \text{and} \qquad R = \{r_1, r_2, \ldots, r_k\}$$

Then, there are exactly $n^2 k$ possible relational edges $e \in (V \times R \times V)$, and $E \subseteq V \times R \times V$ is a set of m elements. Hence, there are $\binom{n^2 k}{m}$ possible choices of E, and hence, at most $\binom{n^2 k}{m}$ non-isomorphic choices of G. Since k was chosen arbitrarily, this completes the proof. □

Lemma 22. *For each pair* (n, m)*, there exists a number* $C_{n,m}$ *such that the edge cover time, using the algorithm from Appendix A.8.2, of any knowledge graph in* $\mathbb{K}_{n,m}$ *is at most* $C_{n,m}$*.*

Proof. The result follows from Lemmas 20 and 21. As two isomorphic graphs have identical cover time, we can set $C_{n,m}$ to be the maximum of cover times of representatives of all isomorphic classes, which, by finiteness of both, is well-defined. □

Lemma 23. *Let* $G \in \mathbb{K}_{n,m}$ *be a graph,* $q = (h_q, r_q, ?)$ *be a link query over* G*, and* $\bar{\eta}$ *be a walk over* G*. If* $\bar{\eta}$ *traverses all edges of* G*, then using only the output* $w(\bar{\eta}; G, q, \cdot, \cdot)$ *of the recording function* w *detailed in Appendix A.8.2, we can construct a graph-query pair* (H, q') *isomorphic to* (G, q)*.*

Proof. Suppose that $\bar{\eta} = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \ldots \xrightarrow{r_\ell} v_\ell$ visits all edges of $G = (V, E, R)$ and let ℓ be its length. Recall the anonymization functions $\mathrm{id}_V(\cdot; \bar{\eta})$ and $\mathrm{id}_R(\cdot; \bar{\eta})$ as defined in Appendix A.8.2. The output $w(\bar{\eta}; G, q, \cdot, \cdot)$ (the embedding functions provided as the last two arguments are irrelevant) is a sequence of tuples $S_0, S_1, \ldots, S_\ell$ with each S_i equal to:

$$S_i = \big(\mathrm{id}_V(v_i; \bar{\eta}), \mathrm{id}_R(r_i; \bar{\eta}), \mathrm{dir}_i, \delta_{v_i = h_q}, \delta_{r_i = r_q}, \cdot, \cdot\big)$$

Consider a graph $H = (V', E', R')$ constructed as follows:

- the vertices V' correspond to the anonymized node ids $\mathrm{id}_V(v_i; \bar{\eta})$:

$$V' = \{\mathrm{id}_V(v; \bar{\eta}) \mid v \in V\}$$

Since each vertex must have been visited by $\bar{\eta}$, this is well-defined.

- the relation types R' are the anonymized relation ids $\mathrm{id}_R(r_i; \bar{\eta})$:

$$R' = \{\mathrm{id}_V(r; \bar{\eta}) \mid r \in R\}$$

  Again, this is well-defined, as each relation must have been noticed by $\bar{\eta}$.

- the edges E' are reconstructed from the consecutive step encodings using the anonymized vertex and relation indices and the direction dir_i:

$$\begin{aligned} E' = &\{(\mathrm{id}_V(v_{i-1}), \mathrm{id}_R(r_i), \mathrm{id}_V(v_i)) \mid \mathrm{dir}_i = 0, 1 \leq i \leq l\} \\ &\cup \{(\mathrm{id}_V(v_i), \mathrm{id}_R(r_i), \mathrm{id}_V(v_{i-1})) \mid \mathrm{dir}_i = 1, 1 \leq i \leq l\} \end{aligned}$$

and a query $q' = (\mathrm{id}_V(v_i; \bar{\eta}), \mathrm{id}_R(r_j, \bar{\eta}), ?)$ for i, j such that $\delta_{v_i=h_q} = 1$ and $\delta_{r_j=r_q} = 1$.

Then by the definition of w (Appendix A.8.2), it is straightforward that the pair $(\mathrm{id}_V(\cdot; \bar{\eta}), \mathrm{id}_R(\cdot; \bar{\eta}))$ defines an isomorphism from (G, q) to (H, q'). Indeed, both these functions are injective by construction, and as $\bar{\eta}$ witnesses all nodes and relations, they are well-defined bijections. For each unique edge traversed by $\bar{\eta}$, there exists a unique edge in E' translated to the anonymized space, which implies an isomorphism between E and E'. Finally, by utilizing the flags $\delta_{v_i=h_q}$ and $\delta_{r_j=r_q}$, we can identify the query head node and relation in the new graph. All things considered, we can reconstruct the pair (G, q), up to isomorphism, from the output of $w(\bar{\eta}; G, q, \cdot, \cdot)$. □

We are now ready to prove the main result regarding the universality of Flock as an approximation of link invariant functions. The outline of the proof is as follows: 1) Using the upper-bound on the edge cover time of graphs in $\mathbb{K}_{n,m}$ derived in Lemma 22, we can bound the probability of sampling a walk that visits all edges, 2) Once such a walk is sampled, we can recover the graph and query, up to isomorphism, from its anonymized form (Lemma 23), 3) Lastly, we can return the value of the approximated function for the derived isomorphic instance. Since the approximated function is link invariant, if the reconstructed graph matches the original one, we return precisely the correct value.

Proposition 29 (Proposition 15)**.** *With a powerful enough sequence processor f_θ, the Flock framework described in Section 4.2.3 is a universal approximator of link invariant functions over $\mathbb{K}_{n,m}$ for all pairs (n, m).*

Proof. Let $\varphi : \mathbb{K}_{n,m} \to (V \times R \times V \to [0, 1])$ be a link invariant function over $\mathbb{K}_{n,m}$ returning values from the interval $[0, 1]$. Let $G = (V, E, R) \in \mathbb{K}_{n,m}$, $q = (h, r, ?)$ be a link prediction query over G and $t \in V$ be a target node. Pick some $\epsilon, \delta > 0$. Our goal is to show that:

$$\mathbb{P}(|\varphi(G)((h, r, t)) - X_\theta(G, (h, r, ?))(t)| < \epsilon) > 1 - \delta \tag{A.91}$$

For simplicity, let us consider a situation where only a single walk $\bar{\eta}$ of length ℓ is sampled by the model (otherwise, omit additional walks). We will also restrict the argument to a single refinement case – the result can be extended to multiple refinement steps by returning $\Delta\mathbf{v}, \Delta\mathbf{r} = 0$ during all additional iterations. Consider a sequence processor f_θ that given the output $w(\bar{\eta}; G, q, \cdot, \cdot)$ of the recording protocol, creates a graph-query pair (H, q') with $q' = (h_{q'}, r_{q'}, ?)$ using the strategy described in the proof of Lemma 23, and returns a vector $\mathbf{h} \in \mathbb{R}^{l+1}$ whose i^{th} entry is equal $\mathbf{h}_i = \varphi(H)((h_{q'}, r_{q'}, \mathrm{id}_V(v_i; \bar{\eta}))$ where v_i is the i^{th} node visited by $\bar{\eta}$. The consensus protocol c, provided t was visited by $\bar{\eta}$, can then identify t as one of the v_j and pull the corresponding embedding $\mathbf{h}_j = \varphi(H)((h_{q'}, r_{q'}, \mathrm{id}_V(t; \bar{\eta}))$,

returning it as the output $\mathbf{v}(t) = \mathbf{h}_j$ (note that no matter which specific value of j is chosen, this value will be the same). Finally, the classification head can work as an identity operation, returning $X_\theta(G, q)(t) = \mathbf{v}(t) = \varphi(H)((h_{q'}, r_{q'}, \mathrm{id}_V(t; \bar{\eta}))$.

