Pure Transformers are Powerful Graph Learners

Jinwoo Kim[1*] **Tien Dat Nguyen**[1] **Seonwoo Min**[2] **Sungjun Cho**[2]
Moontae Lee[2,3] **Honglak Lee**[2†] **Seunghoon Hong**[1,2†]
[1]KAIST [2]LG AI Research [3]University of Illinois Chicago

Abstract

We show that standard Transformers without graph-specific modifications can lead to promising results in graph learning both in theory and practice. Given a graph, we simply treat all nodes and edges as independent tokens, augment them with token embeddings, and feed them to a Transformer. With an appropriate choice of token embeddings, we prove that this approach is theoretically at least as expressive as an invariant graph network (2-IGN) composed of equivariant linear layers, which is already more expressive than all message-passing Graph Neural Networks (GNN). When trained on a large-scale graph dataset (PCQM4Mv2), our method coined **Tokenized Graph Transformer (TokenGT)** achieves significantly better results compared to GNN baselines and competitive results compared to Transformer variants with sophisticated graph-specific inductive bias. Our implementation is available at `https://github.com/jw9730/tokengt`.

1 Introduction

In recent years, Transformer [66] has served as a versatile architecture in a broad class of machine learning problems, such as natural language processing [17, 7], computer vision [18], and reinforcement learning [9], 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 [31, 30, 55].

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 [48]. 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 [67, 49, 19], using global self-attention in conjunction with message-passing GNN [56, 41, 34], and injecting edge information into global self-attention via attention bias [70, 76, 29, 52]. 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 [31]. In addition, deviating from pure self-attention, these methods may inherit the issues of message-passing such as oversmoothing [38, 8, 50], and become incompatible with useful engineering techniques *e.g.*, linear attention [63] 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 [66, 18]. Perhaps surprisingly, we show that this simple approach yields a powerful graph learner both in theory and practice.

[*]Work done during an internship at LG AI Research.
[†]Equal corresponding authors.

Preprint. Under review.

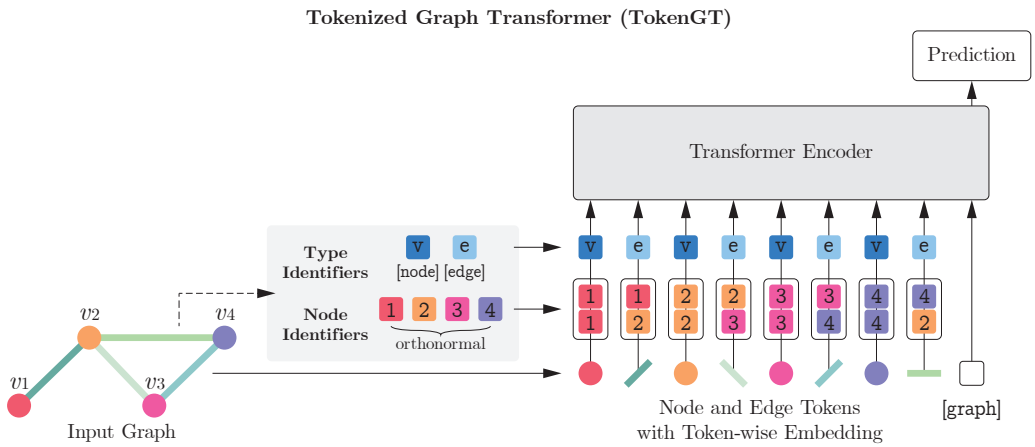

Figure 1: Overview of Tokenized Graph Transformer (TokenGT). We treat all nodes and edges of an input graph as independent tokens, augment them with orthonormal node identifiers and trainable type identifiers, and feed them to a standard Transformer encoder. For graph-level prediction, we follow the common practice [17, 18] of using an extra trainable `[graph]` token.

As a key theoretical 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 [45]. 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) [45, 34], which is already more expressive than all message-passing GNNs [21]. This also immediately grants the model with the expressive power at least as good as the 2-dimensional Weisfeiler-Lehman (WL) graph isomorphism test [44], which is often sufficient for real-world graph data [81]. 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 [27]. 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 [76, 29, 52]. Finally, we demonstrate that TokenGT can naturally utilize efficient approximations in Transformers in contrast to these variants, using kernel attention [11] that enables linear computation cost without much degradation in performance.

2 Tokenized Graph Transformer (TokenGT)

In this section, we present the Tokenized Graph Transformer (TokenGT), a pure Transformer architecture for graphs with token-wise embeddings composed of *node identifiers* and *type identifiers* (Figure 1). Our goal in this section is to provide a practical overview – for theoretical analysis of the architecture, we guide the readers to Section 3.

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 is inappropriate as graph connectivity is discarded. To thoroughly 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 theoretically 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. Their implementation details can be found in Appendix A.3.1.

- Orthogonal random features (ORFs), *e.g.*, rows of random orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ obtained with QR decomposition of random Gaussian matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ [77, 12].
- 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 [20].

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 5 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 5. 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 [20]. 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 [17] and ViT [18]. 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 [66], which is an alternating stack of multihead self-attention layers (MSA) and feedforward MLP layers. We provide further details in Appendix A.1.1.

Inductive Bias Similar to Transformers in language and vision [17, 18], Tokenized Graph Transformer 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 [21, 45, 44, 34]. 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 3 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) [21].

3 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, is provably *at least* as expressive as the second-order Invariant Graph Network (2-IGN [45]), which is built upon *all* possible permutation equivariant linear layers on a graph. This provides solid theoretical 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 (k-IGN [45]).

3.1 Preliminary: Permutation Symmetry and Invariant Graph Networks

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. Nevertheless, in Section 5 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)* [45, 44], 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 1. *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 the following:*

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

where each $L^{(t)}_{k \to k}$ is equivariant linear layer [45] 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 theoretical properties of k-IGN, including universal approximation [46] and alignment to k-Weisfeiler-Lehman (k-WL) graph isomorphism test [44, 10]. In particular, it is known that k-IGNs are theoretically at least as powerful as the k-WL test [44]. It is also known that 2-IGNs are already more expressive [45, 34] than all message-passing GNNs under the framework of Gilmer et al. (2017) [21].

The core building block of IGN is invariant and equivariant *linear* layers [45] with maximal expressiveness while respecting node permutation symmetry. The layers are defined as follows:

Definition 2. *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 $\mathbf{X} \in \mathbb{R}^{n^k\times d}$:*

$$L_{k\to l}(\mathbf{X})_{\mathbf{i}} = \sum_{\mu}\sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}\mathbf{X}_{\mathbf{j}} w_{\mu} + \sum_{\lambda} \mathbf{C}^{\lambda}_{\mathbf{i}} b_{\lambda}, \tag{2}$$

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.1.1. 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 Eq. (2) 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.

3.2 Can Self-Attention Approximate Equivariant Basis?

Now, we present an intuition that connects Transformer (Section 2) and equivariant linear layer (Definition 2). For that, we write out the multihead self-attention layer as follows:

$$\mathrm{MSA}(\mathbf{X})_i = \sum_{h=1}^{H}\sum_{j} \boldsymbol{\alpha}^h_{ij}\mathbf{X}_j w^V_h w^O_h \text{ where } \boldsymbol{\alpha}^h = \mathrm{softmax}\left(\frac{\mathbf{X}w^Q_h(\mathbf{X}w^K_h)^\top}{\sqrt{d_H}}\right), \tag{3}$$

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 Eq. (3) is analogous to the masked sum with basis tensor $\mathbf{B}^{\mu}$ in Eq. (2) 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 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 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 $w_1, w_2 \in \mathbb{R}^{d\times d}$, $b \in \mathbb{R}^d$:

$$L_{1\to 1}(\mathbf{X}) = \mathbf{I}\mathbf{X}w_1 + (\mathbf{1}\mathbf{1}^\top - \mathbf{I})\mathbf{X}w_2 + \mathbf{1}b^\top. \tag{4}$$

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), 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 Eq. (3) 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:

$$\boldsymbol{\alpha}^1 = \mathrm{softmax}(a\mathbf{I}) \to \mathbf{I} \text{ as } a \to \infty. \tag{5}$$

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:

$$\boldsymbol{\alpha}^2 = \mathrm{softmax}\left(-a\mathbf{I}\right) \to \frac{1}{n-1}(\mathbf{1}\mathbf{1}^\top - \mathbf{I}) \text{ as } a \to \infty. \tag{6}$$

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 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, Transformer layers can accurately approximate equivariant layers $L_{k\to k}$ via input-agnostic head-wise approximation of each equivariant basis.

3.3 Pure Transformers are Powerful Graph Learners

We now present our main theoretical results that extend the discussions in Section 3.2 to any order k. Note that $k = 2$ corresponds to TokenGT for graphs presented in Section 2. With $k > 2$, we naturally extend TokenGT to hypergraphs. All proofs can be found in Appendix A.1.

We first introduce generalized node and type identifiers (Section 2) 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 in Section 3.2. 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, 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 $\boldsymbol{\alpha}^h$ (Eq. (3)) 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 2) 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 1. *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 2) 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 1 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 2. *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 1) to arbitrary precision.*

This directly leads to the following corollary:

Corollary 1. *A Transformer on node and type identifiers in Theorem 2 is at least as expressive as k-IGN composed of order-k equivariant linear layers.*

Corollary 1 allows us to draw previous theoretical results on the expressiveness of k-IGN [44, 45, 34] and use them to lower-bound the provable expressiveness of a standard Transformer:

Corollary 2. *A Transformer on node and type identifiers in Theorem 2 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) [21].*

4 Related Work

We outline relevant work including equivariant neural networks, theory on expressive power of Transformers and their connection to modeling equivariance, and Transformers for graphs.

