Pure Transformers are Powerful Graph Learners

Anonymous Author(s)
Affiliation
Address
email

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 **Soft Graph Transformer (SGT)** achieves significantly better results compared to GNN baselines and competitive results compared to Transformer variants with sophisticated graph-specific inductive bias.

1 Introduction

In recent years, Transformer [35] has been served as a versatile architecture in a broad class of machine learning problems, such as natural language processing [8, 1], computer vision [9], reinforcement learning [3], to name a few. It is because the fully-attentional structure of the Transformer is general and powerful enough to take, process, and relate inputs and outputs in 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 [17, 16, 31].

In graph learning, inspired by the breakthroughs, multiple works tried combining self-attention into graph neural network (GNN) architecture where message passing was previously dominant [27]. 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 [36, 28, 10], using global self-attention in conjunction with message-passing GNN [32, 23, 18], and injecting edge information into global self-attention via attention bias [38, 40, 15, 30]. Yet, despite descent 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 [17]. In addition, deviating from pure self-attention, these methods may inherit the issues of message passing such as oversmoothing [21, 2, 29], or become incompatible with useful engineering techniques such as linear attention [34] developed for standard self-attention.

Instead, we explore the opposite direction of *applying a standard Transformer directly for graphs*. For this, we simply treat all nodes and edges as independent tokens, augment it with appropriate token-wise embeddings, and feed the tokens as an 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 [35, 9]. Perhaps surprisingly, we show that this simple approach yields a powerful graph learner in terms of both theory and experiment.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

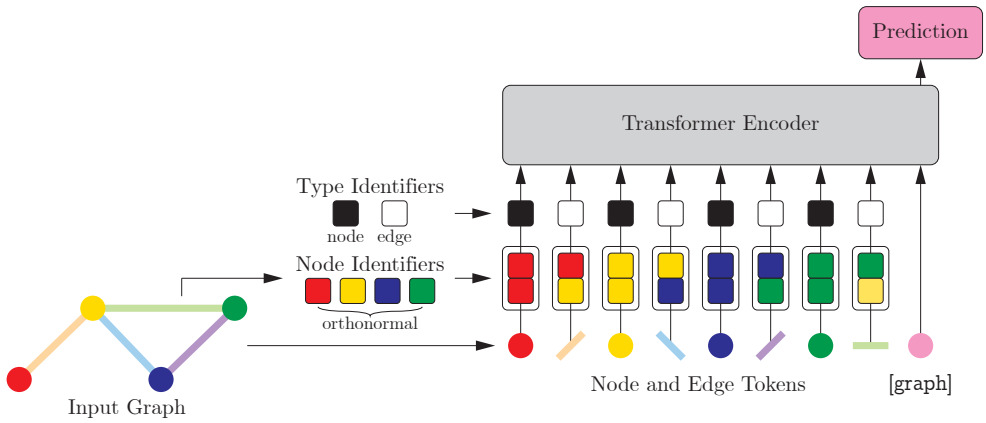

Figure 1: Overview of Soft Graph Transformer (SGT). We treat all nodes and edges of an input graph as independent tokens, augment them with orthonormal node identifiers and trainable type identifiers, and feed them to a standard Transformer encoder. To perform graph-level prediction, we follow the common practice [8, 9] of using an extra trainable token `[graph]`.

As a key theoretical result, we prove that with appropriate token-wise embeddings, self-attention over node and edge tokens can approximate *any* permutation equivariant linear operator on a graph [25]. 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 strong theoretical guarantee that, with the embeddings and enough attention heads, a Transformer is at least as expressive as second-order invariant graph network (2-IGN) [25, 18], which is already more expressive than all message-passing GNNs [12]. This also immediately grants the model with the expressive power at least as good as 2-dimensional Weisfeiler-Lehman (WL) graph isomorphism test [24], which is often sufficient for real-world graph data [43]. We further extend our theoretical result to *hypergraphs* with order-k hyperedges, and show 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 Soft Graph Transformer (SGT), mainly on PCQM4Mv2 large-scale quantum chemical property prediction benchmark involving 3M molecular graphs [14]. Despite the absence of graph-specific architectural modification, SGT obtains significantly better performance than all GNN baselines, showing that the benefits of Transformer architecture combined with large scale training overtakes the benefit of hard inductive bias of GNNs. Furthermore, SGT achieves competitive performance compared to Transformer variants with strong graph-specific modifications [40, 15, 30]. Finally, we show that SGT can naturally utilize efficient approximations in Transformers in contrast to these variants and demonstrate the results with kernelized attention [5] that enables the linear-time complexity without much degradation in performance.