We claim that if the sampled walk $\bar{\eta}$ traverses all edges of G, then the output of the Flock model described above satisfies:

$$\varphi(G)((h, r, t)) = X_\theta(G, (h, r, ?))(t)$$

By Lemma 23, in such case, the reconstructed pair (H, q') is isomorphic to (G, q) by the isomorphism $id = (\mathrm{id}_V(\cdot; \bar{\eta}), \mathrm{id}_R(\cdot; \bar{\eta}))$. Since φ is link invariant, we can write:

$$\begin{aligned}
\varphi(G)((h, r, t)) &= \varphi(\mathrm{id}(G))((\mathrm{id}_V(h; \bar{\eta}), \mathrm{id}_R(r; \bar{\eta}), \mathrm{id}_V(t; \bar{\eta}))) \\
&= \varphi(H)((h_{q'}, r_{q'}, \mathrm{id}_V(t; \bar{\eta}))) \\
&= X_\theta(G, (h, r, ?))(t)
\end{aligned}$$

Therefore, whenever the walk $\bar{\eta}$ witnesses all edges of G, the output of the Flock model satisfies:

$$\varphi(G)((h, r, t)) = X_\theta(G, (h, r, ?))(t)$$

Hence, to show Equation (A.91), it suffices to prove that we can uniformly choose the length ℓ of the random walk so that the probability of $\bar{\eta}$ covering all the edges is greater than $1 - \delta$. By Markov's inequality:

$$\begin{aligned}
\mathbb{P}(\bar{\eta} \text{ does not cover all edges}) &= \mathbb{P}(\text{it takes } > \ell \text{ steps for } \eta \text{ to cover edges of } G) \\
&\leq \frac{\mathbb{E}[\#\text{steps such that } \eta \text{ covers all edges of } G]}{\ell} \\
&= \frac{C_E(G)}{\ell}
\end{aligned}$$

But by Lemma 22, $C_E(G) \leq C_{n,m}$ for some constant $C_{n,m}$. Hence, taking $\ell > \frac{C_{n,m}}{\delta}$, we get:

$$\mathbb{P}(\bar{\eta} \text{ does not cover all edges}) \leq \frac{C_E(G)}{\ell} \leq \frac{C_{n,m}}{\ell} < \delta$$

This means that for such a choice of ℓ:

$$\mathbb{P}(\bar{\eta} \text{ witnesses all edges of } G) > 1 - \delta$$

which leads to the conclusion that for $\ell > \frac{C_{n,m}}{\delta}$, the proposed Flock framework satisfies:

$$\mathbb{P}(|\varphi(G)((h, r, t)) - X_\theta(G, (h, r, ?))(t)| < \epsilon) > 1 - \delta$$

for any choice of $G = (V, E, R) \in \mathbb{K}_{n,m}$ and $(h, r, t) \in V \times R \times V$. □

Proof of invariance (Proposition 16)

First, let us recall the definition of invariance for the context of knowledge graphs and the associated notion of invariance in probability. We say that a function φ taking KGs as input is invariant if for any pair of isomorphic KGs $G \simeq H$ it produces the same input, i.e. $G \simeq H \implies \varphi(G) = \varphi(H)$.

We extend the notion of invariance for further types of inputs, not limited to full knowledge

graphs, particularly to random walks and link prediction queries. Let $G = (V, E, R) \in \mathbb{K}_{n,m}$ and let $H = (V', E', R') \simeq G$ be a KG isomorphic to G via the isomorphism $\mu = (\pi, \phi)$. For any $h \in V$ and $r \in R$, we identify the link prediction query $q = (h, r, ?)$ in H using the isomorphism μ as:

$$\mu(q) = \mu((h, r, ?)) = (\pi(h), \phi(r), ?)$$

Similarly, let $\eta = v_0 \xrightarrow{r_1} \ldots \xrightarrow{r_\ell} v_\ell$ be a walk of length ℓ in G. The view of η with μ is defined as:

$$\mu\left(v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \ldots \xrightarrow{r_\ell} v_\ell\right) = \pi(v_0) \xrightarrow{\phi(r_1)} \pi(v_1) \xrightarrow{\phi(r_2)} \ldots \xrightarrow{\phi(r_\ell)} \pi(v_\ell)$$

Let f be a function taking inputs drawn from KGs. We call f invariant if for any pair of isomorphic graphs $G \overset{\mu}{\simeq} H$ and an associated isomorphism $\mu = (\pi, \phi)$, f satisfies

$$f(x) = f(\mu(x))$$

where x can be, e.g., a walk or link prediction query. In words, invariance means that the function preserves output under the re-identifications of the input graph and the induced transformations of queries and walks.

This notion extends to functions with multiple inputs, where we enforce the transformation on each graph-related input. For example, a function φ taking a KG, query and a d-dimensional vector is invariant if it satisfies:

$$\forall G \overset{\mu}{\simeq} H, q, \mathbf{v} \in \mathbb{R}^d : \quad \varphi(G, q, \mathbf{v}) = \varphi(\mu(G), \mu(q), \mathbf{v})$$

Following the definition of *invariance in probability*, provided in Section 4.2.2, we extend all the definitions above to the stochastic case, replacing equality ($=$) with equality in distribution ($\overset{d}{=}$).

We can now prove the main propositions stated in Section 4.2.4. Let's begin with the more general:

Proposition 30 (Proposition 16)**.** *Suppose that the walk sampling protocol η is invariant in probability and both the recording protocol w and the consensus protocol c are invariant. Then, regardless of the choice of the deterministic sequence processor f_θ, the corresponding Flock model is invariant in probability.*

Proof. Let $(V, E, R) = G \simeq H = (V', E', R')$ be isomorphic knowledge graphs with isomorphism $\mu = (\pi, \phi)$ transforming G into H. Our goal is to show that when the statement conditions are met for a Flock model X_θ with I refinement steps, then for any link prediction query $q = (h, r, ?)$ and any target node $t \in V$, the prediction of Flock for t over (G, q) is an identical random variable to the prediction for $\pi(t)$ over $(H, \mu(q))$, i.e.

$$X_\theta(G, q)(t) \overset{d}{=} X_\theta(H, \mu(q))(\pi(t))$$

where $\mu(q) = (\pi(h), \phi(r), ?)$. Recall that these predictions are defined as:

$$X_\theta(G, q)(t) \coloneqq \text{head}(\mathbf{v}^{(I)}(t) + \mathbf{r}^{(I)}(r))$$
$$X_\theta(H, \mu(q))(\pi(t)) \coloneqq \text{head}(\mathbf{v}'^{(I)}(\pi(t)) + \mathbf{r}'^{(I)}(\phi(r)))$$

As head is a deterministic map, it suffices to show that the final embeddings $\mathbf{v}^{(I)}, \mathbf{r}^{(I)}$ for (G, q) and $\mathbf{v}'^{(I)}, \mathbf{r}'^{(I)}$ for $(H, \mu(q))$ satisfy:

$$\mathbf{v}^{(I)}(v) \overset{d}{=} \mathbf{v}'^{(I)}(\pi(v)) \quad \text{and} \quad \mathbf{r}^{(I)}(r) \overset{d}{=} \mathbf{r}'^{(I)}(\phi(r)) \qquad \forall v \in V, r \in R$$

We will prove this result by induction on the number of layers i. The base case $i = 0$ is trivial, as we initialize the embeddings of all nodes with a pretrained vector $\mathbf{v}_0$, and all relations with $\mathbf{r}_0$.

For the induction step, suppose the claim holds for i. We drop the superscript (i) for readability. The result for $i + 1$ becomes apparent by unfolding the definitions of invariance of the considered components. Since η is invariant in probability, we have

$$\mu(\eta(G)) \overset{d}{=} \eta(H) \tag{A.92}$$

Let $\eta_1, \ldots, \eta_n$ be the random walks over G using η and $\eta'_1, \ldots, \eta'_n$ be random walks over H. Now, $\eta_1, \ldots, \eta_n$ are independent and identically distributed random variables, each following the distribution $\eta_j \sim \eta(G)$. Similarly, using Equation (A.92):

$$\eta'_j \sim \eta(H) \overset{d}{=} \mu(\eta(G)) \implies \eta'_j \overset{d}{=} \mu(\eta_j) \tag{A.93}$$

As the recording protocol w is invariant, $w(\eta_j) = w(\mu(\eta_j))$ for all j, which with Equation (A.93) yields:

$$\mathbf{z}_j := w(\eta_j) = w(\mu(\eta_j)) \overset{d}{=} w(\eta'_j) := \mathbf{z}'_\mathbf{j} \tag{A.94}$$

Then, f_θ is a deterministic map, so Equation (A.94) implies:

$$\mathbf{h}_j := f_\theta(\mathbf{z}_j) \overset{d}{=} f_\theta(\mathbf{z}'_j) := \mathbf{h}'_j$$

