Attending to Graph Transformers

Anonymous authors
Paper under double-blind review

Abstract

Recently, transformer architectures for graphs emerged as an alternative to established techniques for machine learning with graphs, such as (message-passing) graph neural networks. So far, they have shown promising empirical results, e.g., on molecular prediction datasets, often attributed to their ability to circumvent graph neural networks' shortcomings, such as over-smoothing and over-squashing. Here, we derive a taxonomy of graph transformer architectures, bringing some order to this emerging field. We overview their theoretical properties, survey structural and positional encodings, and discuss extensions for important graph classes, e.g., 3D molecular graphs. Empirically, we probe how well graph transformers can recover various graph properties, how well they can deal with heterophilic graphs, and to what extent they prevent over-squashing. Further, we outline open challenges and research direction to stimulate future work.

1 Introduction

Graph-structured data are prevalent across application domains ranging from chemo- and bioinformatics (Barabasi & Oltvai, 2004; Reiser et al., 2022) to image (Simonovsky & Komodakis, 2017) and social-network analysis (Easley & Kleinberg, 2010), clearly underlining the importance of machine learning methods for graph data. In recent years, *(message-passing) graph neural networks* (GNNs) (Chami et al., 2022; Gilmer et al., 2017; Morris et al., 2021) were the dominant paradigm in machine learning for graphs. However, with the rise of transformer architectures (Vaswani et al., 2017) in natural language processing (Lin et al., 2021b) and computer vision (Han et al., 2022), recently, a large number of works in the field focused on designing transformer architectures capable of dealing with graphs, so-called *graph transformers* (GTs).

Graph transformers have already shown promising performance (Ying et al., 2021), e.g., by topping the leaderboard of the OGB Large-Scale Challenge (Hu et al., 2021; Masters et al., 2022) in the molecular property prediction track. The superiority of GTs over standard GNN architecture is often explained by GNNs' bias towards encoding local structure and being unable to capture global or long-range information, often attributed to phenomena such as *over-smoothing* (Li et al., 2018), *under-reaching* (Barceló et al., 2020), and *over-squashing* (Alon & Yahav, 2021; Di Giovanni et al., 2023). Many papers (Rampášek et al., 2022) speculate that GTs do not suffer from such effects as they aggregate information over all nodes in a given graph and hence are not limited to local structure bias. However, to make GTs aware of graph structure, one has to equip them with so-called *structural* and *postional encodings*. Here, structural encodings are, e.g., additional node features to make the GT aware of (sub-)graph structure. In contrast, positional encodings make a node aware of its position in the graph concerning the other nodes.

Present Work. Here, we derive a taxonomy of state-of-the-art GT architectures, giving an organized overview of recent developments. In addition, we survey common positional and structural encodings and clarify how they are related to GTs' theoretical properties, e.g., their expressive power to capture graph structure. Additionally, we investigate these properties empirically by probing how well GTs can recover various graph properties, deal with heterophilic graphs, and to what extent GTs alleviate the over-squashing phenomenon. Further, we outline open challenges and research direction to stimulate future work. Our categorization, theoretical clarification, and experimental study present a useful handbook for the GT and the broader graph machine-learning community. Its insights and principles will help spur novel research results and avenues.

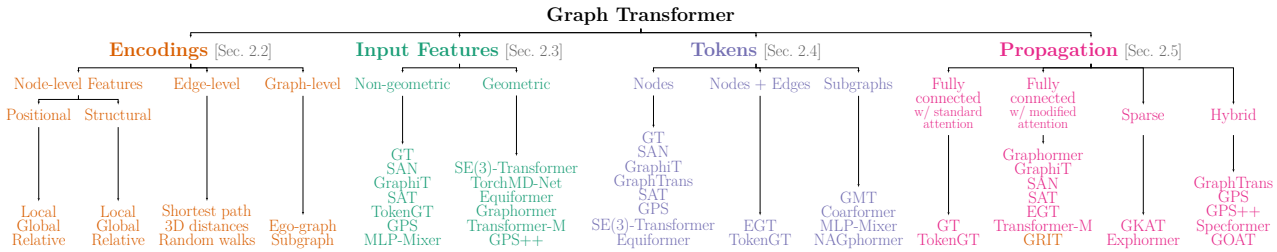

Figure 1: Categorization of graph transformers along four main categories with representative architectures: Encodings; see Section 2.2, input Features; see Section 2.3, tokens; see Section 2.4, propagation; see Section 2.5. See also Figure 3 detailing how the different branches translate into changes to the original transformer.