2 Soft Graph Transformer (SGT)

In this section, we present the Soft Graph Transformer (SGT), a standard 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 $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an input graph with nodes $\mathcal{V} = \{v_1, ..., v_n\}$ and edges $\mathcal{E} = \{e_1, ..., e_m\} \in \mathcal{V}^2$ 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 an input to the Transformer, but it is inappropriate as the graph connectivity structure is discarded. To thoroughly represent graph structure, we augment the input with token-wise embedding constructed with *orthonormal node identifiers* for representing connectivity structure of the tokens and *type identifiers* that encode whether a token is node or edge. Despite seemingly simple, in Section 3 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 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 a 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. It allows the Transformer to identify and exploit the connectivity structure of a graph, for instance by putting more weights to incident pairs when the local operation is important.

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

- Orthogonal random features (ORFs), *e.g.*, rows of random orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$ obtained with QR decomposition of random Gaussian matrix $\mathbf{G} \in \mathbb{R}^{n \times n}$ [41, 6].
- 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 [11].

Among the two methods, node identifiers generated as ORFs do not encode any information about the graph structure as they are completely random. This means the Transformer that operates on the ORF-based node identifiers needs to compile and to recognize graph structure only from the incidence information provided by the node identifiers. Although this is challenging, perhaps surprisingly, we empirically show in Section 4 that Transformers are strong enough to learn meaningful structural representations out of ORF-based node identifiers and outperform GNNs on large scale task.

In contrast to ORFs, Laplacian eigenvectors provide a kind of graph positional embeddings (graph PEs) that describes the distance between nodes on a graph. Due to the positional information, it yields better performance compared to ORFs in our experiments in Section 4. One interesting aspect of Laplacian eigenvectors is that they are a natural generalization of sinusoidal positional embeddings of NLP Transformers to graphs as the eigenvectors of 1D chain graph are sine and cosine functions [11]. Thus, by choosing Laplacian eigenvectors as node identifiers, our approach can be interpreted as a direct extension of NLP Transformer for inputs involving relational structures.

Type Identifiers The second component of token-wise embedding is 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 [8] and ViT [9]. We utilize the state 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 $\mathbf{Z}^{(0)} = [\mathbf{X}_{\texttt{[graph]}}; \mathbf{X}^{in} w^{in}] \in \mathbb{R}^{(1+n+m) \times d}$ serve as the input to the main encoder. As an encoder, we adopt the standard Transformer [35], an alternating stack of multihead self-attention layers (MSA) and feedforward MLP layers. We provide the details in the supplementary file.

Inductive Bias Similar to Transformers in language and vision [8, 9], Soft Graph Transformer treats input nodes and edges as independent tokens and simply applies self-attention on 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 [19, 12, 25, 24, 18]. SGT deviates from the paradigm – such information is provided entirely as a part of input using token-wise embeddings, and the model has to learn from data how to interpret and utilize the information. Although such weak inductive bias might raise questions on the expressiveness of the model, our theoretical analysis in Section 3 shows that SGT is a powerful graph learner thanks to the token-wise embeddings and expressive power of self-attention. For example, we show that SGT is more expressive than all message-passing GNNs under the framework of Gilmer et al. (2017) [12].

3 Theoretical Analysis

We now present theoretical results. Our key result is that SGT, a standard Transformer with node and type identifiers presented in Section 2, is provably *at least* as expressive as second-order Invariant Graph Network (2-IGN [25]) which is built upon *all* possible permutation equivariant linear layers on a graph. This provides strong theoretical guarantees for SGT, such as being at least as powerful as 2-WL graph isomorphism test and being 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 formulation of order-k SGT that are at least as expressive as order-k IGN (k-IGN [25]).

3.1 Preliminary: Permutation Symmetry and Invariant Graph Networks

Representing and Processing Sets and (Hyper)Graphs Given a set of n nodes, their features are represented as $\mathbf{X} \in \mathbb{R}^{n \times d}$ where $\mathbf{X}_i \in \mathbb{R}^d$ is feature of i-th node. As the nodes are unordered, the set should be treated invariant to their renumbering. Let S_n the symmetric group or the group of permutations π on $[n] = \{1, ..., n\}$. By $\pi \cdot \mathbf{X}$ we denote the matrix obtained by permuting rows of $\mathbf{X}$ according to π, *i.e.*, $(\pi \cdot \mathbf{X})_i = \mathbf{X}_{\pi^{-1}(i)}$. Here, $\mathbf{X}$ and $\pi \cdot \mathbf{X}$ represent the same 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 arbitrary node renumbering by $\pi \in S_n$, 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 same (hyper)graph for all π. Due to such symmetry, when building a function $F(\mathbf{X}) \approx T$ for tensor $\mathbf{X}$ and target T, a suitable way is making them *invariant* $F(\pi \cdot \mathbf{X}) = F(\mathbf{X})$ when target is a vector or *equivariant* $F(\pi \cdot \mathbf{X}) = \pi \cdot F(\mathbf{X})$ when 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 with, but slightly different from the sparse representation of a graph with edge set $\mathcal{E}$ used in Section 2. Nevertheless, in Section 4 we empirically verify that our key theoretical findings work equally well for both dense and sparse graphs.