Let $(\Delta\mathbf{v}, \Delta\mathbf{r}) = c(\mathbf{h}_{1:N}, \eta_{1:N})$, $(\Delta\mathbf{v}', \Delta\mathbf{r}') = c(\mathbf{h}'_{1:N}, \eta'_{1:N})$ be the outputs of the consensus protocol. We will denote by $c_\mathbf{v}$ and $c_\mathbf{r}$, the restrictions to the first and second output, e.g. $\Delta\mathbf{v} = c_\mathbf{v}(\mathbf{h}_{1:N}, \eta_{1:N})$. Let $\mathbf{x} \in \mathbb{R}^d$ be a vector, and denote by $\mathcal{W}(G)$ the space of walks over G. For any vertex $v \in V$, the probability that $\Delta\mathbf{v}(v) = \mathbf{x}$ equals:

$$\begin{aligned}
\mathbb{P}(\Delta\mathbf{v}(v) = \mathbf{x}) &= \sum_{\bar{\eta} \in \mathcal{W}(G)^n} \mathbb{P}(\Delta\mathbf{v}(v) = \mathbf{x} | \eta_{1:N} = \bar{\eta}) \cdot \mathbb{P}(\eta_{1:N} = \bar{\eta}) \\
&= \sum_{\bar{\eta} \in \mathcal{W}(G)^n} \mathbb{P}(c_\mathbf{v}(\mathbf{h}_{1:N}, \eta_{1:N}) = \mathbf{x} | \eta_{1:N} = \bar{\eta}) \cdot \mathbb{P}(\eta_{1:N} = \bar{\eta}) \\
&= \sum_{\bar{\eta} \in \mathcal{W}(G)^n} \mathbb{P}(c_\mathbf{v}(f_\theta(w(\eta_{1:N})), \eta_{1:N}) = \mathbf{x} | \eta_{1:N} = \bar{\eta}) \cdot \mathbb{P}(\eta_{1:N} = \bar{\eta}) \\
&= \sum_{\bar{\eta} \in \mathcal{W}(G)^n} \mathbb{P}(c_\mathbf{v}(f_\theta(w(\bar{\eta})), \bar{\eta}) = \mathbf{x} | \eta_{1:N} = \bar{\eta}) \cdot \mathbb{P}(\eta_{1:N} = \bar{\eta}) \\
&= \sum_{\substack{\bar{\eta} \in \mathcal{W}(G)^n \\ c_\mathbf{v}(f_\theta(w(\bar{\eta})), \bar{\eta})(v) = \mathbf{x}}} \mathbb{P}(\eta_{1:N} = \bar{\eta})
\end{aligned}$$

Similarly, we can derive:

$$\mathbb{P}(\Delta\mathbf{v}'(\pi(v)) = \mathbf{x}) = \sum_{\substack{\bar{\eta}' \in \mathcal{W}(H)^n \\ c_\mathbf{v}(f_\theta(w(\bar{\eta}')), \bar{\eta}')(\pi(v)) = \mathbf{x}}} \mathbb{P}(\eta'_{1:N} = \bar{\eta}')$$

Using the invariance of the consensus protocol and the invariance of $f_\theta \circ w$, we can write:

$$\begin{aligned} c_{\mathbf{v}}(f_\theta(w(\bar{\eta}')), \bar{\eta}')(\pi(v)) &= c_{\mathbf{v}}(f_\theta(w(\mu(\bar{\eta}))), \mu(\bar{\eta}))(\pi(v)) \\ &= c_{\mathbf{v}}(f_\theta(w(\bar{\eta})), \mu(\bar{\eta}))(\pi(v)) \\ &= c_{\mathbf{v}}(f_\theta(w(\bar{\eta})), \bar{\eta})(v) \end{aligned}$$

The graph isomorphism μ defines a bijection between walks $\mathcal{W}(G)$ in G and walks $\mathcal{W}(H)$ in H, so we can use this correspondence to deduce:

$$\begin{aligned} \mathbb{P}(\Delta\mathbf{v}'(\pi(v)) = \mathbf{x}) &= \sum_{\substack{\bar{\eta}' \in \mathcal{W}(H)^n \\ c_{\mathbf{v}}(f_\theta(w(\bar{\eta}')), \bar{\eta}')(\pi(v)) = \mathbf{x}}} \mathbb{P}(\eta'_{1:N} = \bar{\eta}') \\ &= \sum_{\substack{\mu(\bar{\eta}) \in \mathcal{W}(H)^n \\ c_{\mathbf{v}}(f_\theta(w(\mu(\bar{\eta}))), \mu(\bar{\eta}))(\pi(v)) = \mathbf{x}}} \mathbb{P}(\eta'_{1:N} = \mu(\bar{\eta})) \\ &= \sum_{\substack{\bar{\eta} \in \mathcal{W}(G)^n \\ c_{\mathbf{v}}(f_\theta(w(\bar{\eta})), \bar{\eta})(v) = \mathbf{x}}} \mathbb{P}(\eta'_{1:N} = \mu(\bar{\eta})) \end{aligned} \tag{A.95}$$

Since η is invariant in probability, $\mathbb{P}(\eta_{1:N} = \bar{\eta}) = \mathbb{P}(\eta'_{1:N} = \mu(\bar{\eta}))$. Applying this to Equation (A.95) yields:

$$\begin{aligned} \mathbb{P}(\Delta\mathbf{v}'(\pi(v)) = \mathbf{x}) &= \sum_{\substack{\bar{\eta} \in \mathcal{W}(G)^n \\ c_{\mathbf{v}}(f_\theta(w(\bar{\eta})), \bar{\eta})(v) = \mathbf{x}}} \mathbb{P}(\eta'_{1:N} = \mu(\bar{\eta})) \\ &= \sum_{\substack{\bar{\eta} \in \mathcal{W}(G)^n \\ c_{\mathbf{v}}(f_\theta(w(\bar{\eta})), \bar{\eta})(v) = \mathbf{x}}} \mathbb{P}(\eta_{1:N} = \bar{\eta}) = \mathbb{P}(\Delta\mathbf{v}(v) = \mathbf{x}) \end{aligned}$$

As $\mathbf{x}$ was chosen arbitrarily, we can conclude that $\Delta\mathbf{v}(v) \overset{d}{=} \Delta\mathbf{v}'(\pi(v))$. The proof for relations follows analogously, considering $c_{\mathbf{r}}$ instead of $c_{\mathbf{v}}$. This allows us to write:

$$\begin{aligned} \Delta\mathbf{v}(v) &\overset{d}{=} \Delta\mathbf{v}'(\pi(v)) \qquad \forall v \in V \\ \Delta\mathbf{r}(r) &\overset{d}{=} \Delta\mathbf{r}'(\phi(r)) \qquad \forall r \in R \end{aligned} \tag{A.96}$$

By the induction hypothesis, $\mathbf{v}^{(i)}(v) \overset{d}{=} \mathbf{v}'^{(i)}(\pi(v))$ for all $v \in V$ and $\mathbf{r}^{(i)}(r) \overset{d}{=} \mathbf{r}'^{(i)}(r)$ for all $r \in R$. Therefore, by Equation (A.96), combined with properties of sums of random variables:

$$\begin{aligned} \mathbf{v}^{(i+1)}(v) := \mathbf{v}^{(i)}(v) + \Delta\mathbf{v}(v) &\overset{d}{=} \mathbf{v}'^{(i)}(\pi(v)) + \Delta\mathbf{v}'(\pi(v)) := \mathbf{v}'^{(i+1)}(\pi(v)) \qquad \forall v \in V \\ \mathbf{r}^{(i+1)}(r) := \mathbf{r}^{(i)}(r) + \Delta\mathbf{r}(r) &\overset{d}{=} \mathbf{r}'^{(i)}(\phi(r)) + \Delta\mathbf{r}'(\phi(r)) := \mathbf{r}'^{(i+1)}(\phi(r)) \qquad \forall r \in R \end{aligned}$$

which completes the induction step, and hence the proof. □

We can use the conclusion from Proposition 16 to prove the probabilistic invariance of the architecture proposed in Section 4.2.3. To be able to apply it, we first need to verify the invariance of all used components, which we formalize in the following lemmas.