Related Work. Since GTs emerged recently, only a few surveys exist. Notably, Min et al. (2022a) provide a high-level overview of some of the recent GT architectures. Different from the present work, they do not discuss GTs' theoretical and practical shortcomings and miss out on recent architectural advancements. Chen et al. (2022a) gives an overview of GTs for computer vision. Finally, Rampášek et al. (2022) provide a general recipe for classifying GT architectures, focusing on devising empirically well-performing architectures rather than giving a detailed, principled overview of the literature.

1.1 Background

A *graph* G is a pair $(V(G), E(G))$ with a *finite* set of *nodes* $V(G)$ and a set of *edges* $E(G) \subseteq \{\{u, v\} \subseteq V \mid u \neq v\}$. For ease of notation, we denote an edge $\{u, v\}$ as (u, v) or (v, u). In the case of *directed graphs*, $E(G) \subseteq \{(u, v) \in V^2 \mid u \neq v\}$. Throughout the paper, we set $n \coloneqq |V(G)|$ and $m \coloneqq |E(G)|$. A *node-attributed graph* G is a triple $(V, E, \mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{n \times d}$, for $d > 0$, is a *node feature matrix* and $\mathbf{X}_v$ is the node feature of node $v \in V(G)$. Similarly, we can represent edge features by an *edge feature matrix* $\mathbf{E} \in \mathbb{R}^{m \times e}$, for $e > 0$, where $\mathbf{E}_{vw}$ is the edge feature of edge $(v, w) \in E(G)$. The *neighborhood* of $v \in V(G)$ is $N(v) \coloneqq \{u \in V(G) \mid (v, u) \in E(G)\}$. We say that two graphs G and H are *isomorphic* if there exists an edge-preserving bijection $\varphi \colon V(G) \to V(H)$, i.e., (u, v) is in $E(G)$ if and only if $(\varphi(u), \varphi(v))$ is in $E(H)$ for all $u, v \in V(G)$. We denote a multiset by $\{\!\{\ldots\}\!\}$.

Equivariance and Invariance. Operations on graphs need to respect their symmetries, such as being agnostic to the node permutations or other (group) transformations, such as rotation, leading to the definitions of *equivariance* and *invariance*. In general (Fuchs et al., 2021), given a transformation $\mathbf{T}$, a function f is equivariant if transforming the vector input $\mathbf{x}$ is equal to transforming the output of the function f, i.e., $f(\mathbf{T}\mathbf{x}) = \mathbf{T}f(\mathbf{x})$. A function g is invariant if transforming the vector input $\mathbf{x}$ does not change the output, i.e., $g(\mathbf{T}\mathbf{x}) = g(\mathbf{x})$. In the 3D Euclidean space, 3D translations, rotations, and reflections form the $E(3)$ group. Translation and rotation form the $SE(3)$ group. Rotations form the $SO(3)$ group, rotations and reflections form the $O(3)$ group.

Graph Transformers. A transformer is a stack of alternating blocks of *multi-head attention* and fully-connected *feed-forward networks*. Let G be a graph with node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$.[1] In each layer, $t > 0$, given node feature matrix $\mathbf{X}^{(t)} \in \mathbb{R}^{n \times d}$, a single attention head computes

$$\mathsf{Attn}(\mathbf{X}^{(t)}) \coloneqq \operatorname{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{1}$$

where the softmax is applied row-wise, d_k denotes the feature dimension of the matrices $\mathbf{Q}$ and $\mathbf{K}$, with $\mathbf{X}^{(0)} \coloneqq \mathbf{X}$. Here, the matrices $\mathbf{Q}, \mathbf{K}$, and $\mathbf{V}$ are the result of projecting $\mathbf{X}^{(t)}$ linearly,

$$\mathbf{Q} \coloneqq \mathbf{X}^{(t)}\mathbf{W}_Q, \quad \mathbf{K} \coloneqq \mathbf{X}^{(t)}\mathbf{W}_K, \quad \text{and} \quad \mathbf{V} \coloneqq \mathbf{X}^{(t)}\mathbf{W}_V,$$

[1] For simplicity, we learn attention between a graph's nodes. However, in Section 2.4, we extend this to, e.g., edges or subgraphs.

using three matrices $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d_K}$, and $\mathbf{W}_V \in \mathbb{R}^{d \times d}$, with optional bias terms omitted for clarity. Now, multi-head attention $\mathsf{MultiHead}(\mathbf{X}^{(t)})$ concatenates multiple (single) attention heads, followed by an output projection to the feature space of $\mathbf{X}^{(t)}$. By combining the above with additional residual connections and normalization, the transformer layer updates features $\mathbf{X}^{(t)}$ via

$$\mathbf{X}^{(t+1)} \coloneqq \mathsf{FFN}\big(\mathsf{MultiHead}\big(\mathbf{X}^{(t)}\big) + \mathbf{X}^{(t)}\big). \tag{2}$$