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 [13, 14, 71, 64, 47, 51, 58, 6, 34, 37]. Ravanbakhsh et al. (2017) [54] showed that any equivariant layer for discrete group actions is equivalent to a specific parameter sharing structure. Zaheer et al. (2017) [80] and Maron et al. (2019) [45] derived the parameter sharing for node permutation-symmetric data (sets and (hyper)graphs), which gives the maximally expressive equivariant linear layers and k-IGN in Section 3.1. 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 involving 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 [18, 31, 30, 7, 17, 9, 37]. The success of this approach is, to some degree, attributed to the high expressive power of Transformers that allows learning diverse functions suited for the data at hand [79, 37, 3, 4, 39]. Recent theory has shown that Transformers are expressive enough to even model certain equivariant functions [1, 15, 37]. Andreoli et al. (2019) [1] cast self-attention and convolution into a unified framework using basis tensors similar to ones in Section 3.1. Cordonnier et al. (2020) [15] advanced the idea and showed that Transformers with relative positional encodings can approximate any image convolution layers. Lee et al. (2019) [37] and Kim et al. (2021) [34] showed that Transformers can model equivariant linear layers for sets [80], which can be viewed as the first-order case of our theory (see Section 3.2). 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 in 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 [48]. To incorporate the connectivity of edges, early methods restricted self-attention to local neighborhoods (thus reducing to message-passing) [19, 49, 67] or used global self-attention with auxiliary message-passing modules [56, 41]. As message-passing suffers from limited expressive power [75] and oversmoothing [38, 8, 50], recent works often discard them and use global self-attention on nodes with heuristic modifications to process edges [76, 29, 52, 36, 40]. Ying et al. (2021) [76] proposed to inject edge encoding based on shortest paths through self-attention bias. Kreuzer et al. (2021) [36] proposed to incorporate edges into self-attention matrix via elementwise multiplication. On the contrary, we leave the self-attention unmodified and provide both nodes and edges with certain token-wise embeddings (Section 2) as its input. To incorporate graph structure into nodes, on the other hand, some approaches focus on developing graph positional encoding, *e.g.*, based on Laplacian eigenvectors [20, 40, 36]. While these can be directly incorporated into our work via auxiliary node identifiers for better performance, we leave this as future work. We further note that current graph Transformers that utilize Laplacian positional encoding rely on heuristic edge encoding [29, 36] while ours does not. Another closely related approach is the Higher-order Transformer [34] which generalizes k-IGN with masked self-attention. While it is highly complex to implement due to hard-coded head-wise equivariant masks, our method can be implemented effortlessly using any available implementation of standard Transformer. Furthermore, our method is more flexible as the model can choose to use different attention heads to focus on a specific equivariant operator (*e.g.*, local propagation) if needed. We further discuss the compatibility of linear attention and graph Transformers in Appendix A.2.

5 Experiments

We first conduct a synthetic experiment that directly confirms our key claims in Lemma 1 (Section 3). Then, we empirically explore the capability of Tokenized Graph Transformer (TokenGT) (Section 2) using the PCQM4Mv2 large-scale quantum chemistry regression benchmark [27]. We further present experiments on transductive node classification involving large graphs in Appendix A.4.3.

Table 1: 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.

		dense input		sparse input	
node id.	type id.	train L2 ↓	test L2 ↓	train L2 ↓	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

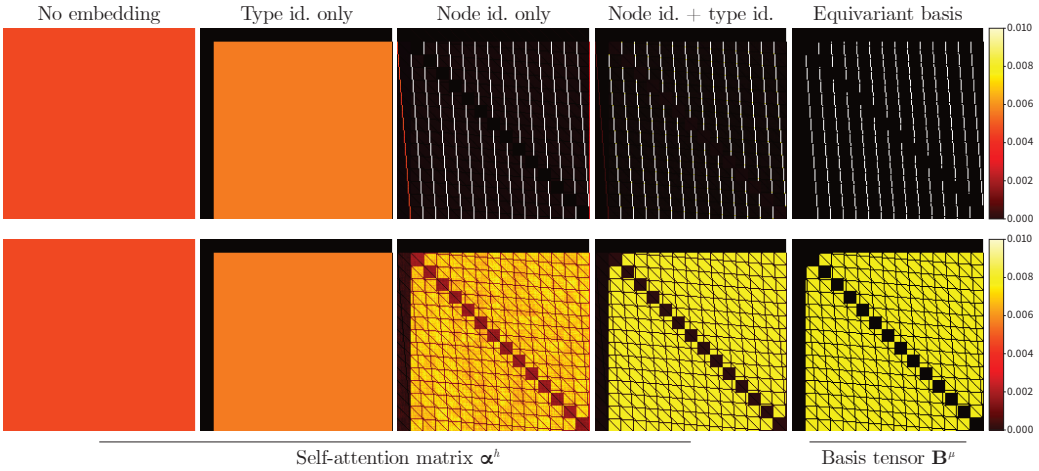

Figure 2: Self-attention maps learned under various node and type identifier configurations for two target equivariant basis tensors (out of 15). For better visualization, we clamp the entries by 0.01. Self-attention learns acute patterns coherent to equivariant basis when orthonormal node identifiers and type identifiers are both provided as input. More images can be found in Appendix A.4.1.

5.1 Approximating Second-Order Equivariant Basis

As in Theorem 1 and 2 (Section 3), 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 2). Specifically, Lemma 1 states that such capability depends on the ability of each self-attention head $\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^H$ (Eq. (3)) to accurately approximate each equivariant basis $\mathbf{B}^{\mu_1}, \ldots, \mathbf{B}^{\mu_{\mathrm{bell}(2k)}}$ (Definition 2) 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 (Eq. (3)) with $\mathrm{bell}(2+2) = 15$ 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 pairwise edges as input as in Section 3, whereas the sparse graph considers only the present m edges as in Section 2. Further details can be found in Appendix A.3.2.

We outline the results in Table 1. 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.

Table 2: Results on PCQM4Mv2 large-scale graph regression benchmark. We mainly report the Mean Absolute Error (MAE) on the validation set, and report MAE on the hidden test set if possible.

method	# parameters	valid MAE ↓	test-dev MAE ↓	asymptotics
Message-passing GNNs				
GCN [27]	2.0M	0.1379	0.1398	$\mathcal{O}(n+m)$
GIN [27]	3.8M	0.1195	0.1218	$\mathcal{O}(n+m)$
GAT	6.7M	0.1302	N/A	$\mathcal{O}(n+m)$
GCN-VN [27]	4.9M	0.1153	0.1152	$\mathcal{O}(n+m)$
GIN-VN [27]	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 [61]	48.3M	0.0864	N/A	$\mathcal{O}(n^2)$
EGT [29]	89.3M	0.0869	0.0872	$\mathcal{O}(n^2)$
GRPE [52]	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	$\mathcal{O}(n+m)$

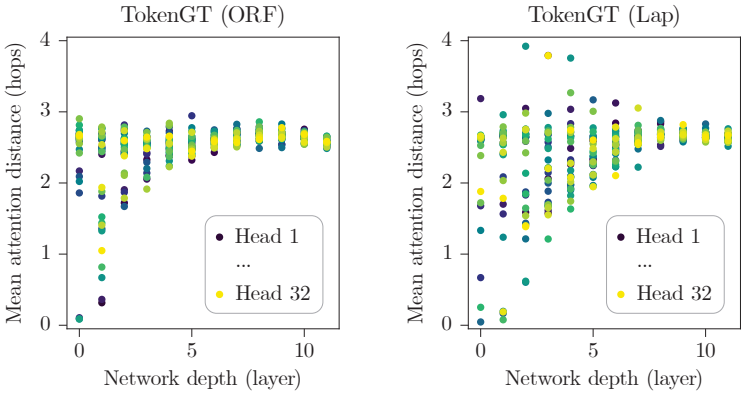

Figure 3: Attention distance by head and network depth. Each dot shows mean attention distance in hops across graphs of a head at a layer. The visualization is inspired by Dosovitskiy et al. (2020) [18]. More images can be found in Appendix A.4.2.

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 that the incidence information implicitly provided via node identifiers plays a key role in approximation.

In Figure 2, we provide a visualization of self-attention maps learned under various node and type identifier choices. Additional results can be found in Appendix A.4.1.

5.2 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 [66, 18], we explore the capability of TokenGT on the PCQM4Mv2 quantum chemistry regression dataset [27], 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 [76] 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 [11] 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. Further details are in Appendix A.3.3.

We provide the results in Table 2. 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 [20], we observe a performance boost to 0.0910 MAE, competitive to Transformers with sophisticated graph-specific modifications (*e.g.*, shortest path-based spatial encoding [76]). While such 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 [11], resulting in TokenGT (Lap) + Performer with the best performance among $\mathcal{O}(n+m)$ models (0.0935 MAE). Further discussion on the empirical performance of our model can be found in Appendix A.5.

While our theory in Section 3 *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 3). 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 [18], suggesting that hybrid architectures based on convolution to aid ViT [16, 78] 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 [36, 40].

6 Conclusion

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 Tokenized Graph Transformer (TokenGT) performs significantly better than all GNNs and is competitive with Transformer variants with strong graph-specific architectural components [76, 29, 52].

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 [7, 18], deepening [68, 72], hybrid architectures [16, 78], and self-supervision [17, 7, 24], 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 some interesting research directions that stem from our work. As our approach enables to view a graph as $(n+m)$ tokens, 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*.

Acknowledgement This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) (No. 2022-0-00926, 2022-0-00959, 2021-0-02068, and 2019-0-00075) and the National Research Foundation of Korea (NRF) (No. 2021R1C1C1012540) grants funded by the Korea government (MSIT).

References

[1] J.-M. Andreoli. Convolution, attention and structure embedding. *arXiv*, 2019. (Cited on 7)

[2] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *arXiv*, 1999. (Cited on 10)

[3] S. Bhattamishra, A. Patel, and N. Goyal. On the computational power of transformers and its implications in sequence modeling. In *CoNLL*, 2020. (Cited on 7)

[4] S. Bhojanapalli, C. Yun, A. S. Rawat, S. J. Reddi, and S. Kumar. Low-rank bottleneck in multi-head attention models. In *ICML*, 2020. (Cited on 7)

[5] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? In *ICLR*, 2022. (Cited on 11)

[6] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Process. Mag.*, 2017. (Cited on 7)

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. (Cited on 1, 7, 10)

[8] C. Cai and Y. Wang. A note on over-smoothing for graph neural networks. *arXiv*, 2020. (Cited on 1, 7)

[9] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, 2021. (Cited on 1, 7)

[10] Z. Chen, L. Chen, S. Villar, and J. Bruna. Can graph neural networks count substructures? In *NeurIPS*, 2020. (Cited on 4)

[11] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *ICLR*, 2021. (Cited on 2, 9, 10, 11)

[12] K. M. Choromanski, M. Rowland, and A. Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *NeurIPS*, 2017. (Cited on 3, 10)

[13] T. Cohen and M. Welling. Group equivariant convolutional networks. In *ICML*, 2016. (Cited on 7)

[14] T. S. Cohen and M. Welling. Steerable cnns. In *ICLR*, 2017. (Cited on 7)

[15] J. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020. (Cited on 7)

[16] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*, 2021. (Cited on 10)

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. (Cited on 1, 2, 3, 4, 7, 10)

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. (Cited on 1, 2, 3, 4, 7, 9, 10)