Invariant Graph Network We mainly develop our theoretical analysis upon *Invariant Graph Networks (IGNs)* [25, 24], a family of expressive graph networks defined on tensor representation of graphs derived from their permutation symmetry. 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 [25] 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 [26] and alignment to k-Weisfeiler-Lehman (k-WL) graph isomorphism test [24, 4]. In particular, it is known that k-IGNs are theoretically at least as powerful as k-WL test [24]. It is also known that 2-IGNs are already more expressive [25, 18] than all message-passing GNNs under the framework of Gilmer et al. (2017) [12].

The core building block of IGN is invariant and equivariant *linear* layers 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 the supplementary file. For now, it is sufficient to know that the basis tensors are binary tensors that form 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 weighted sum of values with self-attention matrix $\boldsymbol{\alpha}^h$ in Eq. (3) is analogous to 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 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 see that this can be possible, but only if we provide appropriate auxiliary information for input. Let us consider first-order layer $L_{1\to 1}$ as an example. 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 $\mathbf{X}^{in} = [\mathbf{X}, \mathbf{P}] \in \mathbb{R}^{n\times(d+d_p)}$. Now, 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 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 query and key projection weights, the softmax becomes arbitrarily close to 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 input-agnostic approximation of $\boldsymbol{\alpha}^1$ to $\mathbf{I}$. Notably, we can achieve 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 i-th query to 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 accurate approximation.

Overall, we see that simple auxiliary input $\mathbf{P}$ suffices for two attention heads to accurately approximate equivariant basis of $L_{1\to 1}$. Thus our question now becomes 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 SGT for graphs presented in Section 2. With $k > 2$, we naturally extend SGT to hypergraphs. All proofs can be found in the supplementary file.

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 node identifier $\mathbf{P} \in \mathbb{R}^{n \times d_p}$ as an orthonormal matrix with n rows, and 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 provide some examples. For $k = 1$ (sets), each i-th entry is augmented as $[\mathbf{X}_i, \mathbf{P}_i, \mathbf{E}^{\gamma_1}]$, consistent to 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 to SGT in Section 2 that 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 a hidden dimension $d_\mathcal{T}$ of a Transformer. We now show self-attention on $\mathbf{X}^{in} w^{in}$ can perform accurate approximation of 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 an 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 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 an 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 expressiveness of k-IGN [24, 25, 18] 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) [12].*

Table 1: Second-order equivariant basis approximation. We report L2 error averaged over heads. For Random/ORF (first-order), we sample random embeddings independently for each token.

node id.	type id.	dense input train L2 ↓	dense input test L2 ↓	sparse input train L2 ↓	sparse input test L2 ↓
×	×	48.53	52.77	29.39	33.37
×	◯	32.79	39.15	15.60	19.53
Random (first-order)	◯	32.65	36.67	15.60	15.58
ORF (first-order)	◯	32.68	38.90	15.60	15.58
Random	×	5.913	5.486	8.137	7.947
ORF	×	5.509	5.081	7.148	6.943
Laplacian eigenvector	×	5.421	5.065	0.267	0.076
Random	◯	0.366	0.238	1.055	0.888
ORF	◯	0.080	0.009	0.127	**0.013**
Laplacian eigenvector	◯	**0.053**	**0.005**	**0.097**	0.015