Graph Neural Networks. Intuitively, GNNs learn a vector representing each node in a graph by aggregating information from neighboring nodes. Formally, let G be a graph with node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. A GNN architecture consists of a stack of neural network layers, i.e., a composition of permutation-equivariant parameterized functions. Each layer aggregates local neighborhood information, i.e., the neighbors' features, around each node and then passes this aggregated information on to the next layer. Following Gilmer et al. (2017) and Scarselli et al. (2009), in each layer, $t \geq 0$, we compute vertex features

$$\mathbf{h}_v^{(t+1)} \coloneqq \mathsf{UPD}^{(t)}\Big(\mathbf{h}_v^{(t)},\ \mathsf{AGG}^{(t)}\big(\{\!\!\{\mathbf{h}_u^{(t)} \mid u \in N(v)\}\!\!\}\big)\Big) \in \mathbb{R}^d,$$

where $\mathsf{UPD}^{(t)}$ and $\mathsf{AGG}^{(t)}$ may be differentiable parameterized functions, e.g., neural networks. For example, GNNs often compute a vector for node v by using sum aggregation (Morris et al., 2019), i.e.,

$$\mathbf{h}_v^{(t+1)} \coloneqq \sigma\Big(\mathbf{h}_v^{(t)}\mathbf{W}_1^{(t)} + \sum_{w \in N(v)} \mathbf{h}_w^{(t)}\mathbf{W}_2^{(t)}\Big),$$

where σ is a non-linearity applied pointwise, $\mathbf{W}_1^{(t)}$ and $\mathbf{W}_2^{(t)} \in \mathbb{R}^{d \times d}$ are parameter matrices, and $\mathbf{h}_v^{(0)} \coloneqq \mathbf{X}_v$. As noticed by Mialon et al. (2021), we can rewrite attention as defined in Eq. (1) as

$$\mathsf{Attn}(\mathbf{X}^{(t)})_v = \sum_{u \in V(G)} \frac{k_{\exp}(\mathbf{X}_v^{(t)}, \mathbf{X}_u^{(t)})}{\sum_{w \in V(G)} k_{\exp}(\mathbf{X}_v^{(t)}, \mathbf{X}_w^{(t)})} \mathbf{X}_u^{(t)}\mathbf{W}_V, \tag{3}$$

for $v \in V(G)$, where

$$k_{\exp}(\mathbf{X}_v^{(t)}, \mathbf{X}_w^{(t)}) \coloneqq \exp\big(\mathbf{X}_v^{(t)}\mathbf{W}_Q\mathbf{X}_w^{(t)}\mathbf{W}_K/\sqrt{d_K}\big)$$

and obtain an aggregation reminiscent of a GNN. In fact, we may view a GT as a special GNN, operating on a complete graph, where the attention score weights the importance of each node during the sum aggregation. Conversely, Velickovic et al. (2018) propose graph attention networks (GAT), a GNN that replaces sum aggregation with self-attention. We note that their definition of self-attention differs from Equation (1) in that GAT replaces the dot-product with concatenation and subsequent projection.

2 The Landscape of Graph Transformers

In the following, we outline our taxonomy of GTs, see also Figure 1, bringing some order to the growing set of GT architectures. We start by discussing the theoretical properties of GTs that heavily rely on structural and positional encodings, which we study subsequently. Further, we discuss different approaches to dealing with essential classes of input node features, e.g., 3D coordinates in the case of molecules. We then study how to *tokenize* a graph, i.e., partition a graph into atomic entities between which the attention is computed, e.g., nodes. Then, we review how GTs organize message propagation in the graph through global, sparse, or hybrid attention. Finally, we overview representative applications of GTs.

2.1 Theoretical Properties

It is crucial to understand that the general GT architecture of Eq. (2) is less expressive in distinguishing non-isomorphic graphs than standard GNNs. Hence, it is also weaker in approximating permutation-invariant and -equivariant functions over graphs (Chen et al., 2019). GTs are weaker since, without sufficiently expressive structural and positional encodings, they cannot capture any graph structure besides the number

of nodes and hence equal DeepSets-like architectures (Zaheer et al., 2020) in expressive power. Thus, for GTs to capture non-trivial graph structure information, they are crucially dependent on such encodings; see below. In fact, by leveraging the results in Chen et al. (2019), it is easy to show that GTs can only become maximal expressive, i.e., universal function approximators, if they have access to maximally expressive structural bias, e.g., structural encodings. However, this is equivalent to solving the graph isomorphism problem (Chen et al., 2019). Moreover, we stress that GNN architectures equipped with the same encodings will also possess the same expressive power. Hence, in terms of expressive power, GTs do not have an advantage over GNNs.