Lemma 24. *The choice of the first step $v_0 \xrightarrow{r_1} v_1$ of the uniform random walk algorithm described in Appendix A.8.2 is invariant.*

Proof. Let $G = (V, E, R)$ be a graph and let $H \simeq G$ be an isomorphic graph, with the isomorphism $\mu = (\pi, \phi)$ taking G to H. Consider a link prediction query $q = (h, r, ?)$ over G, and its identification

$q' = \mu(q) = (\pi(h), \phi(r), ?)$. The goal is to show that when using η described in Appendix A.8.2 for (G, q) and (H, q'), the first steps:

$$V_0 \xrightarrow{R_1} V_1 \qquad \text{and} \qquad U_0 \xrightarrow{S_1} U_1$$

of the execution of η over G and H, respectively, satisfy the following property:

$$\pi(V_0) \xrightarrow{\phi(R_1)} \pi(V_1) \overset{d}{=} U_0 \xrightarrow{S_1} U_1$$

By definition of η, there are three scenarios of choosing the first step, each with probability $\frac{1}{3}$. Hence, it suffices to show that within each scenario, the selection process is invariant in probability:

- **Scenario 1:** selecting the query head as the first node, then proceeding by random. First, π takes the head node of q to the head node of q'. Secondly, as isomorphisms preserve the number of neighboring nodes and number of edges between a pair of nodes, we have:

$$\begin{aligned}
\mathbb{P}(V_1 = v \mid V_0 = h) &= \begin{cases} \frac{1}{|\mathcal{N}_h|} & \text{if } v \in \mathcal{N}_h \\ 0 & \text{if } v \notin \mathcal{N}_h \end{cases} \\
&= \begin{cases} \frac{1}{|\mathcal{N}_{\pi(h)}|} & \text{if } \pi(v) \in \mathcal{N}_{\pi(h)} \\ 0 & \text{if } \pi(v) \notin \mathcal{N}_{\pi(h)} \end{cases} \quad = \mathbb{P}(U_1 = \pi(v) \mid U_0 = \pi(h))
\end{aligned}$$

  and

$$\begin{aligned}
\mathbb{P}(R_1 = r \mid V_1 = w) &= \begin{cases} \frac{1}{|E_{(w,h)}|} & \text{if } r(w,h) \in E_{(w,h)} \\ 0 & \text{otherwise} \end{cases} \\
&= \begin{cases} \frac{1}{|E_{(\pi(w),\pi(h))}|} & \text{if } \phi(r)(\pi(w),\pi(h)) \in E_{(\pi(w),\pi(h))} \\ 0 & \text{otherwise} \end{cases} \\
&= \mathbb{P}(S_1 = \phi(r) \mid U_1 = \pi(w))
\end{aligned}$$

- **Scenario 2:** selecting an edge with query relation type at random. Here, we use the fact that isomorphisms preserve the number of edges of a given type. Hence, μ defines a bijection between the sets of edges with type r in G and type $\phi(r)$ in H, which allows us to conclude that this scenario is also invariant in probability.

- **Scenario 3:** selecting the first step completely at random. This case is similar to Scenario 1 – using the invariance of the number of neighboring nodes under isomorphism, we can repeat similar calculations in a straightforward manner to show probabilistic invariance.

Either way, we find that the selection process of the first step of η over G translates naturally via μ to the choice of the first step over H, proving the desired statement. □

Lemma 25. *Suppose that the first step $v_0 \xrightarrow{r_1} v_1$ is chosen in an invariant manner. Then, the uniform random walk with no backtracking algorithm η is invariant in probability.*

Proof. Let $G = (V, E, R)$ be a knowledge graph, and let ℓ be the length of random walks. Let H be a KG isomorphic to G via the isomorphism $\mu = (\pi, \phi)$. We aim to show that:

$$\mu(\eta(G, \ell)) = \pi(V_0) \xrightarrow{\phi(R_1)} \pi(V_1) \xrightarrow{\phi(R_2)} \dots \xrightarrow{\phi(R_\ell)} \pi(V_\ell) \overset{d}{=} U_0 \xrightarrow{S_1} U_1 \xrightarrow{S_2} \dots \xrightarrow{S_\ell} U_\ell = \eta(H, \ell)$$

Let $\bar{\eta} = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \dots \xrightarrow{r_\ell} v_\ell \in \mathcal{W}(G)$ be a walk of length ℓ over G. It suffices to show that the

probability of sampling $\bar{\eta}$ from G is identical to the probability of sampling $\mu(\bar{\eta})$ from H:

$$\mathbb{P}(\eta(G,\ell)=\bar{\eta}) = \mathbb{P}(\eta(H,\ell)=\mu(\bar{\eta}))$$

To see this, let us expand the definitions of $\mathbb{P}(\eta(G,\ell)=\bar{\eta})$:

$$\begin{aligned}\mathbb{P}(\eta(G,\ell)=\bar{\eta}) =& \mathbb{P}(V_0=v_0)\\ &\cdot \mathbb{P}(V_1=v_1 \mid V_0=v_0)\\ &\cdot \prod_{i=0}^{\ell-2} \mathbb{P}(V_{i+2}=v_{i+2} \mid V_{i+1}=v_{i+1}, V_i=v_i)\\ &\cdot \prod_{i=0}^{\ell-1} \mathbb{P}(R_{i+1}=r_{i+1} \mid V_{i+1}=v_{i+1}, V_i=v_i)\end{aligned} \tag{A.97}$$

and $P(\eta(H,\ell)=\mu(\bar{\eta}))$:

$$\begin{aligned}\mathbb{P}(\eta(H,\ell)=\mu(\bar{\eta})) =& \mathbb{P}(U_0=\pi(v_0))\\ &\cdot \mathbb{P}(U_1=\pi(v_1) \mid U_0=\pi(v_0))\\ &\cdot \prod_{i=0}^{\ell-2} \mathbb{P}(U_{i+2}=\pi(v_{i+2}) \mid U_{i+1}=\pi(v_{i+1}), U_i=\pi(v_i))\\ &\cdot \prod_{i=0}^{\ell-1} \mathbb{P}(S_{i+1}=\phi(r_{i+1}) \mid U_{i+1}=\pi(v_{i+1}), U_i=\pi(v_i))\end{aligned} \tag{A.98}$$

Given that the graph isomorphism preserves the number of neighbors for each node and is a bijection, we can easily verify using the definitions from Equation (A.102) that the following indeed hold:

$$\begin{aligned}\mathbb{P}(V_{i+2}=v_{i+2} \mid V_{i+1}=v_{i+1}, V_i=v_i) &= \mathbb{P}(U_{i+2}=\pi(v_{i+2}) \mid U_{i+1}=\pi(v_{i+1}), U_i=\pi(v_i))\\ \mathbb{P}(R_{j+1}=r_{j+1} \mid V_{j+1}=v_{j+1}, V_j=v_j) &= \mathbb{P}(S_{j+1}=\phi(r_{j+1}) \mid U_{j+1}=\pi(v_{j+1}), U_j=\pi(v_j))\end{aligned} \tag{A.99}$$

for all $i \in \{0,1,\ldots,\ell-2\}, j \in \{1,\ldots,\ell-1\}$. Moreover, by the assumption that the first step $V_0 \xrightarrow{R_1} V_1$ is invariant, we have:

$$\mathbb{P}((V_0,R_1,V_1)=(v_0,r_1,v_1)) = \mathbb{P}((U_0,S_1,U_1)=(\pi(v_0),\phi(r_1),\pi(v_1))) \tag{A.100}$$

But by the laws of conditional probability:

$$\begin{aligned}\mathbb{P}((V_0,R_1,V_1)=(v_0,r_1,v_1)) &= \mathbb{P}(R_1=r_1 \mid V_0=v_0, V_1=v_1)\cdot \mathbb{P}(V_0=v_0, V_1=v_1)\\ &= \mathbb{P}(R_1=r_1 \mid V_0=v_0, V_1=v_1)\cdot \mathbb{P}(V_1=v_0 \mid V_0=v_0)\cdot \mathbb{P}(V_0=v_0)\end{aligned}$$

and analogously:

$$\begin{aligned}&\mathbb{P}((U_0,S_1,U_1)=(\pi(v_0),\phi(r_1),\pi(v_1)))\\ &\quad= \mathbb{P}(S_1=\phi(r_1) \mid U_0=\pi(v_0), U_1=\pi(v_1))\cdot \mathbb{P}(U_1=\pi(v_0) \mid U_0=\pi(v_0))\cdot \mathbb{P}(U_0=\pi(v_0))\end{aligned}$$

Substituting these equalities into Equation (A.100) and multiplying both sides by the equalities from Equation (A.99) for all choices of $i \in \{0,1,\ldots,\ell-2\}, j \in \{1,\ldots,\ell-1\}$, we get precisely the equality of

the right sides of equations Equation (A.97) and Equation (A.98). Hence,

$$\mathbb{P}(\eta(G,\ell)=\bar{\eta})=\mathbb{P}(\eta(H,\ell)=\mu(\bar{\eta}))$$

and we can conclude that $\mu(\eta(G,\ell)) \stackrel{d}{=} \eta(H,\ell)$, and the algorithm η is invariant in probability. □