[19] V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv*, 2020. (Cited on 1, 7)

[20] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *arXiv*, 2020. (Cited on 3, 7, 10, 14)

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017. (Cited on 2, 4, 6)

[22] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov, and K. I. Sofeikov. Approximation with random bases: Pro et contra. *Inf. Sci.*, 2016. (Cited on 14)

[23] B. Hanin and M. Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv*, 2017. (Cited on 1, 7)

[24] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick. Masked autoencoders are scalable vision learners. *arXiv*, 2021. (Cited on 10)

[25] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv*, 2018. (Cited on 11)

[26] K. Hornik, M. B. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 1989. (Cited on 7)

[27] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. *arXiv*, 2021. (Cited on 2, 7, 9, 11)

[28] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. (Cited on 11)

[29] M. S. Hussain, M. J. Zaki, and D. Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv*, 2021. (Cited on 1, 2, 7, 9, 10)

[30] A. Jaegle, S. Borgeaud, J. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. J. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver IO: A general architecture for structured inputs & outputs. *arXiv*, 2021. (Cited on 1, 7, 9)

[31] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. (Cited on 1, 7, 9)

[32] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020. (Cited on 9)

[33] N. Keriven and G. Peyré. Universal invariant and equivariant graph neural networks. In *NeurIPS*, 2019. (Cited on 14)

[34] J. Kim, S. Oh, and S. Hong. Transformers generalize deepsets and can be extended to graphs and hypergraphs. In *NeurIPS*, 2021. (Cited on 1, 2, 4, 6, 7, 14)

[35] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. (Cited on 14)

[36] D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou. Rethinking graph transformers with spectral attention. *NeurIPS*, 2021. (Cited on 7, 10, 9, 14)

[37] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019. (Cited on 7, 9)

[38] Q. Li, Z. Han, and X. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018. (Cited on 1, 7)

[39] V. Likhosherstov, K. Choromanski, and A. Weller. On the expressive power of self-attention matrices. *arXiv*, 2021. (Cited on 7)

[40] D. Lim, J. Robinson, L. Zhao, T. E. Smidt, S. Sra, H. Maron, and S. Jegelka. Sign and basis invariant networks for spectral graph representation learning. *arXiv*, 2022. (Cited on 7, 10, 14)

[41] K. Lin, L. Wang, and Z. Liu. Mesh graphormer. In *ICCV*, 2021. (Cited on 1, 7)

[42] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. (Cited on 11)

[43] X. Ma, X. Kong, S. Wang, C. Zhou, J. May, H. Ma, and L. Zettlemoyer. Luna: Linear unified nested attention. *NeurIPS*, 2021. (Cited on 9)

[44] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman. Provably powerful graph networks. In *NeurIPS*, 2019. (Cited on 2, 4, 6, 1, 14)

[45] H. Maron, H. Ben-Hamu, N. Shamir, and Y. Lipman. Invariant and equivariant graph networks. In *ICLR*, 2019. (Cited on 2, 4, 5, 6, 7, 1, 14)

[46] H. Maron, E. Fetaya, N. Segol, and Y. Lipman. On the universality of invariant networks. In *ICML*, 2019. (Cited on 4, 14)

[47] H. Maron, O. Litany, G. Chechik, and E. Fetaya. On learning sets of symmetric elements. In *ICML*, 2020. (Cited on 7)

[48] E. Min, R. Chen, Y. Bian, T. Xu, K. Zhao, W. Huang, P. Zhao, J. Huang, S. Ananiadou, and Y. Rong. Transformer for graphs: An overview from architecture perspective. *arXiv*, 2022. (Cited on 1, 7)

[49] D. Q. Nguyen, T. D. Nguyen, and D. Phung. Universal graph transformer self-attention networks. In *WWW*, 2022. (Cited on 1, 7)

[50] K. Oono and T. Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2020. (Cited on 1, 7)

[51] H. Pan and R. Kondor. Permutation equivariant layers for higher order interactions. In *AISTATS*, 2022. (Cited on 7)

[52] W. Park, W. Chang, D. Lee, J. Kim, and S. won Hwang. Grpe: Relative positional encoding for graph transformer. *arXiv*, 2022. (Cited on 1, 2, 7, 9, 10)

[53] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong. Random feature attention. In *ICLR*, 2021. (Cited on 9)

[54] S. Ravanbakhsh, J. G. Schneider, and B. Póczos. Equivariance through parameter-sharing. In *ICML*, 2017. (Cited on 7)

[55] S. Reed, K. Żolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent. *arXiv*, 2022. (Cited on 1)

[56] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang. Self-supervised graph transformer on large-scale molecular data. In *NeurIPS*, 2020. (Cited on 1, 7)

[57] B. Rozemberczki, C. Allen, and R. Sarkar. Multi-scale attributed node embedding. *J. Complex Networks*, 2021. (Cited on 13)

[58] H. Serviansky, N. Segol, J. Shlomi, K. Cranmer, E. Gross, H. Maron, and Y. Lipman. Set2graph: Learning graphs from sets. In *NeurIPS*, 2020. (Cited on 7, 14)

[59] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann. Pitfalls of graph neural network evaluation. *arXiv*, 2018. (Cited on 13)

[60] Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li. Efficient attention: Attention with linear complexities. In *WACV*, 2021. (Cited on 9)

[61] Y. Shi, S. Zheng, G. Ke, Y. Shen, J. You, J. He, S. Luo, C. Liu, D. He, and T. Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arXiv*, 2022. (Cited on 9)

[62] A. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv*, 2021. (Cited on 11)

[63] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient transformers: A survey. *arXiv*, 2020. (Cited on 1)

[64] N. Thomas, T. E. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv*, 2018. (Cited on 7)

[65] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *CVPR*, 2015. (Cited on 10)

[66] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017. (Cited on 1, 3, 9)

[67] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018. (Cited on 1, 7, 11, 14)

[68] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. Deepnet: Scaling transformers to 1, 000 layers. *arXiv*, 2022. (Cited on 10)

[69] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv*, 2020. (Cited on 9)

[70] X. Wang, Z. Tu, L. Wang, and S. Shi. Self-attention with structural position representations. In *EMNLP-IJCNLP*, 2019. (Cited on 1)

[71] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018. (Cited on 7)

[72] N. Wies, Y. Levine, D. Jannai, and A. Shashua. Which transformer architecture fits my data? A vocabulary bottleneck in self-attention. In *ICML*, 2021. (Cited on 10)

[73] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. On layer normalization in the transformer architecture. In *ICML*, 2020. (Cited on 11)

[74] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *AAAI*, 2021. (Cited on 9)

[75] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *ICLR*, 2019. (Cited on 7, 14)

[76] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T. Liu. Do transformers really perform bad for graph representation? In *NeurIPS*, 2021. (Cited on 1, 2, 7, 9, 10, 11, 14)

[77] F. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar. Orthogonal random features. In *NeurIPS*, 2016. (Cited on 3, 10)

[78] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu. Incorporating convolution designs into visual transformers. In *ICCV*, 2021. (Cited on 10)

[79] C. Yun, S. Bhojanapalli, A. S. Rawat, S. J. Reddi, and S. Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *ICLR*, 2020. (Cited on 7, 1)

[80] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. J. Smola. Deep sets. In *NeurIPS*, 2017. (Cited on 7)

[81] M. Zopf. 1-wl expressiveness is (almost) all you need. *arXiv*, 2022. (Cited on 2)

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section 5.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3 and Section 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3.
   (b) Did you include complete proofs of all theoretical results? [Yes] We include them in the supplementary file.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include them in the supplementary file.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Section 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.2.
   (b) Did you mention the license of the assets? [Yes] See Section 5.2.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include them in the supplementary file.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Proofs

A.1.1 Extended Preliminary (Cont. from Section 3.1)

Before proceeding to the proofs, we first provide additional preliminary material that supplements Section 3.1. We begin by formally defining multihead self-attention and Transformer. Our definition is equivalent to Vaswani et al. (2017) [66], except we omit layer normalization for simplicity as in [79, 23, 34]. Specifically, a multihead self-attention layer MSA : $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is defined as:

$$\boldsymbol{\alpha}^h = \text{softmax}\left(\mathbf{X}w_h^Q(\mathbf{X}w_h^K)^\top/\sqrt{d_H}\right), \tag{7}$$

$$\text{MSA}(\mathbf{X})_i = \sum_{h=1}^{H}\sum_{j=1}^{n}\boldsymbol{\alpha}_{ij}^h\mathbf{X}_j w_h^V w_h^O, \tag{8}$$

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 [79] 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:

$$\mathbf{H} = \mathbf{X} + \text{MSA}(\mathbf{X}), \tag{9}$$

$$\mathcal{T}(\mathbf{X}) = \mathbf{H} + \text{MLP}(\mathbf{H}), \tag{10}$$

where 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 MLP : $\mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ is a tokenwise MLP with hidden dimension d_F.

We now provide the complete definition of invariant graph networks (IGNs) [45, 44] and maximally expressive equivariant linear layers [45] summarized in Section 3.1. We first recall Definition 1 and 2:

Definition 1. *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 the following:*

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

where each $L_{k\to k}^{(t)}$ is equivariant linear layer [45] 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 $\mathbb{R}$.

Definition 2. *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 $\mathbf{X} \in \mathbb{R}^{n^k\times d}$:*

$$L_{k\to l}(\mathbf{X})_{\mathbf{i}} = \sum_{\mu}\sum_{\mathbf{j}}\mathbf{B}_{\mathbf{i},\mathbf{j}}^{\mu}\mathbf{X}_{\mathbf{j}}w_\mu + \sum_{\lambda}\mathbf{C}_{\mathbf{i}}^{\lambda}b_\lambda, \tag{2}$$

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 now define *equivalence classes* and *basis tensors* mentioned briefly in Definition 2. The equivalence classes are defined upon a specific *equivalence relation* $\sim$ on the index space of higher-order tensors as follows:

Definition 3. *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 3 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 4. *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:*

$$\mathbf{B}_{\mathbf{i}}^{\gamma} = \begin{cases} 1 & \mathbf{i} \in \gamma \\ 0 & \textit{otherwise} \end{cases} \tag{11}$$

For a given l, it is known that there exist bell(l) order-l equivalence classes $\{\gamma_1, ..., \gamma_{\text{bell}(l)}\} = [n]^l/_\sim$ regardless of n [45]. This gives bell(l) order-l basis tensors $\mathbf{B}^{\gamma_1}, ..., \mathbf{B}^{\gamma_{\text{bell}(l)}}$ accordingly. Thus, an equivariant linear layer $L_{k\to l}$ in Definition 2 has bell$(l+k)$ weights and bell(l) biases.