Cai et al. (2023) study the connection between GNNs and graph transformers, showing that under mild assumptions, GNNs with a virtual node connected to all other nodes are sufficient to simulate a graph transformer. Conversely, Kim et al. (2022) propose a hierarchy of (higher-order) transformers and respective positional encodings that is aligned with k-IGNs (Maron et al., 2019), i.e., for each $k > 1$, there exists a corresponding higher-order transformer that can simulate a k-IGN. The relationship between k-IGNs and k-WL (Maron et al., 2019) then implies the connection between transformers and the k-WL hierarchy as well.

2.2 Structural and Positional Encodings

As outlined in the previous subsection, GTs are crucially dependent on structural and positional encodings to capture graph structure. Although there is no formal definition or distinction between the two, structural encodings make the GT aware of graph structure on a *local*, *relative*, or *global* level. Such encodings can be attached to node-, edge-, or graph-level features. Examples of local structural encodings include annotating node features with node degrees (Chen et al., 2022b), the diagonal of the m-step random-walk matrix (Dwivedi et al., 2022), the time-derivative of the heat-kernel diagonal (Kreuzer et al., 2021), enumerate or count predefined substructures and the node's role within (Bouritsas et al., 2022), or Ricci curvature (Topping et al., 2022). Examples of edge-level relative structural encodings include relative shortest-path distances (Chen et al., 2022a) or Boolean features indicating if two nodes are in the same substructure (Bodnar et al., 2021). Examples of graph-level global structural encodings include eigenvalues of the adjacency or Laplacian (Kreuzer et al., 2021), or graph properties such as diameter, number of connected components, or treewidth.

On the other hand, positional encodings make, e.g., a node, aware of its relative position to the other nodes in a graph. Hence, two such encodings should be close to each other if the corresponding nodes are close in the graph. Again, we can distinguish between local, global, or relative encodings. Examples of node-level local positional encodings include the shortest-path distance of a node to a hub or central node or the sum of each column of the non-diagonal elements of the m-step random walk matrix. Example of edge-level relative positional encodings are pair-wise node distances (Chen et al., 2022a; Beaini et al., 2021; Kreuzer et al., 2021; Mialon et al., 2021; Li et al., 2020) and relative random walk encodings (Ma et al., 2023). Examples of node-level global positional encodings include eigenvectors of the graph Laplacian (Kreuzer et al., 2021; Dwivedi & Bresson, 2020) (or of the Magnetic Laplacian in case of directed graphs (Geisler et al., 2023)), or unique identifiers for each connected component of the graph. Lim et al. (2022) propose SignNet and BasisNet, two positional encodings also based on the eigenvectors of the graph Laplacian which generalize a number of previously introduced structural and positional encodings such as those based on random walks (Dwivedi & Bresson, 2020; Mialon et al., 2021) or PageRank (Li et al., 2020).

When designing such encodings, one must ensure equivariance or invariance to the nodes' ordering. Such equivariance is trivially satisfied for simple encodings such as node degree but not for more powerful encodings based on eigenvectors of the adjacency or Laplacian matrix (Lim et al., 2022). It is an ongoing effort to design equivariant Laplacian-based encodings (Lim et al., 2022; Wang et al., 2022).

2.3 Input Features

Besides characterizing GTs based on their use of structural and positional encodings, we can also characterize them based on their ability to deal with different node and edge features. To this end, we devise two families of input features. First, we consider so-called *non-geometric features* where nodes and edges have feature vectors in $\mathbb{R}^d$, i.e., graphs are described with a tuple $(V, E, \mathbf{X}, \mathbf{E})$. Secondly, we consider so-called *geometric features* where nodes and edges features contain geometric information, e.g., 3D coordinates for nodes $\mathbf{X}^{3\text{D}} \in \mathbb{R}^3$.

Therefore, graphs are described with $(V, E, \mathbf{X}, \mathbf{E}, \mathbf{X}^{3D}, \mathbf{E}^{3D})$. We categorize GT architectures as non-geometric and those supporting both features in the following.

Non-geometric GTs (Chen et al., 2022b; Choromanski et al., 2021; Dwivedi & Bresson, 2020; He et al., 2022; Kim et al., 2022; Kreuzer et al., 2021; Jain et al., 2021; Ma et al., 2023; Mialon et al., 2021; Rampášek et al., 2022) are most common and follow equations in Section 1.1. Graphs with non-geometric features do not have explicit geometric inductive bias. Examples of such features include encoded node attributes in citation networks or learnable atom-type embeddings in molecular graphs. Non-geometric features are supposed to be *equivariant* to node permutations, and transformers provide such equivariance by default. Structural and positional features (Section 2.2) are often added to increase the expressive power of GTs.