Corollary 5. *The random walk algorithm presented in Appendix A.8.2 is invariant in probability.*

Lemma 26. *The recording protocol w, as described in Appendix A.8.2, is invariant, provided that the embedding functions $\mathbf{v}$ and $\mathbf{r}$ are invariant.*

Proof. Let $G=(V,E,R)$ and $H=(V',E',R')$ be isomorphic knowledge graphs with the isomorphism $\mu=(\pi,\phi)$ taking G to H. Let $q=(h_q,r_q,?)$ be a link prediction query over G, and $\mu(q)=(\pi(h_q),\phi(r_q),?)$ be its identification in H. Let $\bar{\eta}=v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \ldots \xrightarrow{r_\ell} v_\ell \in \mathcal{W}(G)$ be a walk over G, and $\bar{\eta}'=\mu(\bar{\eta})=\pi(v_0) \xrightarrow{\phi(r_1)} \pi(v_1) \xrightarrow{\phi(r_2)} \ldots \xrightarrow{\phi(r_\ell)} \pi(v_\ell)$ be the analogous walk over H. To prove that the recording protocol w outlined in Appendix A.8.2 is invariant, it suffices to show that the encoding of each step:

$$S_i=\big(\mathrm{id}_V(v_i;\bar{\eta}),\mathrm{id}_R(r_i;\bar{\eta}),\mathrm{dir}_i,\delta_{v_i=h_q},\delta_{r_i=r_q},\mathbf{v}(v_i),\mathbf{r}(r_i)\big)$$

is identical for $\bar{\eta}$ and $\bar{\eta}'$. We will show this for each component:

- since π defines a bijection between nodes in G and H, for any i, we have:

$$\mathrm{id}_V(v_i;\bar{\eta})=\arg\min_t[v_t=v_i]=\arg\min_t[\pi(v_t)=\pi(v_i)]=\mathrm{id}_V(\pi(v_i);\bar{\eta}')$$

- similarly to the point above, ϕ is a bijection between relations of G and H, so we can write:

$$\begin{aligned}\mathrm{id}_R(r_i;\bar{\eta}) &= \arg\min_t\left[r_t=r_i \vee r_t=r_i^{-1}\right]\\ &= \arg\min_t\left[\phi(r_t)=\phi(r_i) \vee \phi(r_t)=\phi(r_i)^{-1}\right]\\ &= \mathrm{id}_R(\phi(r_i);\bar{\eta}')\end{aligned}$$

- dir_i is clearly preserved, as the isomorphism μ preserves the directions of edges,

- as π,ϕ are bijections the masks $\delta_{v_i=h_q},\delta_{r_i=r_q}$, representing whether the i'th node and relation match the types in the query, satisfy:

$$\begin{aligned}v_i=h_q &\iff \pi(v_i)=\pi(h_q) \quad\implies\quad \delta_{v_i=h_q}=\delta_{\pi(v_i)=\pi(h_q)}\\ r_i=r_q &\iff \phi(r_i)=\phi(r_q) \quad\implies\quad \delta_{r_i=r_q}=\delta_{\phi(r_i)=\phi(r_q)}\end{aligned}$$

- $\mathbf{v}$ and $\mathbf{r}$ are invariant by assumption, so:

$$\mathbf{v}(v_i)=\mathbf{v}(\pi(v_i)) \qquad \text{and} \qquad \mathbf{r}(r_i)=\mathbf{r}(\phi(r_i))$$

Combining all these observations, we can conclude that $w(\bar{\eta};G,q,\mathbf{v},\mathbf{r})=w(\mu(\bar{\eta});H,\mu(q),\mathbf{v},\mathbf{r})$ and w is indeed invariant. □

Lemma 27. *The consensus protocol c, as described in Appendix A.8.2, is invariant.*

Proof. Let $G = (V, E, R)$ be a knowledge graph and H be isomorphic to G via an isomorphism $\mu = (\pi, \phi)$. Let $\bar{\eta}_{1:N} \in \mathcal{W}(G)$ be a sequence of walks in G. To show that the output of the consensus protocol is invariant, we need to prove that for each $v \in V$ and $r \in R$, the following holds:

$$\Delta\mathbf{v}(v) = \mathbf{v}'(\pi(v)) \qquad \text{and} \qquad \Delta\mathbf{r}(r) = \Delta\mathbf{r}'(\phi(r)) \tag{A.101}$$

where $(\Delta\mathbf{v}, \Delta\mathbf{r}) = c(\mathbf{h}, \bar{\eta}_{1:N})$ and $(\Delta\mathbf{v}', \Delta\mathbf{r}') = c(\mathbf{h}, \mu(\bar{\eta}_{1:N}))$ for $\mathbf{h} = (\mathbf{s}^{(V)}, \mathbf{s}^{(R)}, \mathbf{a}^{(V)}, \mathbf{a}^{(R)})$.

The result follows from the fact that π and ϕ are bijections – whenever v is the j^{th} vertex visited in the walk $\bar{\eta}_i$, the j^{th} node of $\mu(\bar{\eta}_i)$ must be $\pi(v)$ (and vice versa). An analogous result holds for the relations. Hence, the aggregation performed by c for v (resp. r) over $\bar{\eta}_{1:N}$ is equivalent to the aggregation for $\pi(v)$ (resp. $\phi(r)$) over $\mu(\bar{\eta}_{1:N})$, and Equation (A.101) is indeed satisfied. □

Proposition 31 (Proposition 17)**.** *Flock with components as described in Section 4.2.3 is invariant in probability.*

Proof. The result follows naturally from aggregating the results of Corollary 5 and Lemmas 26 and 27, followed by applying Proposition 16. □

A.8.2 Flock details

In this section, we expand on the descriptions of individual components of Flock summarized in Section 4.2.3: the random walk algorithm, the recording protocol, the sequence processor, and the consensus protocol.

Uniform random walk

Let $G = (V, E, R)$ be a knowledge graph, and let ℓ be the length of random walks. For each node $v \in V$, we will denote by $\mathcal{N}(v)$ the set of neighbors of v:

$$\mathcal{N}(v) = \{w \in V : \exists r \in R.(v, r, w) \in E \vee (w, r, v) \in E\}$$

and by $E(v, w)$, the set of relational edges from v to w (allowing for the inverse direction):

$$\begin{aligned} E(v, w) = &\{(v, r, w) \in R \times \{v\} \times \{w\} : (v, r, w) \in E\} \\ &\cup \{(v, r^{-1}, w) \in R^{-1} \times \{v\} \times \{w\} : (w, r, v) \in E\} \end{aligned}$$

where R^{-1} is the set symbolizing the inverses of relation types in R. The uniform random walk with no backtracking $\eta(G, \ell)$ of length ℓ over G, represented as:

$$V_0 \xrightarrow{R_1} V_1 \xrightarrow{R_2} \cdots \xrightarrow{R_\ell} V_\ell$$

is a second-order Markov process that follows the rules:

$$\mathbb{P}(V_{i+2} = v \mid V_{i+1} = w, V_i = u) = \begin{cases} 0 & \text{if } v = u \text{ and } |\mathcal{N}_w| > 1 \\ 1 & \text{if } v = u \text{ and } \mathcal{N}_w = \{u\} \\ \frac{1}{|\mathcal{N}_w|-1} & \text{if } v \neq u \text{ and } v \in \mathcal{N}_w \\ 0 & \text{if } v \notin \mathcal{N}_w \end{cases}$$
$$\mathbb{P}(R_{j+1} = r \mid V_{j+1} = w, V_j = u) = \begin{cases} \frac{1}{|E_{(w,u)}|} & \text{if } r(w,u) \in E_{(w,u)} \\ 0 & \text{otherwise} \end{cases} \tag{A.102}$$

for all $i \geq 0, j \geq 1$. Intuitively, at each step of the walk, we first select a neighbor (except for the vertex chosen one step ago) of the current node uniformly at random (disregarding multi-edges and edge directions), and then sample an edge between these two nodes uniformly at random. If the current node has only one neighbor, we are forced to return to it.