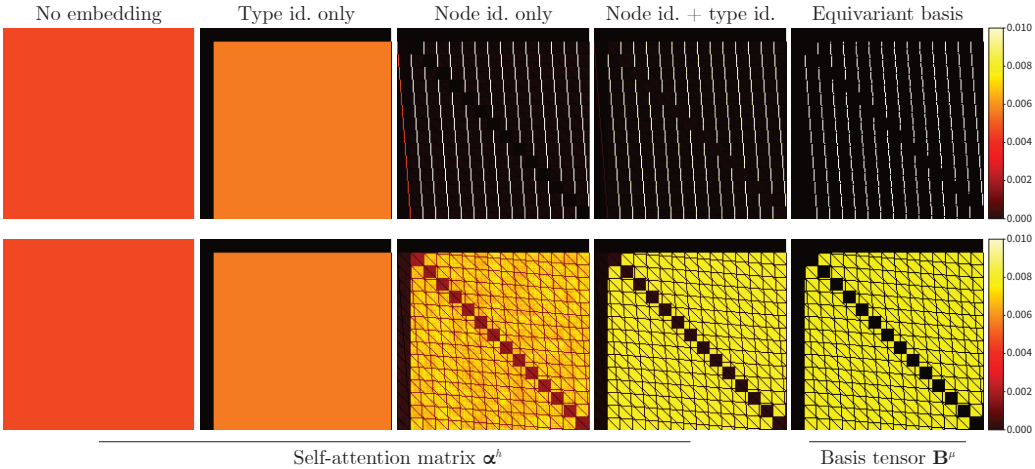

Figure 2: Self-attention maps learned under various node and type identifier configurations for two target equivariant basis tensors (out of 15). For better visualization, we clamp the entries by 0.01. Self-attention learns acute pattern coherent to equivariant basis when orthonormal node identifiers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

4 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 Soft Graph Transformer (SGT) (Section 2) using PCQM4Mv2 large-scale quantum chemistry regression benchmark [14].

4.1 Approximating Second-Order Equivariant Basis

As in Theorem 1 and 2 (Section 3), our argument on expressive power of SGT 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 $\boldsymbol{\alpha}^1, ..., \boldsymbol{\alpha}^H$ (Eq. (3)) to accurately approximate each equivariant basis $\mathbf{B}^{\mu_1}, ..., \mathbf{B}^{\mu_{\mathrm{bell}(2k)}}$ (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 in 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 the supplementary file.

Table 2: Results on PCQM4Mv2 large-scale graph regression benchmark. Since the test data is not available, we report the Mean Absolute Error (MAE) on the validation dataset.

method	# parameters	validate MAE ↓	asymptotics
Message-passing GNNs			
GCN [14]	2.0M	0.1379	$\mathcal{O}(n+m)$
GIN [14]	3.8M	0.1195	$\mathcal{O}(n+m)$
GAT	4.9M	0.1302	$\mathcal{O}(n+m)$
GCN-VN [14]	6.7M	0.1153	$\mathcal{O}(n+m)$
GIN-VN [14]	3.8M	0.1083	$\mathcal{O}(n+m)$
GAT-VN	6.7M	0.1192	$\mathcal{O}(n+m)$
GAT-VN	55.2M	0.1361	$\mathcal{O}(n+m)$
Transformers with strong graph-specific modifications			
Graphormer [33]	48.3M	0.0864	$\mathcal{O}(n^2)$
EGT [15]	89.3M	0.0869	$\mathcal{O}(n^2)$
GRPE [30]	46.2M	0.0890	$\mathcal{O}(n^2)$
Pure Transformers			
Transformer	48.5M	0.2340	$\mathcal{O}((n+m)^2)$
SGT (ORF)	48.6M	0.0962	$\mathcal{O}((n+m)^2)$
SGT (Lap)	48.5M	0.0910	$\mathcal{O}((n+m)^2)$
SGT (Lap) + Performer	48.5M	0.0935	$\mathcal{O}(n+m)$

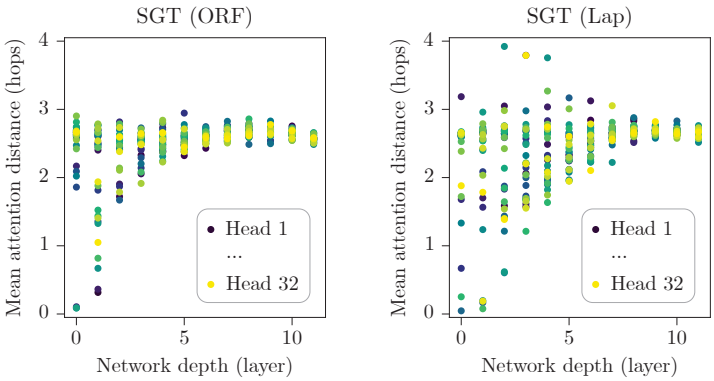

Figure 3: Attention distance by head and network depth. Each dot shows mean attention distance in hops across graphs of a head at a layer. The visualization is inspired by Dosovitskiy et al. (2020) [9].