Let us consider the first-order equivariant layer $L_{1\to 1}$ as an example. We have 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 4, 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 $\mathbf{X} \in \mathbb{R}^{n\times d}$,

$$L_{1\to 1}(\mathbf{X}) = \mathbf{I}\mathbf{X}w_1 + (\mathbf{1}\mathbf{1}^\top - \mathbf{I})\mathbf{X}w_2 + \mathbf{1}b^\top, \tag{12}$$

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 bell$(4) = 15$ weights and bell$(2) = 2$ biases.

A.1.2 Proof of Lemma 1 (Section 3.3)

To prove Lemma 1, we need to show that each basis tensor $\mathbf{B}^\mu$ (Eq. (11)) in weights of equivariant linear layers (Eq. (2)) can be approximated by the self-attention coefficient $\boldsymbol{\alpha}^h$ (Eq. (7)) to arbitrary precision up to normalization if its input is augmented by node and type identifiers (Section 3.3).

From Definition 4, 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 (Eq. (8)) 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 2, which comes from Lemma 1 of Kim et al. (2021) [34] (we repeat their proof here for completeness).

Lemma 2. *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 2 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 3. *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 $(\mathbf{i}^2, \mathbf{j}^2) \in \mu$*

Proof. ($\Rightarrow$) If $(\mathbf{i}, \mathbf{j}) \in \mu$, from Lemma 2 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 $\mathcal{G}$.

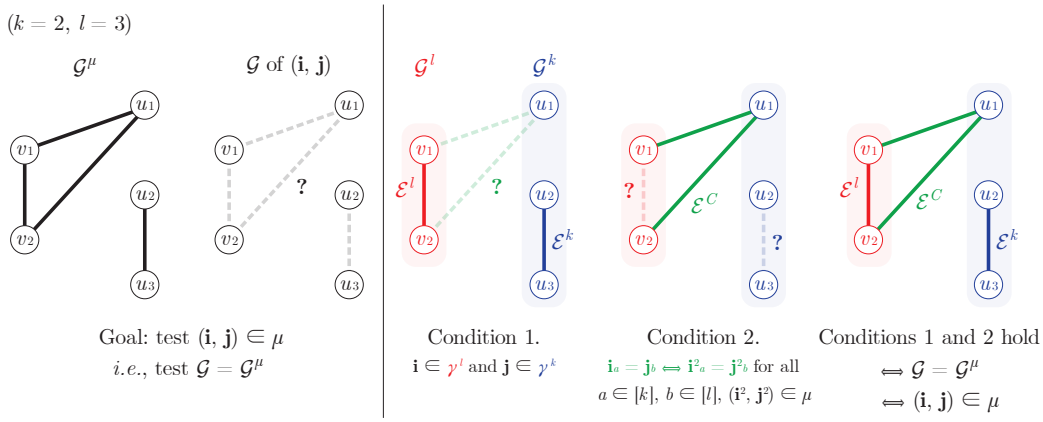

Figure 4: An exemplary illustration of testing $(\mathbf{i}, \mathbf{j}) \in \mu$ as a combination of simpler tests, based on equivalence classes μ, γ^l, and γ^k represented as graphs $\mathcal{G}^\mu$, $\mathcal{G}^l$, and $\mathcal{G}^k$, respectively.

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:

$$\mathcal{G}^l = (\mathcal{V}^l, \mathcal{E}^l) \text{ where } \mathcal{V}^l = \{v_1, ..., v_l\}, \tag{13}$$

$$\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k) \text{ where } \mathcal{V}^k = \{u_1, ..., u_k\}, \tag{14}$$

$$\mathcal{G}^\mu = (\mathcal{V}^\mu, \mathcal{E}^\mu) \text{ where } \mathcal{V}^\mu = \mathcal{V}^l \cup \mathcal{V}^k = \{v_1, ..., v_l, u_1, ..., u_k\}. \tag{15}$$

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 4, we provide an exemplary illustration of testing $(\mathbf{i}, \mathbf{j}) \in \mu$ following the above discussion. □

With Lemma 3, 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 5 and Property 1, we encode these tests into a single *scoring function* that can be later implemented by self-attention.

Definition 5. *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:*

$$\delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon) = \mathbb{1}_{\mathbf{i} \in \gamma^l} + (1 - \epsilon)\mathbb{1}_{\mathbf{i} \notin \gamma^l} + \mathbb{1}_{\mathbf{j} \in \gamma^k} + (1 - \epsilon)\mathbb{1}_{\mathbf{j} \notin \gamma^k} + \sum_{a \in [l]} \sum_{b \in [k]} \text{sgn}(a, b)\mathbb{1}_{\mathbf{i}_a = \mathbf{j}_b}, \tag{16}$$

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 $\text{sgn}(\cdot, \cdot)$ is defined as follows:

$$\text{sgn}(a, b) = \begin{cases} +1 & \mathbf{i}^2_a = \mathbf{j}^2_b \forall (\mathbf{i}^2, \mathbf{j}^2) \in \mu \\ -1 & \mathbf{i}^2_a \neq \mathbf{j}^2_b \forall (\mathbf{i}^2, \mathbf{j}^2) \in \mu \end{cases}. \tag{17}$$

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 1.

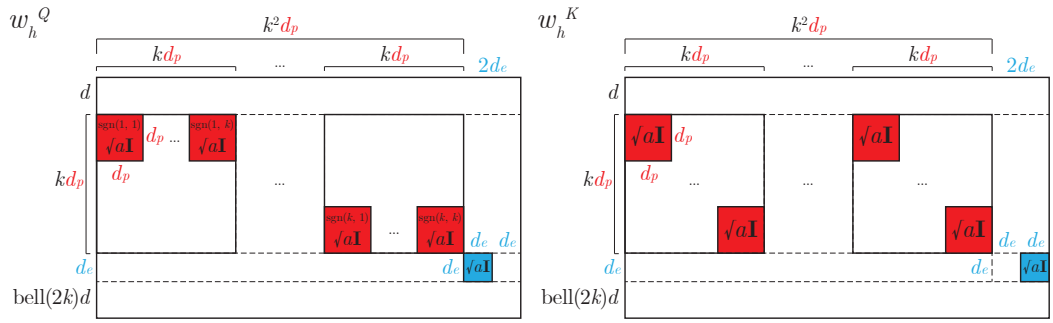

Figure 5: Query and key projection matrices w_h^Q, w_h^K (Eq. (18), Eq. (19)). Uncolored cells are zeros.

Property 1. *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 $\delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon)$ (Eq. (16)) outputs the maximum possible value.*

Proof. As shown in Lemma 3, $(\mathbf{i}, \mathbf{j}) \in \mu$ 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 Eq. (16), 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 $\text{sgn}(a, b) = 1$ and $\mathbb{1}_{\mathbf{i}_a = \mathbf{j}_b} = 0$ for $\text{sgn}(a, b) = -1$ for all $a \in [l], b \in [k]$. As a result, Eq. (16) 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 $\text{sgn}(a, b) = 1$ or $\mathbb{1}_{\mathbf{i}_a = \mathbf{j}_b} = 1$ for $\text{sgn}(a, b) = -1$ for some $a \in [l], b \in [k]$. *Any* of these violations decrements the output of Eq. (16) 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 $\boldsymbol{\alpha}^h$ (Eq. (3)) 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 2) to arbitrary precision up to normalization.*

Proof. Let us first recall the node and type identifiers (Section 3.3) 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 $\mathbf{E} \in \mathbb{R}^{\text{bell}(k) \times d_e}$ with bell(k) rows $\mathbf{E}^{\gamma_1}, ..., \mathbf{E}^{\gamma_{\text{bell}(k)}}$, 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 $d_\mathcal{T}$.

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 $\boldsymbol{\alpha}^h$ (Eq. (7)) to approximate each basis tensor $\mathbf{B}^{\mu_h}$ of $L_{k \to k}$ (Eq. (2)) 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 Eq. (7), 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 3). We set head dimension $d_H = k^2 d_p + 2d_e$ and set $w_h^Q, w_h^K, b_h^Q, b_h^K$ as follows:

$$(w_h^Q)_{ij} = \begin{cases} \text{sgn}(s,r)\sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \leq I + d_p & \text{for} \quad I = d + (s-1)d_p, \\ J < j \leq J + d_p & \text{for} \quad J = (s-1)kd_p + (r-1)d_p, \\ \text{for all } s, r \in [k] \end{cases} \\ \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \leq I + d_e & \text{for} \quad I = d + kd_p, \\ J < j \leq J + d_e & \text{for} \quad J = k^2 d_p, \end{cases} \\ 0 & \text{otherwise} \end{cases}, \tag{18}$$

$$(w_h^K)_{ij} = \begin{cases} \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \leq I + d_p & \text{for} \quad I = d + (r-1)d_p, \\ J < j \leq J + d_p & \text{for} \quad J = (s-1)kd_p + (r-1)d_p, \\ \text{for all } s, r \in [k] \end{cases} \\ \sqrt{a}\mathbf{I}_{i-I,j-J} & \begin{cases} I < i \leq I + d_e & \text{for} \quad I = d + kd_p, \\ J < j \leq J + d_e & \text{for} \quad J = k^2 d_p + d_e, \end{cases} \\ 0 & \text{otherwise} \end{cases}, \tag{19}$$

$$(b_h^Q)_j = \begin{cases} \sqrt{a}\mathbf{E}_{j-J}^{\gamma^K} & J < j \leq J + d_e \quad \text{for} \quad J = k^2 d_p \\ 0 & \text{otherwise} \end{cases}, \tag{20}$$

$$(b_h^K)_j = \begin{cases} \sqrt{a}\mathbf{E}_{j-J}^{\gamma^Q} & J < j \leq J + d_e \quad \text{for} \quad J = k^2 d_p + d_e \\ 0 & \text{otherwise} \end{cases}, \tag{21}$$

where $a > 0$ is a positive real, $\mathbf{I}$ is an identity matrix, and $\text{sgn}(\cdot, \cdot)$ is the sign function defined in Eq. (17) (Definition 5). In Figure 5 we provide an illustration of the query and key weights w_h^Q, w_h^K.