The initial conditions depend on the selected scenario. Given a query $q = (h, r, ?)$ over G, we can describe the process of selecting the first step $V_0 \xrightarrow{R_1} V_1$ as setting either (each with probability $\frac{1}{3}$):

- $V_0 = h$ and selecting the first step uniformly at random as described above, meaning:

$$\mathbb{P}(V_1 = v \mid V_0 = h) = \begin{cases} \frac{1}{|\mathcal{N}_h|} & \text{if } v \in \mathcal{N}_h \\ 0 & \text{if } v \notin \mathcal{N}_h \end{cases}$$
$$\mathbb{P}(R_1 = r \mid V_1 = w) = \begin{cases} \frac{1}{|E_{(w,h)}|} & \text{if } r(w,h) \in E_{(w,h)} \\ 0 & \text{otherwise} \end{cases}$$

- setting $R_1 = r$ and selecting $V_0 \xrightarrow{R_1} V_1$ uniformly at random from edges with type r.
- choosing V_0 uniformly at random, and then sampling the first step at random as well:

$$\mathbb{P}(V_0 = w) = \frac{1}{|V|}$$
$$\mathbb{P}(V_1 = w \mid V_0 = v) = \begin{cases} \frac{1}{|\mathcal{N}_w|} & \text{if } v \in \mathcal{N}_w \\ 0 & \text{if } v \notin \mathcal{N}_w \end{cases}$$
$$\mathbb{P}(R_1 = r \mid V_1 = v, V_0 = w) = \begin{cases} \frac{1}{|E_{(w,v)}|} & \text{if } r(w,v) \in E_{(w,v)} \\ 0 & \text{otherwise} \end{cases}$$

For the relation prediction objective, we add one more scenario, similar to the first one described above, but substituting $V_0 = t$ instead. For that problem, each scenario is chosen with probability $\frac{1}{4}$.

Recording function

Given a KG $G = (V, E, R)$, a query $q = (h_q, r_q, ?)$, a walk $\bar{\eta} = v_0 \xrightarrow{r_1} v_1 \xrightarrow{r_2} \ldots \xrightarrow{r_\ell} v_\ell$ of length ℓ over G, and a set of embeddings $\mathbf{v}$ of nodes V and $\mathbf{r}$ of relations R, our recording function w first splits the walk into a sequence of $\ell + 1$ steps:

$$(r_0, v_0), (r_1, v_1), \ldots (r_\ell, v_\ell)$$

with $r_0 = r_\varnothing$ being a special marker for no relation. Each step (r_i, v_i) is transformed into a 7-tuple:

$$S_i = \big(\mathrm{id}_V(v_i;\bar{\eta}), \mathrm{id}_R(r_i;\bar{\eta}), \mathrm{dir}_i, \delta_{v_i=h_q}, \delta_{r_i=r_q}, \mathbf{v}(v_i), \mathbf{r}(r_i)\big)$$

where:

- $\mathrm{id}_V(v_i;\bar{\eta})$ and $\mathrm{id}_R(r_i;\bar{\eta})$ are the anonymized id's of the node v_i and relation r_i, evaluated as:

$$\mathrm{id}_V(v_i;\bar{\eta}) = \arg\min_t [v_t = v_i]$$
$$\mathrm{id}_R(r_i;\bar{\eta}) = \arg\min_t \left[r_t = r_i \vee r_t = r_i^{-1}\right]$$

- dir_i denotes the direction in which we follow the edge. We set $\mathrm{dir}_i = 0$ if $r_i \in R$ (the edge is traversed from head to tail) and $\mathrm{dir}_i = 1$ if $r_i \in R^{-1}$ (the edge is taken in the reverse direction).

- $\delta_{v_i=h_q}$ and $\delta_{r_i \sim r_q}$ are binary flags representing whether the current node v_i is the query head v_q and if the relation r_i is either the queried relation r_q or its inverse r_q^{-1}.

- $\mathbf{v}(v_i), \mathbf{r}(r_i)$ are the embeddings of v_i and r_i, respectively.

The output of w for $\bar{\eta}$ given $G, q, \mathbf{v}, \mathbf{r}$ is then:

$$w(\bar{\eta}; G, q, \mathbf{v}, \mathbf{r}) = (S_0, S_1, \ldots, S_\ell)$$

Sequence processor

Once the sampled walks are anonymized by the recording protocol w, the output for each walk $\bar{\eta}_i$:

$$w(\bar{\eta}; G, q, \mathbf{v}, \mathbf{r}) = (S_0, S_1, \ldots, S_\ell)$$

is passed through the sequence processor f_θ, parametrized by the following modules:

- $\mathbf{A}_v, \mathbf{A}_r \in \mathbb{R}^{(\ell+1)\times d}$: embedding tables for anonymized vertices and relations, respectively,
- $\mathbf{D} \in \mathbb{R}^{2\times d}$: look-up table for the direction embedding,
- $\mathbf{Q}_h, \mathbf{Q}_r \in \mathbb{R}^{2\times d}$: embedding tables for the binary query labels,
- $\mathbf{V}, \mathbf{R} : \mathbb{R}^d \to \mathbb{R}^d$: linear maps applied to the passed embeddings of vertices and relations,
- $\boldsymbol{\Omega}$: a bi-directional GRU [Cho et al., 2014] cell equipped with RMSNorm [Zhang and Sennrich, 2019] and SwiGLU [Shazeer, 2020] activation function.

For each step, encoding S_i of the form:

$$S_i = \big(\mathrm{id}_V(v_i;\bar{\eta}_i), \mathrm{id}_R(r_i;\bar{\eta}_i), \mathrm{dir}_i, \delta_{v_i=h_q}, \delta_{r_i=r_q}, \mathbf{v}(v_i), \mathbf{r}(r_i)\big)$$

we evaluate the processed embedding $\mathbf{c_i}$ of S_i as a sum of the corresponding encoded components:

$$\begin{aligned}\mathbf{c}_i =& \mathbf{A}_v(\mathrm{id}_V(v_i;\bar{\eta}_i)) + \mathbf{A}_r(\mathrm{id}_R(r_i;\bar{\eta}_i)) + \mathbf{D}(\mathrm{dir}_i) \\ &+ \mathbf{Q_h}(\delta_{v_i=h_q}) + \mathbf{Q}_r(\delta_{r_i=r_q}) + \mathbf{V}(\mathbf{v}(v_i)) + \mathbf{R}(\mathbf{r}(r_i))\end{aligned}$$

These are then passed to the GRU cell $\mathbf{\Omega}$, which fuses the features across the whole walk and produces multi-head embeddings of vertices and relations, as well as the associated weights:

$$\left(\mathbf{s}_i^{(V)}, \mathbf{s}_i^{(R)}, \mathbf{a}_i^{(V)}, \mathbf{a}_i^{(R)}\right) = \mathbf{\Omega}([\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_\ell])$$

where $\mathbf{s}_i^{(V)}, \mathbf{s}_i^{(R)} \in \mathbb{R}^{(\ell+1)\times h\times d_h}$ and $\mathbf{a}_i^{(V)}, \mathbf{a}_i^{(R)} \in \mathbb{R}^{(\ell+1)\times h}$. Stacking all N of them gives us the final output of the sequence processor.

Consensus protocol

Given walks $\bar{\eta}_{1:N}$ over $G = (V, E, R)$ and the outputs $\mathbf{s}^{(V)}, \mathbf{s}^{(R)}, \mathbf{a}^{(V)}, \mathbf{a}^{(R)}$ of the sequence processor, the consensus protocol c aggregates the signal for each node by evaluating a weighted sum over the appearances of this node across the walks. More precisely, for each node $v \in V$, we find all pairs of indices (i, j), such that the j^{th} node visited in $\bar{\eta}_i$ was v, and concatenate the weighted sums of embeddings produced by each head, with weights exponentially proportional to the scores $\mathbf{a}^{(V)}$:

$$\Delta\mathbf{v}(v) = \bigoplus_{k=1}^{h} \frac{\sum\limits_{\substack{i,j \\ \bar{\eta}_i(v_j)=v}} \exp\left(\mathbf{a}_{i,j,k}^{(V)}\right) \cdot \mathbf{s}_{i,j,k}^{(V)}}{\sum\limits_{\substack{i,j \\ \bar{\eta}_i(v_j)=v}} \exp\left(\mathbf{a}_{i,j,k}^{(V)}\right)}$$

Similarly, we aggregate the encodings for relations, considering their occurrences in both directions:

$$\Delta\mathbf{r}(r) = \bigoplus_{k=1}^{h} \frac{\sum\limits_{\substack{i,j \\ \bar{\eta}_i(r_j)\in\{r,r^{-1}\}}} \exp\left(\mathbf{a}_{i,j,k}^{(R)}\right) \cdot \mathbf{s}_{i,j,k}^{(R)}}{\sum\limits_{\substack{i,j \\ \bar{\eta}_i(r_j)\in\{r,r^{-1}\}}} \exp\left(\mathbf{a}_{i,j,k}^{(R)}\right)}$$

In both formulas above, $\bigoplus$ denotes concatenation.