3D molecular graphs provide geometric features describing nodes and edges, e.g., 3D coordinates of atoms, angles of bonds, or torsion angles of planes. Building GTs supporting geometric features is more challenging as geometric features have to be *invariant* or *equivariant* to certain group transformations, such as rotation, depending on the task. Further, the architectures must be invariant for graph-level molecular property prediction tasks. In contrast, models must be equivariant in node-level tasks such as predicting structural conformers or force fields.

Joshi et al. (2023) showed that equivariant geometric models are more expressive than invariant ones. Here, we first describe $SO(3)$, $SE(3)$, and $E(3)$ equivariant models and then turn the attention to invariant models. TorchMD-NET (Thölke & Fabritiis, 2022) achieves $SO(3)$ equivariance by incorporating interatomic distances into the attention operation via exponential normal radial basis functions (RBF). SE(3)-Transformer (Fuchs et al., 2020) was one of the first attempts to incorporate $SE(3)$ equivariance. By using irreducible representations, Clebsch-Gordan coefficients, and spherical harmonics, the authors encode $SE(3)$ equivariance into the attention operation. Equiformer (Liao & Smidt, 2023) further extends this mechanism to complete $E(3)$ equivariance. Graphormer (Shi et al., 2022), Transformer-M (Luo et al., 2022a) and GPS++ (Masters et al., 2022) use Gaussian kernels to encode 3D distances between all pairs of atoms. Tailored for graph-level prediction tasks, GPS++ remains SE(3)-invariant, while Graphormer and Transformer-M introduce an additional SE(3)-equivariant prediction head for node-level molecular dynamics tasks.

2.4 Graph to Sequence Tokenization

The nature of *graph tokenization*, i.e., mapping a graph into a sequence of *tokens*, directly affects the supported features and computational complexity. Here, we identify three approaches to graph tokenization: (1) nodes as tokens, (2) nodes and edges as tokens, and (3) patches or subgraphs as tokens.

Using nodes as input tokens is the most common approach followed by many GTs, e.g., (Dwivedi & Bresson, 2020; Fuchs et al., 2020; Kreuzer et al., 2021; Luo et al., 2022b; Rampášek et al., 2022; Thölke & Fabritiis, 2022; Ying et al., 2021). Here, we often treat structural and positional features as additional node features. Given a graph with n nodes and the attention procedure of Eq. (1), the complexity of such GTs is in $\mathcal{O}(n^2)$. We note that more scalable, sparse attention mechanisms are also possible; see Section 2.5. Edge features, e.g., shortest-path distances (Ying et al., 2021), random walk relative distances (Ma et al., 2023), or relative 3D distances (Luo et al., 2022b; Thölke & Fabritiis, 2022), may be added as an attention bias given the fully computed attention score matrix with n^2 entries. Alternatively, Mialon et al. (2021); Jain et al. (2021); Chen et al. (2022b) leverage a GNN to incorporate node and edge features before applying a transformer on the resulting node features. However, the transformer's quadratic complexity remains the bottleneck.

The second approach uses nodes and edges in the input sequence as employed by EGT (Hussain et al., 2022) and TokenGT (Kim et al., 2022). Turning an input graph into a graph of its edges is often used in molecular GNNs (Gasteiger et al., 2021) and NLP (Yao et al., 2020). In addition to soft modeling the edges, i.e., the *node-to-node* interactions, the attention operation also possibly models higher-order *node-edge* and *edge-edge* interactions that theoretically result in an expressiveness boost (Kim et al., 2022). The input sequence can naturally incorporate node features, their positional encodings, and edge features. A pitfall of this approach is its $\mathcal{O}(n+m)^2$ computational complexity. Since the approach includes edge features in the input sequence, such GTs might benefit from sparse attention mechanisms that do not materialize the full attention matrix.

The third approach relies on *patches* or *subgraphs* as tokens. In visual transformers (Dosovitskiy et al., 2021), such patches correspond to $k \times k$ image slices. A generalization of patches to the graph domain often corresponds to graph coarsening or partitioning (Baek et al., 2021; Chen et al., 2023; Kuang et al., 2022; He et al., 2022). There, tokens are small subgraphs extracted with various strategies. Initial representations of tokens are obtained by passing subgraphs through a GNN using a form of pooling to a single vector. He et al. (2022) adds token position features to the resulting vectors to distinguish coarsened subgraphs better. Finally, these tokens are passed through a transformer with $\mathcal{O}(k^2)$ complexity for a graph with k extracted subgraphs. A similar approach is taken by NAGphormer (Chen et al., 2023) where tokens denote aggregated l-hop neighborhoods of a node. Each node is represented with a sequence of up to L tokens including the node itself and each token for $l \in [1, L-1]$-hop neighborhood. Subgraph tokenization is more scalable since the model never sees the whole graph and might benefit from sampling methods. On the other hand, it makes GTs inherently local and might hinder their long-range capabilities.