With the parameters, $\mathbf{i}$-th query and $\mathbf{j}$-th key entries are computed as follows:

$$\mathbf{X}_\mathbf{i}' w_h^Q + b_h^Q = \sqrt{a}[[\text{sgn}(1,1)\mathbf{P}_{i_1}, ..., \text{sgn}(1,k)\mathbf{P}_{i_1}], ..., [\text{sgn}(k,1)\mathbf{P}_{i_k}, ..., \text{sgn}(k,k)\mathbf{P}_{i_k}], \mathbf{E}^{\gamma^\mathbf{i}}, \mathbf{E}^{\gamma^K}], \tag{22}$$

$$\mathbf{X}_\mathbf{j}' w_h^K + b_h^K = \sqrt{a}[\overbrace{[\mathbf{P}_{j_1}, ..., \mathbf{P}_{j_k}], ..., [\mathbf{P}_{j_1}, ..., \mathbf{P}_{j_k}]}^{k \text{ repeats}}, \mathbf{E}^{\gamma^Q}, \mathbf{E}^{\gamma^\mathbf{j}}]. \tag{23}$$

Then, scaled pairwise dot product of query and key is given as follows:

$$\frac{(\mathbf{X}_\mathbf{i}' w_h^Q + b_h^Q)^\top (\mathbf{X}_\mathbf{j}' w_h^K + b_h^K)}{\sqrt{d_H}} = \frac{a}{\sqrt{d_H}} \left((\mathbf{E}^{\gamma^\mathbf{i}})^\top \mathbf{E}^{\gamma^Q} + (\mathbf{E}^{\gamma^\mathbf{j}})^\top \mathbf{E}^{\gamma^K} + \sum_{a \in [k]} \sum_{b \in [k]} \text{sgn}(a,b) \mathbf{P}_{i_a}^\top \mathbf{P}_{j_b} \right). \tag{24}$$

We refer to the scaled dot product in Eq. (24) 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 6). 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, Eq. (24), and Eq. (16), we have the following:

$$\tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h = \frac{a}{\sqrt{d_H}} \delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon) \ \text{ if } (\mathbf{i}, \mathbf{j}) \in \mu, \tag{25}$$

$$\tilde{\boldsymbol{\alpha}}_{\mathbf{i},\mathbf{j}}^h \leq \frac{a}{\sqrt{d_H}} \delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon) \ \text{ otherwise}, \tag{26}$$

where $\epsilon = 1 - \cos(2\pi/\text{bell}(k))$ and $\delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon)$ is the scoring function in Eq. (16) (Definition 5).

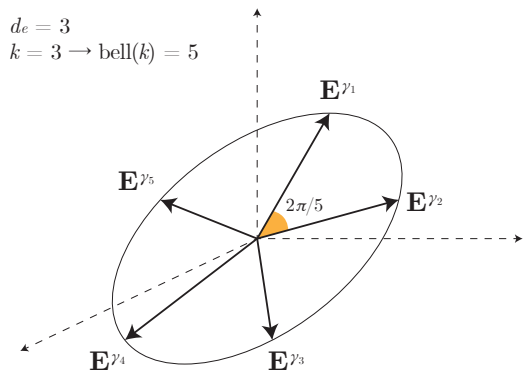

Figure 6: $k = 3$ case example of $\text{bell}(k) = 5$ type identifiers embedded in $d_e = 3$ dimensional space.

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 1 and Eq. (25), all keys $\mathbf{j}$ that give $(\mathbf{i}, \mathbf{j}) \in \mu$ hold the same maximum value $\tilde{\boldsymbol{\alpha}}^h_{\mathbf{i},\mathbf{j}} = \frac{a}{\sqrt{d_H}}\delta(\mathbf{i}, \mathbf{j}; \mu, \epsilon)$, 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 $a \rightarrow \infty$ by scaling up the query and key projection parameters. This pushes softmax arbitrarily close to the hardmax operator, leaving only the maximal entries leading to the following:

$$\boldsymbol{\alpha}^h_{\mathbf{i},\mathbf{j}} = \frac{\exp(\tilde{\boldsymbol{\alpha}}^h_{\mathbf{i},\mathbf{j}})}{\sum_{\mathbf{j}} \exp(\tilde{\boldsymbol{\alpha}}^h_{\mathbf{i},\mathbf{j}})} \rightarrow \frac{\mathbb{1}_{(\mathbf{i},\mathbf{j} \in \mu)}}{\sum_{\mathbf{j}} \mathbb{1}_{(\mathbf{i},\mathbf{j} \in \mu)}} = \frac{\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}} \text{ as } a \rightarrow \infty. \tag{27}$$

Thus, as shown in Eq. (27), the attention coefficient $\boldsymbol{\alpha}^h$ can arbitrarily accurately approximate the normalized basis tensor $\mathbf{B}^{\mu}$ for given equivalence class μ. □

A.1.3 Proof of Theorem 1 (Section 3.3)

Theorem 1. *For all $\mathbf{X} \in \mathbb{R}^{n^k \times d}$ and their augmentation $\mathbf{X}^{in}$, a Transformer layer with* $\text{bell}(2k)$ *self-attention heads that operates on $\mathbf{X}^{in} w^{in}$ can approximate an order-k equivariant linear layer $L_{k \rightarrow k}(\mathbf{X})$ (Definition 2) to arbitrary precision.*

Proof. We continue from the proof of Lemma 1 and assume that each attention matrix $\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^{\text{bell}(2k)}$ in Eq. (7) 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 Eq. (8) we use $d_v = d$ and set $w^V_h \in \mathbb{R}^{d_\mathcal{T} \times d}$ to $w^V_h = [\mathbf{I}; \mathbf{0}]$, 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, $\mathbf{X}'_{\mathbf{i}} w^V_h = \mathbf{X}_{\mathbf{i}}$.

Then, we set output projections $w^O_h \in \mathbb{R}^{d \times d_\mathcal{T}}$ as follows:

$$(w^O_h)_{ij} = \begin{cases} (w_{\mu_h})_{i,j-J} & J < j \leq J + d \text{ for } J = (d + kd_p + d_e) + (h-1)d \\ 0 & \text{otherwise} \end{cases}, \tag{28}$$

where $w_{\mu_1}, ..., w_{\mu_{\text{bell}(2k)}} \in \mathbb{R}^{d \times d}$ are weight matrices of the given equivariant linear layer $L_{k \rightarrow k}$ in Eq. (2) (Definition 2), 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:

$$\mathbf{X}'_{\mathbf{i}} w^V_h w^O_h = \mathbf{X}_{\mathbf{i}} w^O_h = [\mathbf{0}, \mathbf{0}_L, \mathbf{X}_{\mathbf{i}} w_{\mu_h}, \mathbf{0}_R], \tag{29}$$

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.

[3] If such key index $\mathbf{j}$ does not exist, corresponding basis tensor entries are $\mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} = 0 \forall \mathbf{j}$, and approximation target cannot be defined as normalizing denominator $\sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}}$ 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.

[4] we handle the case $\sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} = 0$ later separately as mentioned in footnote 3.

Based on the results, we compute the MSA with skip connection $\mathbf{H} = \mathbf{X}' + \text{MSA}(\mathbf{X}')$ (Eq. (9)):

$$\begin{aligned}
\mathbf{H}_{\mathbf{i}} &= \mathbf{X}'_{\mathbf{i}} + \text{MSA}(\mathbf{X}')_{\mathbf{i}} && (30)\\
&= \left[\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_1\right] \\
&+ \left[\mathbf{0}_2, \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_1}, ..., \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_{\text{bell}(2k)}}\right] && (31)\\
&= \left[\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_1}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_1}, ..., \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_{\text{bell}(2k)}}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_{\text{bell}(2k)}}\right], && (32)
\end{aligned}$$

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 (Eq. (10)) 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 approximation [23, 26]:

$$f(\mathbf{H}_{\mathbf{i}})_j = \begin{cases} -\mathbf{H}_{\mathbf{i},j} + \sum_{h \in [\text{bell}(2k)]} g(\mathbf{H}_{\mathbf{i}})_h \mathbf{H}_{\mathbf{i},j+J} + b(\mathbf{H}_{\mathbf{i}})_j & j \le d \\ 0 & d < j \le (d + kd_p + d_e) \\ -\mathbf{H}_{\mathbf{i},j} & j > (d + kd_p + d_e) \end{cases}, \quad (33)$$

$$g(\mathbf{H}_{\mathbf{i}})_h = \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}, \quad (34)$$

$$b(\mathbf{H}_{\mathbf{i}})_j = (b_{\gamma^{\mathbf{i}}})_j = (\sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma)_j, \quad (35)$$

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 $\mathbf{C}^{\gamma_1}, ..., \mathbf{C}^{\gamma_{\text{bell}(k)}}$ (Eq. (2)).

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 $g(\mathbf{H}_{\mathbf{i}})$ and $b(\mathbf{H}_{\mathbf{i}})$ 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 $\mathbf{H}_{\mathbf{i}}$, 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 $\mathcal{T}(\mathbf{X}') = \mathbf{H} + \text{MLP}(\mathbf{H})$ (Eq. (10)), which is the output of Transformer layer $\mathcal{T}$:

$$\begin{aligned}
\mathcal{T}(\mathbf{X}')_{\mathbf{i}} &= \mathbf{H}_{\mathbf{i}} + \text{MLP}(\mathbf{H})_{\mathbf{i}} && (36)\\
&= \mathbf{H}_{\mathbf{i}} + f(\mathbf{H}_{\mathbf{i}}) && (37)\\
&= \left[\mathbf{X}_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{S}^1_{\mathbf{i}}, ..., \mathbf{S}^{\text{bell}(2k)}_{\mathbf{i}}\right] \\
&+ \left[-\mathbf{X}_{\mathbf{i}} + \sum_{h \in [\text{bell}(2k)]} \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} \mathbf{X}_{\mathbf{j}} w_{\mu_h} + \sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma, \mathbf{0}_5, -\mathbf{S}^1_{\mathbf{i}}, ..., -\mathbf{S}^{\text{bell}(2k)}_{\mathbf{i}}\right], && (38)\\
&= \left[\sum_{h \in [\text{bell}(2k)]} \sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}} \mathbf{X}_{\mathbf{j}} w_{\mu_h} + \sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right], && (39)
\end{aligned}$$

where we write $\mathbf{S}^h_{\mathbf{i}} = \sum_{\mathbf{j}} \frac{\mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}}{\sum_{\mathbf{j}} \mathbf{B}^{\mu_h}_{\mathbf{i},\mathbf{j}}} \mathbf{X}_{\mathbf{j}} w_{\mu_h}$ 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 Eq. (39), note that the Transformer layer $\mathcal{T}(\mathbf{X}')_{\mathbf{i}}$ only updates the first d channels of $\mathbf{X}'_{\mathbf{i}}$ from $\mathbf{X}_{\mathbf{i}}$ to $\sum_{\mu} \sum_{\mathbf{j}} \mathbf{B}^{\mu}_{\mathbf{i},\mathbf{j}} \mathbf{X}_{\mathbf{j}} w_\mu + \sum_{\gamma} \mathbf{C}^{\gamma}_{\mathbf{i}} b_\gamma$. 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. □

[5]Note that the $g(\mathbf{H}_{\mathbf{i}})_h$ 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 footnote 3 and footnote 4.