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, ◯) lead to slightly better results than orthogonal random features (ORF, ◯) presumably due to less stochasticity. Interestingly, it shows that self-attention seems to transfer the learned (pseudo-)equivariant self-attention structure to unseen graphs near perfectly. Non-orthogonal random embeddings lead to inaccurate approximation (Random, ◯), highlighting the importance of orthogonality of node identifiers. 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 1, we provide a visualization of self-attention maps learned under various node and type identifier choices. Additional results can be found in the supplementary file.

4.2 Large-Scale Graph Learning

An exclusive characteristic of SGT 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 [35, 9], we explore the capability of SGT on PCQM4Mv2 quantum chemistry regression dataset [14], one of the current largest containing 3M molecular graphs.

For SGT, we use both node and type identifiers, and use main Transformer encoder configuration based on Graphormer [40] 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 **SGT (ORF)** and **SGT (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 [5]

that approximates the attention computation to linear cost (**SGT (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 of 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 can be found in the supplementary file.

We provide the results in Table 2. A standard Transformer on node and edge tokens cannot recognize graph structure and shows low performance (0.2340 MAE). Yet, as soon as we augment the tokens with node and type identifiers the picture changes. Notably, SGT (ORF) achieves 0.0962 MAE, which is already better than all GNN baselines. This is somewhat surprising result, as both ORF and the Transformer are not aware of graph structure. 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 [11], we observe a performance boost to 0.0910 MAE, competitive to Transformers with sophisticated graph-specific modifications (*e.g.*, shortest path based spatial encoding [40]). While such methods inject graph structure into attention matrix via bias term and therefore strictly require $\mathcal{O}(n^2)$ cost, SGT enables adaptation of kernelization for pure self-attention [5], resulting in SGT (Lap) + Performer with the best performance among $\mathcal{O}(n + m)$ models (0.0935 MAE).

While our theory in Section 3 *guarantees* that a SGT 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 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 number of hops in message-passing GNNs. In both SGT (ORF) and SGT (Lap), in lowest layers, some heads attend globally over the graph while others consistently have small receptive fields (thus acting like local message-passing operator). In deeper layers, the attention distances increase and most heads attend globally. Interestingly, this behavior is highly consistent to Vision Transformers on image patches [9], suggesting that hybrid architectures based on the use of convolution to aid ViT [7, 42] might also work well for graphs. While SGT (ORF) shows relatively consistent attention distance over heads, SGT (Lap) shows higher variance, implying that it learns more diverse attention patterns. Judging from higher performance of SGT (Lap), this suggests that the graph structure information provided by Laplacian eigenvector facilitates learning of useful and diverse attention structures, which calls for future exploration of better node identifiers based on graph PEs [20, 22].

5 Conclusion

We showed that Transformers directly applied to graphs can work well in both theory and practice. In 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, which makes 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 verify with synthetic experiment. In the experiment with PCQM4Mv2 large-scale dataset, we show that Soft Graph Transformer (SGT) performs significantly better than all GNNs and is competitive to Transformer variants with strong graph-specific architectural components [40, 15, 30].

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 require $\mathcal{O}((n + m)^2)$ asymptotic cost due to 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. Adapting Transformer engineering techniques from vision and language domains, such as data scaling [1, 9], deepening [37, 39], hybrid architectures [7, 42], and self-supervision [8, 1, 13] are promising. In 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*.

References

[1] 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, 9)

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

[3] 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)

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

[5] 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, 8, 9)

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

[7] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*, 2021. (Cited on 9)

[8] 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, 9)

[9] 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, 8, 9)

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

[11] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson. Benchmarking graph neural networks. *arXiv*, 2020. (Cited on 3, 9)

[12] 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)

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. B. Girshick. Masked autoencoders are scalable vision learners. *arXiv*, 2021. (Cited on 9)

[14] 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, 8)

[15] 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, 8, 9)

[16] 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)

[17] 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)

[18] 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)

[19] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. (Cited on 4)

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

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

[22] 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 9)

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

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

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

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

[27] 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)

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

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

[30] 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, 8, 9)

[31] 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)

[32] 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)

[33] 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 8)

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

[35] 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, 8)

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

[37] H. Wang, S. Ma, L. Dong, S. Huang, D. Zhang, and F. Wei. Deepnet: Scaling transformers to 1, 000 layers. *arXiv*, 2022. (Cited on 9)

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

[39] 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 9)

[40] 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, 8, 9)

[41] 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)

[42] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu. Incorporating convolution designs into visual transformers. In *ICCV*, 2021. (Cited on 9)

[43] 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 4.
- 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 4.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5.
   (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 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We omitted error bars in Table 1 due to space restriction but will include them in the supplementary file.
   (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 4.
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 4.2.
   (b) Did you mention the license of the assets? [Yes] See Section 4.2.
   (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]