2.5 Message Propagation

Most GTs follow the global all-to-all attention of Vaswani et al. (2017) between all pairs of tokens. In the initial GT (Dwivedi & Bresson, 2020) and TokenGT (Kim et al., 2022) this mechanism is unchanged, relying on token representations augmented with graph structural or positional information. Other models alter the global attention mechanism to bias it explicitly, typically based on the input graph's structural properties. Graphormer (Ying et al., 2021) incorporates shortest-path distances, representation of edges along a shortest path, and node degrees. Transformer-M (Luo et al., 2022a) follows Graphormer and adds kernelized 3D inter-atomic distances. GRPE (Park et al., 2022) considers multiplicative interactions of keys and queries with node and edge features instead of Graphormer's additive bias and additionally augments output token values. SAN (Kreuzer et al., 2021) relies on positional encodings and only adds preferential bias to interactions along input-graph edges over long-distance virtual edges. GraphiT (Mialon et al., 2021) employs diffusion kernel bias, while SAT (Chen et al., 2022b) develops a GNN-based structure-aware attention kernel. EGT (Hussain et al., 2022) includes a mechanism akin to cross-attention to edge tokens to bias inter-node attention and update edge representations. Finally, Zhang et al. (2023) propose Graphormer-GD, a generalization of Graphormer, alongside new expressivity metrics based on graph biconnectivity for which Graphormer-GD attains the necessary expressivity.

As standard global attention incurs quadratic computational complexity, it limits the application of graph transformers to graphs of up to several thousands of nodes. To alleviate this scaling issue, Choromanski et al. (2022) proposed GKAT based on a kernelized attention mechanism of the Performer (Choromanski et al., 2021), scaling linearly with the number of tokens. Another approach to improve GTs' scaling is to consider a reduced attention scope, e.g., based on locality or sparsified instead of dense all-to-all, following expander graph-based propagation (Deac et al., 2022) as in Exphormer (Shirzad et al., 2023). Finally, Wu et al. (2022) propose to view attention as in Equation (3) and then apply the kernel trick to k_{exp} to derive NodeFormer with a computational complexity that scales linearly with the number of nodes.

Finally, hybrid approaches combine several propagation schemes. For example, GPS and GPS++ (Rampášek et al., 2022; Masters et al., 2022) fuse local GNN-like architectures with global all-to-all attention into one layer. While GPS employs standard attention and can utilize linear attention mechanisms such as Performer (Choromanski et al., 2022), GPS++ follows Transformer-M's attention conditioning. Graph-Trans (Jain et al., 2021) is also a hybrid but applies a stack of GNN layers first, followed by a stack of global transformer layers. Specformer (Bo et al., 2023) employs transformer encoder layers for encoding eigenvalue representations and mixes those with node features in the graph convolution-style decoder. GOAT (Kong et al., 2023) lets the nodes attend to k cluster centroids instead of n nodes on the global attention level. The centroids are obtained through k-means and exponential moving average. On the local level, nodes attend to their k-hop neighborhood.

3 Applications of Graph Transformers

Although GTs only emerged recently, they have already been applied in various application areas, most notably in molecular property prediction. In the following, we give an overview of the applications of GTs.

Kan et al. (2022) propose the Brain Network Transformers to predict properties of brain networks, e.g., the presence of diseases, stemming from magnetic resonance imaging. To that, they leverage rows of the adjacency matrix of each node as structural encodings, which showed superior performance over Laplacian-based encodings in previous studies. Moreover, they devise a custom pooling layer leveraging the fact that nodes in the same functional module tend to have similar properties.

Liao & Smidt (2023); Thölke & Fabritiis (2022), see also Section 2.3, devise an equivariant transformer architecture to predict quantum mechanical properties of molecules. To capture the molecular structure, they encode atom types and the atomic neighborhood into a vectorial representation, followed by a multi-head attention mechanism. To predict scalar atom-wise prediction, they rely on gated equivariant blocks (Schütt et al., 2021), which are then aggregated into single molecular predictions.

Hu et al. (2020) develop an approach to apply graph transformers to web-scale heterogeneous graphs. During the attention computation, the authors use different projection matrices for each node and edge relation in the heterogeneous graph.