A.1.4 Proof of Theorem 2 (Section 3.3)

Theorem 2. *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 1) to arbitrary precision.*

Proof. We continue from the proof of Theorem 1, and assume that each Transformer layer $\mathcal{T}$ can approximate a given $L_{k\to k}$ by only updating the first d channels.

Then, based on Theorem 1 we assume the following for each $t < T$:

$$\mathcal{T}^{(t)}(\mathbf{X}')_{\mathbf{i}} = \left[\sigma(L^{(t)}_{k\to k}(\mathbf{X}))_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right] \tag{40}$$

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 1 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 Eq. (33). 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:

$$\mathcal{T}^{(T-1)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}')_{\mathbf{i}} = \left[\sigma \circ L^{(T-1)}_{k\to k} \circ \sigma \circ ... \circ \sigma \circ L^{(1)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, \mathbf{P}_{i_1}, ..., \mathbf{P}_{i_k}, \mathbf{E}^{\gamma^{\mathbf{i}}}, \mathbf{0}_6\right]. \tag{41}$$

For the last layer $\mathcal{T}^{(T)}$, we follow the procedure in the proof of Theorem 1 to approximate $L^{(T)}_{k\to k}$, but slightly tweak Eq. (33) 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':

$$f'(\mathbf{H}_{\mathbf{i}})_j = \begin{cases} -\mathbf{H}_{\mathbf{i},j} & j \le D \\ -\mathbf{H}_{\mathbf{i},j} + \mathbf{C}^{\gamma_a}_{\mathbf{i}} \mathbf{F}_{\mathbf{i},j-(D+(a-1)d)} & D+(a-1)d < j \le D + ad \text{ for all } a \in [\text{bell}(k)] \\ -\mathbf{H}_{\mathbf{i},j} & D + \text{bell}(k)d < j \end{cases}, \tag{42}$$

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 Eq. (33). Recall that $\mathbf{C}^{\gamma_a}_{\mathbf{i}} = 1$ if and only if $\mathbf{i} \in \gamma_a$. Therefore, with Eq. (42), 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 $\mathcal{T}^{(T)}(\mathbf{X}') = \mathbf{H} + \text{MLP}(\mathbf{H})$ (Eq. (10)) is computed as:

$$\begin{aligned} \mathcal{T}^{(T)}(\mathbf{X}')_{\mathbf{i}} &= \mathbf{H}_{\mathbf{i}} + \text{MLP}(\mathbf{H})_{\mathbf{i}} && (43) \\ &= \mathbf{H}_{\mathbf{i}} + f'(\mathbf{H}_{\mathbf{i}}) && (44) \\ &= \left[\mathbf{0}_7, \mathbf{C}^{\gamma_1}_{\mathbf{i}} L^{(T)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, ..., \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} L^{(T)}_{k\to k}(\mathbf{X})_{\mathbf{i}}, \mathbf{0}_8\right], && (45) \end{aligned}$$

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 $\mathcal{T}^{(T)}$ (Eq. (45)) on top of $\mathcal{T}^{(T-1)} \circ ... \circ \mathcal{T}^{(1)}$ (Eq. (41)), we obtain the following:

$$\mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}')_{\mathbf{i}} = \left[\mathbf{0}_7, \mathbf{C}^{\gamma_1}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, ..., \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, \mathbf{0}_8\right]. \tag{46}$$

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:

$$\text{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') = \left[\mathbf{0}_7, \sum_{\mathbf{i}} \mathbf{C}^{\gamma_1}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, ..., \sum_{\mathbf{i}} \mathbf{C}^{\gamma_{\text{bell}}(k)}_{\mathbf{i}} \mathbf{Y}_{\mathbf{i}}, \mathbf{0}_8\right]. \tag{47}$$

Now, we let the final MLP approximate the following function $f'' : \mathbb{R}^{d_\mathcal{T}} \to \mathbb{R}^d$:

$$f''(\mathbf{X}) = \text{MLP}_k\left(\sum_{a\in[\text{bell}(k)]} \mathbf{X}^a w_{\mu_a} + b_f\right) \text{ where } \mathbf{X}^a_j = \mathbf{X}_{D+(a-1)d+j} \text{ for } j \in [d], \tag{48}$$

where $w_{\mu_1}, ..., w_{\mu_{\text{bell}(k)}} \in \mathbb{R}^{d\times d}$ 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 Eq. (42). By plugging in the sum-pooled representation in Eq. (47), we finally obtain the following:

$$\begin{aligned}
\text{MLP} \circ \text{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') &= f'' \circ \text{sumpool} \circ \mathcal{T}^{(T)} \circ ... \circ \mathcal{T}^{(1)}(\mathbf{X}') && (49)\\
&= \text{MLP}_k \left(\sum_{a\in[\text{bell}(k)]} \sum_{\mathbf{i}} \mathbf{C}_{\mathbf{i}}^{\gamma_a} \mathbf{Y}_{\mathbf{i}} w_{\mu_a} + b_f \right) && (50)\\
&= \text{MLP}_k \circ L_{k\to 0}(\mathbf{Y}) && (51)\\
&= \text{MLP}_k \circ L_{k\to 0} \circ L_{k\to k}^{(T)} \circ \sigma \circ ... \circ \sigma \circ L_{k\to k}^{(1)}(\mathbf{X}) && (52)\\
&= F_k(\mathbf{X}), && (53)
\end{aligned}$$

where the last equality comes from Definition 1.

Taken together, we arrive at the conclusion that $\text{MLP}\circ\text{sumpool}\circ\mathcal{T}^{(T)}\circ...\circ\mathcal{T}^{(1)}(\mathbf{X}')$ can approximate $F_k(\mathbf{X})$ to arbitrary precision. □

A.2 Extended Discussion on Linear Attention for Graph Transformers (Section 4)

We provide additional discussion on related work, specifically on why Graphormer [76] based on fully-connected self-attention on nodes is not compatible to many linear attention methods to reduce the memory complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. A range of prior graph Transformers including EGT [29], GRPE [52], and SAN [36] can be analyzed analogously. Let us first remind self-attention with query, key, value $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n\times d}$ and self-attention matrix $\boldsymbol{\alpha} \in \mathbb{R}^{n\times n}$:

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V})_i = \sum_j \boldsymbol{\alpha}_{ij} \mathbf{V}_j \text{ where } \boldsymbol{\alpha}_{ij} = \frac{\exp(\mathbf{Q}_i^\top \mathbf{K}_j/\sqrt{d})}{\sum_k \exp(\mathbf{Q}_i^\top \mathbf{K}_k/\sqrt{d})}. \tag{54}$$

For graphs, as self-attention on nodes alone cannot recognize the edge connectivity, Graphormer incorporates the structural information of an input graph $\mathcal{G}$ into the self-attention matrix $\boldsymbol{\alpha}^{\mathcal{G}} \in \mathbb{R}^{n\times n}$ via attention bias matrix $\mathbf{b}^{\mathcal{G}} \in \mathbb{R}^{n\times n}$ (referred to as the edge/spatial encoding) as the following:

$$\boldsymbol{\alpha}_{ij}^{\mathcal{G}} = \frac{\exp(\mathbf{Q}_i^\top \mathbf{K}_j/\sqrt{d} + \mathbf{b}_{ij}^{\mathcal{G}})}{\sum_k \exp(\mathbf{Q}_i^\top \mathbf{K}_k/\sqrt{d} + \mathbf{b}_{ij}^{\mathcal{G}})}. \tag{55}$$

Unfortunately, this modification immediately precludes the adaptation of many efficient attention techniques developed for pure self-attention. As representative examples, we take Performer [11], Linear Transformer [32], Efficient Transformer [60], and Random Feature Attention [53]. The methods are based on kernelization of the $\text{Att}(\cdot)$ operator as the following:

$$\text{Att}_\phi(\mathbf{Q}, \mathbf{K}, \mathbf{V})_i = \sum_j \frac{\phi(\mathbf{Q}_i)^\top \phi(\mathbf{K}_j)}{\sum_k \phi(\mathbf{Q}_i)^\top \phi(\mathbf{K}_k)} \mathbf{V}_j = \frac{\phi(\mathbf{Q}_i)^\top \left(\sum_j \phi(\mathbf{K}_j)\mathbf{V}_j^\top\right)}{\phi(\mathbf{Q}_i)^\top \left(\sum_k \phi(\mathbf{K}_k)\right)}. \tag{56}$$

The above factorization of the exponential into a pairwise dot product, in turn, eliminates the need to explicitly compute the attention matrix for computing $\text{Att}_\phi(\cdot)$ and consequently reduces both time and memory cost to $\mathcal{O}(n)$. Yet, in Eq. (55), since the bias $\mathbf{b}_{ij}^{\mathcal{G}}$ is added to the dot product *before* the exponential, it is required that the full attention matrix $\boldsymbol{\alpha}^{\mathcal{G}} \in \mathbb{R}^{n\times n}$ is always explicitly computed. Thus, Graphormer and related variations are unable to utilize the method and are bound to $\mathcal{O}(n^2)$.

While above discussion mainly regards kernelization, a wide range of other efficient Transformers, including Set Transformer [37], LUNA [43], Linformer [69], Nyströmformer [74], Perceiver [31], and Perceiver-IO [30] are not applicable to Graphormer due to similar reasons.

A.3 Experimental Details (Section 5)

We provide detailed information on the datasets and models used in our experiments in Section 5. Dataset statistics can be found in Table 3.

Table 3: Statistics of the datasets.

(a) Statistics of Barabási-Albert random graph dataset.

Dataset	Barabási-Albert
Size	1280
Average # node	14.9
Average # edge	47.8

(b) Statistics of PCQM4Mv2 dataset.

Dataset	PCQM4Mv2
Size	3.7M
Average # node	14.1
Average # edge	14.6

A.3.1 Implementation Details of Node and Type Identifiers

In most of our experiments on graph data ($k = 2$), we fix the Transformer encoder configuration and experiment with choices of node identifiers $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ and type identifiers $\mathbf{E} \in \mathbb{R}^{2 \times d_e}$ (Section 2).