Additionally, we say that a consensus protocol c is *invariant* if for any pair of isomorphic KGs $G = (V, E, R)$ and $H = (V', E', R')$, any isomorphism $\mu = (\pi, \phi)$ from G to H, any list of embeddings $\mathbf{h}_{1:N}$ with $\mathbf{h}_i \in \mathbb{R}^d$, and any sequence of sampled walks $\bar{\eta}_{1:N}$ over G, the outputs

$$(\Delta\mathbf{v}, \Delta\mathbf{r}) = c(\mathbf{h}_{1:N}, \bar{\eta}_{1:N})$$
$$(\Delta\mathbf{v}', \Delta\mathbf{r}') = c(\mathbf{h}_{1:N}, \mu(\bar{\eta}_{1:N}))$$

satisfy:

$$\Delta\mathbf{v}(v) = \Delta\mathbf{v}'(\pi(v)) \qquad \forall v \in V$$
$$\Delta\mathbf{r}(r) = \Delta\mathbf{r}'(\phi(r)) \qquad \forall r \in R$$

Acknowledgments in Korean

저를 학생으로 받아주시고 가능성을 믿어 주신 홍승훈 교수님께 진심으로 감사드립니다. 교수님의 지도와 격려 덕분에 연구자로서 지적으로도, 인격적으로도 성장할 수 있었으며, 연구에 전념할 수 있는 환경을 조성해주셨기에 새롭고 도전적인 방향을 시도할 수 있었습니다. 학위논문의 심사위원으로 귀중한 시간과 조언을 내어주신 조경현 교수님, 양홍석 교수님, 안성수 교수님과 시아막 라반바흐시 교수님께도 감사드립니다. 학부생 당시 많은 가르침을 주셨던 김필남 교수님과 백세범 교수님께도 진심으로 감사드립니다.

연구적으로 성장하고 자유로운 연구실 문화를 만들어가는 경험을 함께한 재훈, 동균이형, 휘형 그리고 동훈이형에게 감사합니다. 덕분에 대학원 생활을 즐겁게 할 수 있었고 심층학습의 다양한 문제와 기술을 제 연구분야에 접목해서 생각해 볼 수 있었습니다. 앞으로 연구실을 이끌어갈 세민, 키엣, 찬혁, 지호님, 연우님, 선오님, 찬렬님, 그리고 초기 시행착오를 함께 겪어나간 수호형, 성웅, 성현, 원광님께도 감사드립니다.

학위과정 동안 훌륭한 동료 연구자들과의 협력과 교류로 혼자서는 어려웠을 일들을 해냈고 많이 배웠습니다. 올가, 우마르, 싱위에, 크시슈토프, 막스, 페타르, 지윤님, 경빈님, 성준님, 선우님, 세윤님, 티엔, 아이한, 영민님, 나윤님 그리고 이스마일, 마이클 브론스타인 교수님, 이홍락 교수님, 이주호 교수님께 감사드립니다. 특히 LG AI연구원에서 일할 기회를 주시고 지원과 격려를 아끼지 않으신 이문태 교수님께 감사드립니다.

지금까지의 여정을 지켜봐 준 친구들 수범, 상협, 상준, 재희, 지유, 승혁, 소영누나, 현철이형, 민기형, 준혁이형, 수현이형, 지수형, 유성이형, 수빈이형, 용진이형, 의범이형, 기목, 종우, 찬우 그리고 동생 성재에게도 감사의 말을 전합니다. 저에게 많은 응원이 되었듯 미래에 그들에게도 제가 힘이 되어줄 수 있으면 좋겠습니다. 여기에 미처 적지 못한 다른 모든 친구들에게도 진심으로 감사를 전하고 싶습니다.

마지막으로, 이 자리를 빌어 저의 아버지 김종철과, 지금은 작고하신 저의 어머니 이영희께 깊은 감사의 마음을 올리고자 합니다. 두 분의 사랑과 헌신을 잊지 않고 베풀며 살아가겠습니다.

Acknowledgment

I am deeply grateful to my advisor Seunghoon Hong for accepting me as his student and believing in my potential. His guidance and encouragement helped me grow as a researcher, and thanks to him fostering an environment where I could commit to my research, I was able to pursue new and challenging directions. I would also like to sincerely thank Kyunghyun Cho, Hongseok Yang, Sungsoo Ahn, and Siamak Ravanbakhsh for their valuable time and advice as members of my dissertation committee. I am also grateful to Pilnam Kim and Se-Bum Paik for their mentorship during my undergraduate years.

I would like to thank Jaehoon, Donggyun, Whie, and Dong Hoon for growing together as researchers and building a supportive and open lab culture. Thanks to them, my experience as a graduate student was much more enjoyable and I was able to explore how various fields in deep learning relate to my research area. I also thank Semin, Kiet, Chanhyuk, Jiho, Yeonwoo, Sunoh, and Chanryeol, who will lead the lab going forward, as well as Suho, Seongwoong, Sunghyun, and Wonkwang, who went through the early trials and errors with me.

Throughout my PhD journey, collaboration and exchange with excellent fellow researchers allowed me to accomplish what would have been difficult alone, and I learned a lot from them. My heartfelt thanks go to Olga, Oumar, Xingyue, Krzysztof, Max, Petar, Jiyun, Kyungbin, Sungjun, Seonwoo, Saeyoon, Tien, Ayhan, Youngmin, Nayun, İsmail İlkan Ceylan, Michael Bronstein, Honglak Lee, and Juho Lee. I am especially grateful to Moontae Lee for giving me the opportunity to work at LG AI Research and for his generous support and encouragement.

I would also like to express my gratitude to all of my friends who have been with me throughout this journey. Just as their support has meant so much to me, I hope to be a source of strength for them in the future.

Finally, I want to take this opportunity to express my deepest gratitude to my father, Jongchul Kim and, to my mother, Younghee Lee, who is no longer with us. I will live my life remembering and passing on their love and dedication.

JINWOO KIM

Ph.D. Student
Graph & Geometric DL

jw9730.github.io
jinwoo-kim@kaist.ac.kr

Education

M.S./Ph.D. in Computer Science — Mar 2021 – Feb 2026 (expected)
Korea Advanced Institute of Science and Technology (KAIST) — South Korea
- Advisor: Seunghoon Hong
- Research focus: Deep learning for graphs and geometric data.

B.S. in Brain Engineering and Computer Science (Double Major) — Mar 2016 – Feb 2021
Korea Advanced Institute of Science and Technology (KAIST) — South Korea
- GPA 4.05/4.3 (Summa Cum Laude)

Experience

New York University, Visiting Scholar — Nov 2025 – Jan 2026
Host: Kyunghyun Cho, Rajesh Ranganath — New York, NY, US

LG AI Research, Research Intern — Jan – Jul 2022
Host: Moontae Lee, Honglak Lee — South Korea

Publications

(P: preprint, C: conference, J: journal, W: workshop, *: equal contribution)

[P2] **Inverting Data Transformations via Diffusion Sampling**
Jinwoo Kim*, Sékou-Oumar Kaba*, Jiyun Park, Seunghoon Hong, Siamak Ravanbakhsh
Under review 2025

[P1] **Flock: A Knowledge Graph Foundation Model via Learning on Random Walks**
Jinwoo Kim*, Xingyue Huang*, Krzysztof Olejniczak, Kyungbin Min, Michael Bronstein, Seunghoon Hong, İsmail İlkan Ceylan
arXiv 2025

[C11] **Sequence Modeling with Spectral Mean Flows**
Jinwoo Kim, Max Beier, Petar Bevanda, Nayun Kim, Seunghoon Hong
NeurIPS 2025

[C10] **Revisiting Random Walks for Learning on Graphs**
Jinwoo Kim, Olga Zaghen*, Ayhan Suleymanzade*, Youngmin Ryou, Seunghoon Hong
ICLR 2025; ICML 2024 GRaM Workshop
Spotlight Presentation (380/11672=3.26%)

[C9] **3D Denoisers are Good 2D Teachers: Molecular Pretraining via Denoising and Cross-Modal Distillation**
Sungjun Cho, Dae-Woong Jeong, Sung Moon Ko, Jinwoo Kim, Sehui Han, Seunghoon Hong, Honglak Lee, Moontae Lee
AAAI 2025
Oral Presentation

[C8] **Simulation-Free Training of Neural ODEs on Paired Data**
Semin Kim*, Jaehoon Yoo*, Jinwoo Kim, Yeonwoo Cha, Saehoon Kim, Seunghoon Hong
NeurIPS 2024