Yao et al. (2020) use transformers to tackle the graph-to-sequence problem, i.e., the problem of translating a graph to word sequences. They first translate a graph to its Levi graph, replacing labeled edges with additional nodes to incorporate edge labels. They then split such a graph into multiple subgraphs according to the different edge nodes. Each subgraph uses a standard transformer architecture to learn the vectorial representation for each node. To incorporate graph structure, they mask out non-neighbors of a node, concentrating on the local structure. Finally, they concatenate multiple node representations. Further applications use transformers for rumor detection in microblogs (Khoo et al., 2020), predicting properties of crystals (Yan et al., 2022) or click-through rates (Min et al., 2022b), or leverage them for 3D human pose and mesh reconstruction from a single image (Lin et al., 2021a).

4 Experimental Study

We empirically evaluate two highly discussed aspects of graph transformers: (1) the effectiveness of incorporating *graph structural bias* into GTs, and (2) their ability to reduce *over-smoothing* and *over-squashing*. This study is aimed at both comparing selected methods from our taxonomy as well as to investigate graph transformers more generally in terms of their potential benefits over GNNs. Concretely, we aim to answer the following questions.

Q1 How well do different strategies for incorporating structural awareness into GTs contribute to recovering fundamental structural properties of graphs?

Q2 Does the ability of transformers to reduce over-smoothing lead to improved performance on heterophilic datasets?

Q3 Do graph transformers alleviate over-squashing better than GNN models?

4.1 Structural Awareness of GTs

For question **Q1**, we evaluate the two most prevalent strategies for incorporating graph structure bias into transformers.

Positional and Structural Encodings (Section 2.2). Random-walk structural encodings (RWSE) and Laplacian positional encodings (LapPE), two popular positional or structural encodings for transformers (Rampášek et al., 2022).

Attention Bias (Section 2.5). Attention bias based on spatial information such as shortest-path distance between nodes, following the Graphormer architecture (Ying et al., 2021).

We propose a benchmark of three tasks that require increasingly higher structural awareness of non-geometric graphs. We determine the level of structural awareness necessary to solve a task according to the baseline performance of GIN (Xu et al., 2019), a maximal-expressive GNN reference architecture. In addition, we

Table 1: Hyper-parameter sets for GTs and GNNs with or without PE/SE (SET 1), and for Graphormer models (SET 2).

Hyper-parameter	SET 1	SET 2
Embedding dim.	64	72
Self-attention heads	4	4
Weight decay	10^{-5}	10^{-2}
Learning rate	10^{-3}	10^{-3}
Gradient clip norm	1.0	5.0
LR scheduler	cosine, warm-up	constant
Batch size	96	256

Table 2: Average test accuracy of GTs with structural bias (± SD) over five random seeds on the structural awareness tasks. Difficulty level on top derived from GIN performance. We additionally report the performance of a transformer without any structural bias serving as a baseline.

Model	**Easy**	**Medium**		**Hard**
	EDGES	TRIANGLES-SMALL	TRIANGLES-LARGE	CSL
	2-way Accuracy ↑	**10-way Accuracy ↑**	**10-way Accuracy ↑**	**10-way Accuracy ↑**
GIN	**98.11** ±1.78	71.53 ±0.94	33.54 ±0.30	10.00 ±0.00
Transformer	55.84 ±0.32	12.08 ±0.31	10.01 ±0.04	10.00 ±0.00
Transformer (LapPE)	**98.00** ±1.03	78.29 ±0.25	10.64 ±2.94	**100.00** ±0.00
Transformer (RWSE)	97.11 ±1.73	**99.40** ±0.10	**54.76** ±7.24	**100.00** ±0.00
Graphormer	97.67 ±0.97	**99.09** ±0.31	42.34 ±6.48	90.00 ±0.00

report the performance of a vanilla transformer without any structural bias to understand the relative impact of the positional or structural encodings (PE/SE) and attention biasing.

We first describe the tasks in our benchmark and their estimated difficulty, then outline task-specific hyper-parameters of evaluated models and interpret the observed results; see Table 2 for quantitative results.

Detect Edges (Easy). Detecting whether an edge connects two nodes can be considered the fundamental test for structural awareness. We investigate this task using a custom dataset, EDGES, derived from the ZINC (Dwivedi et al., 2023) dataset. For each graph, we treat the pairs of nodes connected by an edge as positive examples and select an equal number of unconnected nodes as negative examples, resulting in a binary edge detection task with balanced classes. Let P denote the set of pairs selected as either positive or negative examples, and let $\mathbf{h}_v^{(T)}$ denote the feature vector of node v after the last layer T of a model. We make predictions as follows. We first compute the cosine similarity between $\mathbf{h}_v^{(T)}$ and $\mathbf{h}_w^{(T)}$ for each pair (v, w) of nodes in P, resulting in a scalar similarity score. Finally, we apply a linear layer to each similarity score, followed by a sigmoid activation, resulting in binary class probabilities.