For type identifiers $\mathbf{E}$, we set d_e equal to the hidden dimension d of the main encoder $d_e = d$ and initialize and train them jointly with the model.

For orthonormal node identifiers $\mathbf{P}$, we use normalized orthogonal random features (ORFs) or Laplacian eigenvectors obtained as follows:

- For orthogonal random features (ORFs), we use rows of random orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ obtained with QR decomposition of random Gaussian matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ [77, 12].
- For Laplacian eigenvectors, we perform 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 [20].

The model expects d_p-dimensional node identifiers $\mathbf{P} \in \mathbb{R}^{n \times d_p}$, while ORF and Laplacian eigenvectors are n-dimensional. To resolve this, if $n < d_p$, we zero-pad the channels. If $n > d_p$, for ORF we randomly sample d_p channels and discard the rest, and for Laplacian eigenvectors we use d_p eigenvectors with the smallest eigenvalues following common practice [20, 40, 36].

As the Laplacian eigenvectors are defined up to the factor ± 1 after normalized to unit length [20], we randomly flip their signs during training. For PCQM4Mv2 (Section 5.2), we apply random dropout on eigenvectors during training, similar to 2D channel dropout in ConvNets [65]. In our experiments with PCQM4Mv2, we find that both sign flip and eigenvector dropout work as effective regularizers and improves performance on validation data.

A.3.2 Second-Order Equivariant Basis Approximation (Section 5.1)

Dataset For the equivariant basis approximation experiment, we use a synthetic dataset containing Barabási-Albert (BA) random graphs [2]. With $\mathcal{U}$ denoting discrete uniform distribution, each graph is generated by first sampling the number of nodes $n \sim \mathcal{U}(10, 20)$ and the number for preferential attachment $k \sim \mathcal{U}(2, 3)$, then iteratively adding n nodes by linking each new node to k random previous nodes. We do not utilize node or edge attributes and only use edge connectivity. We generate 1152 graphs for training and 128 for testing. Further dataset statistics is provided in Table 3a.

Architecture Each model tested in Table 1 is a single multihead self-attention layer (Eq. (9)) with hidden dimension $d = 1024$, heads $H = \text{bell}(2 + 2) = 15$, and head dimension $d_H = 128$. As for the node identifier dimension, we use $d_p = 24$ for ORF and $d_p = 20$ for Laplacian eigenvectors.

Experimental Setup We experiment with sparse or dense input graph representations. For sparse input, we embed each graph with n nodes and m edges into $\mathbf{X}^{in} \in \mathbb{R}^{(n+m) \times (2d_p + d_e)}$. For dense input, we use all n^2 pairwise edges and obtain $\mathbf{X}^{in} \in \mathbb{R}^{n^2 \times (2d_p + d_e)}$, so that sparse edge connectivity is only used for obtaining Laplacian node identifiers.

For both sparse and dense inputs, we follow the standard procedure in Section 2 to use node and type identifiers to obtain $\mathbf{X}^{in} \in \mathbb{R}^{N \times (2d_p + d_e)}$ where $N = (n + m)$ or n^2, and project it to dimension d with trainable projection w^{in}. We also utilize a special token `[null]` with trainable embedding $\mathbf{X}_{\texttt{[null]}} \in \mathbb{R}^d$ (we shortly explain its use) to obtain the final input $[\mathbf{X}_{\texttt{[null]}}; \mathbf{X}^{in} w^{in}] \in \mathbb{R}^{(1+N) \times d}$.

For an input $[\mathbf{X}_{\texttt{[null]}}; \mathbf{X}^{in}w^{in}]$, the goal is to supervise each of the $H = 15$ self-attention heads with attention matrices $\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^{15}$ to explicitly approximate row-normalized version of each equivariant basis $\mathbf{B}^{\mu_1}, ..., \mathbf{B}^{\mu_{15}} \in \mathbb{R}^{N \times N}$ (Eq. (2)) on the $N = (n + m)$ or n^2 input tokens (except `[null]`). An issue in supervision is that for rows of $\mathbf{B}^{\mu}$ that only contains zeros, normalization is not defined. To sidestep this, for such rows we simply supervise to attend to the special `[null]` token only. For all other rows of $\mathbf{B}^{\mu}$ that contains nonzero entry, we supervise the model to ignore `[null]` token.

Training and Evaluation We train and evaluate all models with L2 loss between attention matrix $\boldsymbol{\alpha}^h$ and normalized basis tensor $\mathbf{B}^{\mu_h}$ (involving `[null]` token) averaged over heads $h = 1, ..., 15$. We train all models with AdamW optimizer [42] on 4 RTX 3090 GPUs each with 24GB. For sparse inputs we use batch size 512, and for dense inputs we use batch size 256 due to increased memory cost. We train all models for 3k steps (which takes about $\sim$1.5 hours) and apply linear learning rate warmup for 1k steps up to 1e-4 followed by linear decay to 0. For all models, we use dropout rate of 0.1 on the input $[\mathbf{X}_{\texttt{[null]}}; \mathbf{X}^{in}w^{in}]$ to prevent overfitting.

A.3.3 Large-Scale Graph Learning (Section 5.2)

Dataset For large-scale learning, we use the PCQM4Mv2 quantum chemistry regression dataset from the OGB-LSC benchmark [27] that contains 3.7M molecular graphs. Along with graph structure, we utilize both node and edge features *e.g.*, atom and bond types following our standard procedure in Section 2. Dataset statistics is provided in Table 3b.

Architecture All our models in Table 2 (under *Pure Transformers*) have the same encoder configuration following Graphormer [76], with 12 layers, hidden dimension $d = 768$, heads $H = 32$, and head dimension $d_H = 24$. We adopt PreLN [73] that places layer normalization before MSA layer (Eq. (9)), MLP layer (Eq. (10)), and the final output projection after the last encoder layer. We implement MLP (Eq. (10)) as a stack of two linear layers with GeLU nonlinearity [25] in between. As for node identifier dimension, we use $d_p = 64$ for ORF and $d_p = 16$ for Laplacian eigenvectors.

As an additional GNN baseline, we run Graph Attention Network (GATv2) [67, 5] under several configurations. For GAT and GAT-VN in Table 2, we use 5-layer GATv2 with hidden dimension 600 and a single attention head, having 6.7M parameters in total. For GAT-VN (large), we use a 10-layer GATv2 with hidden dimension 1200 and a single attention head, having 55.2M parameters in total. For GAT-VN and GAT-VN (large), we use virtual node that helps modeling global interaction [27].

Training and Evaluation We mainly report and compare the Mean Absolute Error (MAE) on the validation data, and report MAE on the hidden test data if possible. We train all models with L1 loss using AdamW optimizer [42] with gradient clipping at global norm 5.0. We use batch size 1024 and train the models on 8 RTX 3090 GPUs with 24GB for $\sim$3 days. We train our models for 1M iterations, and apply linear lr warmup for 60k iterations up to 2e-4 followed by linear decay to 0. For our models in Table 2 except TokenGT (Lap) + Performer, we use the following regularizers:

- Attention and MLP dropout rate 0.1
- Weight decay 0.1
- Stochastic depth [28, 62] with linearly increasing layer drop rate, reaching 0.1 at last layer
- Eigenvector dropout rate 0.2 for TokenGT (Lap) (see Appendix A.3.1)

For TokenGT (Lap) + Performer in Table 2, we load a trained model checkpoint of TokenGT (Lap), change its self-attention to FAVOR+ kernelized attention of Performer [11] that can provably accurately approximate softmax attention, and fine-tune it with AdamW optimizer for 0.1M training steps with 1k warmup iterations and cosine learning rate decay. With batch size 1024 on 8 RTX 3090 GPUs, fine-tuning takes $\sim$ 12 hours. We do not use stochastic depth and eigenvector dropout for fine-tuning. For GAT baselines in Table 2, we use batch size 256 and train the models for 100 epochs with initial learning rate 0.001 decayed with a factor of 0.25 every 30 epochs.

A.4 Additional Experimental Results (Section 5)

We report additional experimental results and discussions that could not be included in the main text due to space restriction.

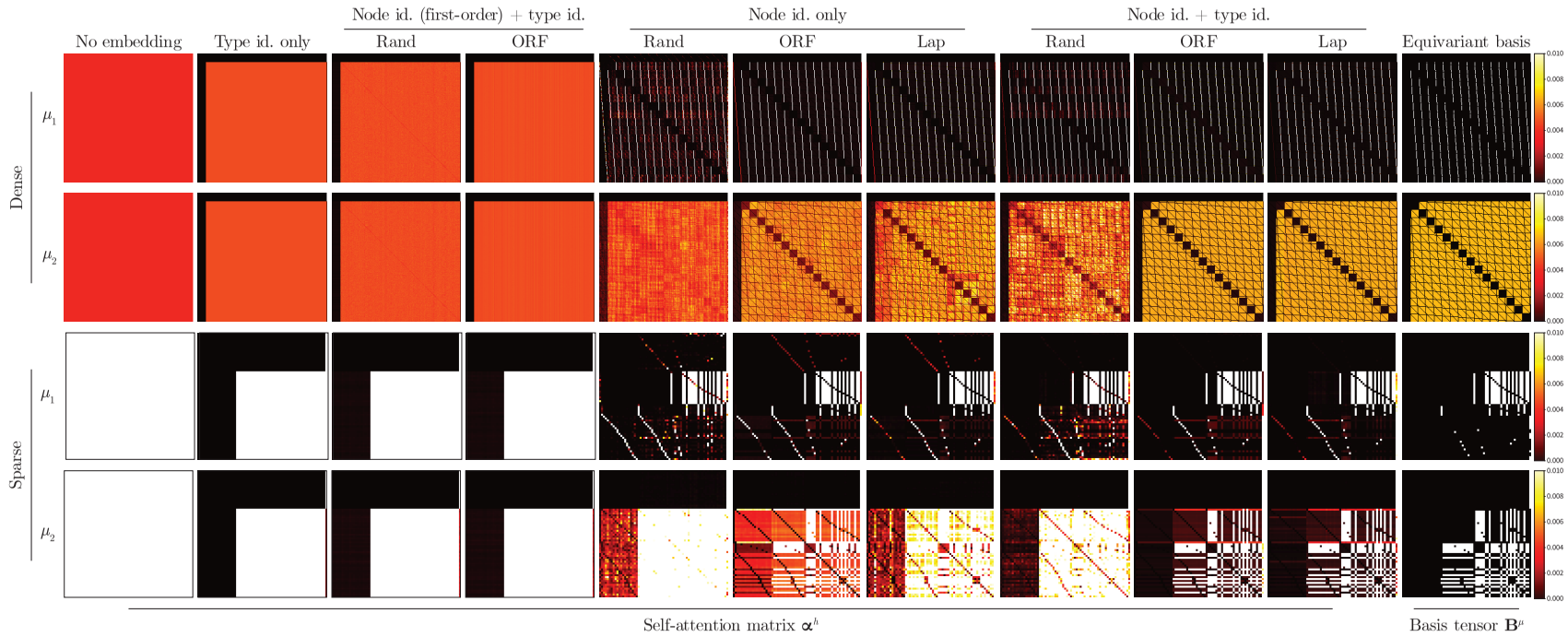

Figure 7: Self-attention maps learned under various node and type identifier configurations for two target equivariant basis tensors (out of 15), for both dense and sparse inputs. For better visualization, we clamp the entries by 0.01. Self-attention learns acute pattern coherent to equivariant basis when orthonormal node identifiers and type identifiers are provided both as input.