[W1] **Learning Symmetrization for Equivariance with Orbit Distance Minimization**
Tien Dat Nguyen*, Jinwoo Kim*, Hongseok Yang, Seunghoon Hong
NeurIPS 2023 NeurReps Workshop

[C7] **Learning Probabilistic Symmetrization for Architecture Agnostic Equivariance**
Jinwoo Kim, Tien Dat Nguyen, Ayhan Suleymanzade, Hyeokjun An, Seunghoon Hong
NeurIPS 2023
Spotlight Presentation (378/12345=3.06%)

[C6] **Universal Few-shot Learning of Dense Prediction Tasks with Visual Token Matching**
Donggyun Kim, Jinwoo Kim, Seongwoong Cho, Chong Luo, Seunghoon Hong
ICLR 2023
Outstanding Paper Award (4/4955=0.08%)

[C5] **Pure Transformers are Powerful Graph Learners**
Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, Seunghoon Hong
NeurIPS 2022

[C4] **Transformers meet Stochastic Block Models: Attention with Data-Adaptive Sparsity and Cost**
Sungjun Cho, Seonwoo Min, Jinwoo Kim, Moontae Lee, Honglak Lee, Seunghoon Hong
NeurIPS 2022

[C3] **Equivariant Hypergraph Neural Networks**
Jinwoo Kim, Saeyoon Oh, Sungjun Cho, Seunghoon Hong
ECCV 2022

[C2] **Transformers Generalize DeepSets and Can be Extended to Graphs and Hypergraphs**
Jinwoo Kim, Saeyoon Oh, Seunghoon Hong
NeurIPS 2021

[C1] **SetVAE: Learning Hierarchical Composition for Generative Modeling of Set-Structured Data**
Jinwoo Kim*, Jaehoon Yoo*, Juho Lee, Seunghoon Hong
CVPR 2021

[J1] **Spontaneous Retinal Waves Can Generate Long-Range Horizontal Connectivity in Visual Cortex**
Jinwoo Kim*, Min Song*, Jaeson Jang, Se-Bum Paik
The Journal of Neuroscience 40(34) 2020

Honors

Awards

- **Outstanding Researcher Award**, KAIST-Mila Prefrontal AI Research Center — 2024
- **ELLIS Mobility Grant** [C10], ICML 2024 GRaM Workshop — 2024
- **Outstanding Paper Award** [C6], ICLR 2023 — 2023
- **Samsung Humantech Paper Award** [C6], Silver Prize — 2023
- **Qualcomm Innovation Fellowship Korea** [C2] — 2022
- **KAIST Engineering Innovator Award**, 1 of 5 Recipients in the College of Engineering — 2020

Scholarships and fellowships

- **Kwanjeong Education Foundation Scholarship** — 2022 – 2023
- **Korea National Science & Technology Scholarship** — 2018 – 2019
- **KAIST Alumni Fellowship** — 2017 – 2020
- **KAIST Presidential Fellowship** — 2016 – 2020
- **Hansung Scholarship for Gifted Students** — 2015 – 2016

Invited Talks

Sequence Modeling with Spectral Mean Flows [C11]
- Ben-Gurion University of the Negev (BGU) (Host: Ilan Naiman) — Dec 2025

Architecture-Agnostic Invariances for Deep Learning [C7, W1, C10]
- Mila – Quebec AI Institute (Host: Minsu Kim, Junyeob Baek) — Jul 2025
- KAIST AI899 Geometric DL (Host: Sungsoo Ahn) — May 2025
- Mila – Quebec AI Institute (Host: Siamak Ravanbakhsh, Sékou-Oumar Kaba) — Dec 2024
- KAIST–Mila Prefrontal AI Research Center (Host: Sungjin Ahn) — Nov 2024
- Sungkyunkwan University (SKKU) (Host: Chang Woo Myung) — Aug 2024
- Pohang University of Science and Technology (POSTECH) (Host: Sungsoo Ahn) — Nov 2023

Universal Few-shot Learning of Dense Prediction Tasks with Visual Token Matching [C6]
- KAIST–Samsung Electronics DS Division Exchange Meetup (Host: Chulmoo Kang) — Aug 2023

Pure Transformers are Powerful Graph Learners [C5]
- Microsoft USA (Host: Nabiha Asghar) — Jan 2023
- NeurIPS 2022 at KAIST (Host: Dongkwan Kim) — Nov 2022
- Learning on Graphs and Geometry Reading Group (LoGaG) (Host: Hannes Stärk) — Aug 2022

Higher-order Transformers for Sets, Graphs, and Hypergraphs [C2]
- Qualcomm Korea (Host: Jaewon Choi) — Jan 2023
- KAIST AI Workshop 21/22 (Host: Dongkwan Kim) — Jan 2022
- NeurIPS 2021 Social: ML in Korea (Host: Jung-Woo Ha) — Dec 2021

Hierarchical Variational Autoencoders for Generative Modeling of Sets [C1]
- Naver AI Author Meetup for CVPR 2021 (Host: Jung-Woo Ha) — Sep 2021
- Korean Conference on Computer Vision 2021 (Host: Jongwoo Lim) — Sep 2021

Retinal Waves and Prenatal Wiring of Primary Visual Cortex [J1]
- Society for Neuroscience, Chicago, IL, US — Oct 2019

Academic Services

Conference Reviewer: AISTATS 2025–2026, ICLR 2025–2026, NeurIPS 2022–2025, ICML 2023–2025, IJCNN 2025, LoG 2022–2025, TAG-DS 2025, IJCNN 2025, ICCV 2025 SP4V Workshop, ICML 2024 GRaM Workshop, CVPR 2022, ACCV 2022
Journal Reviewer: Neural Networks 2023, 2025, IJCV 2025, TMLR 2024

Teaching

Teaching Assistant, KAIST School of Computing
- Undergraduate Research Program (URP) — 2022, 2024
- Introduction to Deep Learning (CS492I) — 2021, 2022, 2023
- Computer Vision (CS576) — 2022, 2023
- School of Computing Colloquium (CS966/CS986) — 2021

Teaching Assistant, Samsung Electronics
- Samsung Research AI Expert Program — 2021 – 2024

Student Mentoring and Collaboration
- Chanhyuk Lee, M.S. Student @ KAIST — 2025 – present
- Jiyun Park, B.S. Student @ KAIST [P2] → M.S. Student @ KAIST — 2024 – present
- Kyungbin Min, B.S. Student @ KAIST [P1] — 2025
- Nayun Kim, B.S. Student @ KAIST [C11] → Intern @ EPFL LTS4 — 2024
- Youngmin Ryou, B.S. Student @ KAIST [C10] → on leave for mandatory military service — 2023 – 2024
- Nicole Shen, B.S. Student @ MIT → Intern @ MIT LIDS — 2024
- Semin Kim, M.S. Student @ KAIST [C8] → Ph.D. Student @ KAIST — 2023 – 2024
- Ayhan Suleymanzade, B.S. Student @ KAIST [C7, C10] → Intern @ TU Munich — 2023 – 2024
- Olga Zaghen, M.S. Student @ UniTrento [C10] → Ph.D. Student @ UvA Amsterdam — 2023
- Tien Dat Nguyen, B.S. Student @ KAIST [C5, C7, W1] → M.S. Student @ UWaterloo — 2021 – 2023
- Daniel Sungho Jung, B.S. Student @ Penn State → Ph.D. Student @ SNU — 2021
- Saeyoon Oh, B.S. Student @ KAIST [C2, C3] → Engineer @ FuriosaAI — 2021

Projects

Extending Language Models for Physical Data Understanding — 2024 – 2025
Korea National Research Foundation (NRF)

Image Inpainting with Visual Commonsense Reasoning — 2021 – 2023
Korea Ministry of Science and ICT

Cooperative Intelligence for Heterogeneous Robots — 2021 – 2023
Korea National Research Foundation (NRF)

References

Seunghoon Hong, Associate Professor at KAIST — seunghoon.hong@kaist.ac.kr
Kyunghyun Cho, Full Professor at New York University — kyunghyun.cho@nyu.edu
Moontae Lee, Head of Superintelligence Lab at LG AI Research — moontae.lee@lgresearch.ai
Siamak Ravanbakhsh, Associate Professor at McGill University — siamak.ravanbakhsh@mcgill.ca
İsmail İlkan Ceylan, Associate Member at University of Oxford — ismail.ceylan@cs.ox.ac.uk