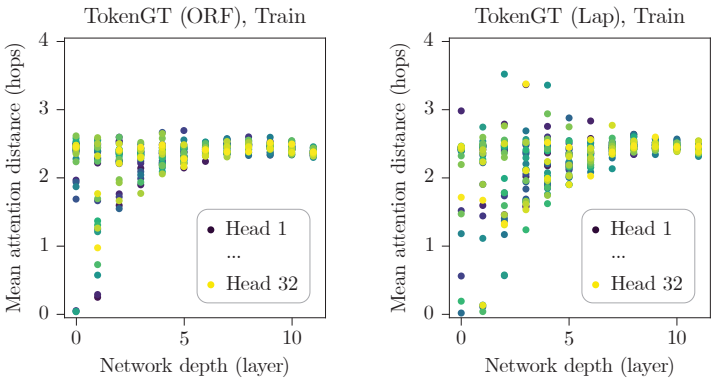

Figure 8: Attention distance by head and network depth, measured for entire PCQM4Mv2 training set. Each dot shows mean attention distance in hops across graphs of a head at a layer.

Table 4: Statistics of the transductive node classification datasets.

Dataset	CS	Physics	Photo	Computers	Chameleon	Crocodile
# nodes	18,333	34,493	7,650	13,752	2,277	11,631
# edges	81,894	247,962	119,081	245,861	36,101	180,020
# classes	15	5	8	10	6	6

Table 5: Results of transductive node classification experiment. OOM denotes out-of-memory error on a 24GB RTX 3080 GPU. We report aggregated test accuracy at best validation accuracy over 7 randomized runs.

	CS	Physics	Photo	Computers	Chameleon	Crocodile
GCN	0.895 ± 0.004	0.932 ± 0.004	0.926 ± 0.008	0.873 ± 0.004	0.593 ± 0.01	0.660 ± 0.01
GAT	0.893 ± 0.005	0.937 ± 0.01	0.947 ± 0.006	$\mathbf{0.914 \pm 0.002}$	0.632 ± 0.011	0.692 ± 0.017
GIN	0.895 ± 0.005	0.886 ± 0.046	0.886 ± 0.017	0.362 ± 0.051	0.479 ± 0.027	0.515 ± 0.041
Graphormer	0.791 ± 0.015	OOM	0.894 ± 0.004	0.814 ± 0.013	0.457 ± 0.011	0.489 ± 0.014
TokenGT (Near-ORF) + Performer	0.882 ± 0.007	0.931 ± 0.009	0.872 ± 0.011	0.82 ± 0.019	0.568 ± 0.019	0.583 ± 0.024
TokenGT (Lap) + Performer	0.902 ± 0.004	0.941 ± 0.007	0.919 ± 0.009	0.86 ± 0.012	0.637 ± 0.032	0.638 ± 0.025
TokenGT (Lap) + Performer + SEB	$\mathbf{0.903 \pm 0.004}$	$\mathbf{0.950 \pm 0.003}$	$\mathbf{0.949 \pm 0.007}$	0.912 ± 0.006	$\mathbf{0.653 \pm 0.029}$	$\mathbf{0.718 \pm 0.012}$

A.4.1 Second-Order Equivariant Basis Approximation (Section 5.1)

In addition to the Figure 2 in the main text that shows learned self-attention maps for dense input, in Figure 7, we provide an extended visualization of self-attention maps for both dense and sparse inputs. Consistent to Lemma 1 and Table 1, self-attention achieves accurate approximation of equivariant basis only when both the orthonormal node identifiers (ORF or Lap) and type identifiers are given.

A.4.2 Large-Scale Graph Learning (Section 5.2)

In addition to the Figure 3 in the main text that shows attention distance measured for the PCQM4Mv2 validation data, in Figure 8, we provide an extended figure of attention distance measured for the entire training set that contains $\sim$3M graphs. Overall we find similar trends as analyzed in Section 5.2.

A.4.3 Transductive Node Classification on Large Graphs (Section 5)

While our main experiment in Section 5.2 focuses on graph-level predictions, TokenGT can in principle be applied to a more broad class of node-level or edge-level graph understanding tasks by putting prediction head on appropriate output tokens. To demonstrate this, we conduct additional experiments on a range of transductive node classification datasets. In contrast to PCQM4Mv2, they involve large graphs with up to tens of thousands of nodes, posing a challenge to methods with $\mathcal{O}(n^2)$ memory complexity.

Dataset We use transductive node classification datasets including co-authorship (CS, Physics) [59], co-purchase (Photo, Computers) [59], and Wikipedia page networks (Chameleon, Crocodile) [57], which involve large-scale graphs. We randomly split the dataset into train, validation, and test sets by randomly reserving 30 random nodes per class for validation and test respectively, and use the rest for training. Dataset statistics is provided in Table 4.

Architecture We experiment with simple variants of TokenGT equipped with Performer kernel attention. Considering large n, an immediate challenge for TokenGT is dealing with the orthonormality assumption on the node identifiers (Lemma 1), as the maximal number of orthonormal node identifiers is upper bounded by its dimension d_p. In this case, it is reasonable to introduce *near-orthonormal* vectors as node identifiers, as it is theoretically guaranteed that we can draw an exponential number $\mathcal{O}(e^{\Omega(d_p)})$ of d_p-dimensional near-orthonormal vectors [22]. For *TokenGT (Near-ORF)*, we used $d_p = 64$-dimensional random node identifiers where each entry is sampled from $\{-1/d_p, +1/d_p\}$ under equal probability [22]. For *TokenGT (Lap)*, we used a subset of the Laplacian eigenvectors as node identifiers, more specifically $d_p/2$ eigenvectors with lowest eigenvalues and $d_p/2$ eigenvectors with highest eigenvalues, and choose d_p in the range of 64 to 100 based on validation performance.

While *Near-ORF* and *Lap* can theoretically serve as an efficient low-rank approximation for orthonormal node identifiers, their approximation can affect the quality of modeled equivariant basis. In particular, equivariant basis (μ) represented as **sparse** basis tensor ($\mathbf{B}^{\mu}$; Definition 4) are expected to be affected more, as they require most entries to be zero. To remedy this, we take a simple approach of residually adding one of such sparse equivariant operators $\mathbf{X}_{ii} \mapsto \mathbf{X}_{ii} + \sum_{j \neq i} \mathbf{X}_{ij}$ explicitly after each Transformer layer. We denote this variant as *TokenGT (Lap) + Performer + SEB*, where SEB abbreviates sparse equivariant basis. This fix is minimal, easy to implement, and highly efficient as it only requires a single `torch.coalesce()` call, and also empirically effective.

Training and Evaluation We report and compare classification accuracy on the test nodes at best validation accuracy aggregated over 7 randomized runs. We train all models with binary cross-entropy loss using Adam optimizer. We compare our models against strong GNN baselines including GCN [35] (message-passing that works well on large graphs), GAT [67] (message-passing based on attention), GIN [75] (message-passing with 2-WL expressiveness guarantee similar to ours), and Graphormer [76] (graph transformer with shortest path-based edge and spatial encoding).

Results The results are in Table 5. Graphormer [76] suffers out-of-memory in the Physics dataset mainly due to the spatial encoding that requires $\mathcal{O}(n^2)$ memory. By constraining the model capacity appropriately, we were able to run Graphormer on other datasets. However, we observe a low performance, presumably due to the memory cost that prevents depth and head scaling. As the spatial encoding is incorporated into the model via attention bias, the model strictly requires $\mathcal{O}(n^2)$ memory and cannot be easily made more efficient. On the other hand, TokenGT variants are able to utilize Performer attention with $\mathcal{O}(m + n)$ cost, which allows using larger models to achieve the best performance in all but one dataset (Computers, where the performance is on par with the best model).

A.5 Extended Discussion on Performance on PCQM4Mv2 (Section 5.2)

As in the Table 2 in the main text, TokenGT currently shows a slightly lower performance compared to the Graphormer and its successors in the PCQM4Mv2 benchmark. We conjecture this is partly because we intentionally keep its components simple to faithfully adhere to the equivariance theory. We discuss some engineering approaches that may enhance the performance of TokenGT *at the cost of differentiating from the theory*. We consider engineering TokenGT to match or outperform sophisticated graph Transformers as a promising and important next research direction.

Node Identifiers Our best performing TokenGT (Lap) currently uses Laplacian eigenvectors [20] as the node identifiers, which has been criticized for issues such as loss of structural information [36] and sign ambiguity [40]. Thus, one could try to relax the theoretical requirement for orthonormality of node identifiers and incorporate more powerful node positional encodings [36, 40] as node identifiers, which could potentially yield better performance in practice.

(Hyper)Edge Tokens TokenGT currently treats an undirected input edge (u, v) as if both directions (u, v) and (v, u) are present, leading to a pair of edge tokens $[\mathbf{X}_{(u,v)}, \mathbf{P}_u, \mathbf{P}_v]$ and $[\mathbf{X}_{(v,u)}, \mathbf{P}_v, \mathbf{P}_u]$. Similarly, an undirected order-k input hyperedge $(v_1, ..., v_k)$ of an higher-order hypergraph is parsed to all possible orderings of node identifiers. While this is a common characteristic of tensor-based permutation equivariant neural networks [45, 46, 33, 58, 44, 34], they can lead to memory overhead and redundancy since multiple tokens represent an identical undirected edge. To avoid this, one can use a single token for each undirected (hyper)edge and pool the node identifiers as $\sum_{i=1}^{k} \rho(\mathbf{P}_{v_i})$. Combined with powerful node identifiers, this approach could potentially enhance the model performance.