Count Triangles (Medium). Counting triangles only requires information within a node's immediate neighborhood. However, more than 1-WL expressivity is required to solve it (Morris et al., 2019). Hence, a standard GIN architecture is not powerful enough. For this task, we evaluate models on the TRIANGLES dataset proposed by Knyazev et al. (2019), which poses triangle counting as a 10-way classification problem. Here, graphs have between 1 and 10 triangles, each corresponding to one class. The dataset specifies a fixed train/validation/test split, which we adopt in our experiments. Graphs in the train and validation split are roughly the same size. The test set is a mixture of two graph distributions, where 50% are graphs with a similar size to those in the training and validation set (TRIANGLES-SMALL) and 50% are graphs of larger size (TRIANGLES-LARGE). We separately report model performance for TRIANGLES-SMALL and TRIANGLES-LARGE to study the ability of transformers with different structural biases to generalize to larger

graphs. We analyzed the datasets' class balance and report that each test set contains 5000 graphs with 500 graphs per class. For more details about the dataset, see Knyazev et al. (2019).

Distinguish Circular Skip Links (CSL) (Hard). A 1-WL limited model cannot distinguish non-isomorphic CSL graphs (Murphy et al., 2019) as the task requires an understanding of distance (Morris et al., 2019). Here, we evaluate models on the CSL dataset (Dwivedi et al., 2023), which contains 150 graphs with skip-link lengths ranging from 2 to 16 and poses a 10-way classification problem. We follow Dwivedi et al. (2023) in training with 5-fold cross-validation.

Hyper-parameters. To simplify hyper-parameter selection, we hand-designed two general sets of hyper-parameters; see Table 1. For Edges and Triangles, we fix a parameter budget of around 200k for the transformer models, resulting in six layers for each model with the respective embedding sizes specified in Table 1. Further, we train Graphormer on 1k epochs. Due to the small number of graphs in the CSL dataset, we fix a parameter budget of around 100k for the transformer models, resulting in three layers for each model with the exact embedding sizes as above. Further, we train Graphormer on 2k epochs. We repeat each experiment on five random seeds and report the model accuracy's mean and standard deviation.

For our 1-WL-equivalent reference model, we chose the GIN layer (Xu et al., 2019). To improve comparability with the transformer models, we use a feed-forward neural network composed of the same components, using the same hyper-parameters as for transformers. For Graphormer, we use the feed-forward neural network specified by Ying et al. (2021). Further, we train GIN with the Set 1 hyper-parameters. For Edges and Triangles, this results in around 150k parameters, while for CSL, the GIN model contains approximately 75k parameters.

Answering Q1. Table 2 shows that GTs supplemented with structural bias generally perform well on all three tasks with a few exceptions. First, the GT with Laplacian positional encodings performs sub-par on the Triangles task. However, it is still an improvement over the 1-WL-equivalent GIN. We hypothesize this is due to an expressivity limit of Laplacian encodings regarding triangle counting. Secondly, we observe that all models generalize poorly to larger graphs on the Triangles dataset. Lastly, we observe that on CSL, Graphormer cannot surpass 90% accuracy. A deeper analysis revealed that the shortest-path distributions can only distinguish 9 out of the 10 classes correctly, meaning that Graphormer is theoretically limited to at most 90% accuracy on CSL.

The above failure cases highlight that current graph transformers still suffer from limited expressivity, and no clear expressivity hierarchy exists for the used positional or structural encodings. Moreover, GTs may generalize poorly to larger graphs. At the same time, we demonstrate a general superiority of structurally biased GTs over standard 1-WL-equivalent models such as GIN. Both the transformer with RWSE as well as Graphormer solve Edges, Triangles-small, and CSL almost perfectly, two of which pose a challenge for GIN, especially on CSL where GIN performs no better than random.

4.2 Reduced Over-smoothing in GTs?

Graph transformers are often ascribed with an ability to circumvent GNNs' over-smoothing problem due to their global attention mechanism. Thus, we set out to benchmark several variants of GCN (Kipf & Welling, 2017), hybrid GPS models, and Graphormer on six heterophilic transductive datasets: Actor (Tang et al., 2009); Cornell, Texas, Wisconsin (CMU, 2001); Chameleon and Squirrel (Rozemberczki et al., 2021). In addition, we also consider the recently proposed heterophilic datasets in (Platonov et al., 2023), which are significantly larger (ranging from 10k to 45k nodes), where we benchmark GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018), as well as GPS. In all of the above datasets, we expect over-smoothing and under-reaching to be limiting factors.

For the six small datasets, we broadly follow the Set 1 hyper-parameters (Table 1). However, we perform a grid search for each model variant to select the embedding size (32, 64, or 96) and dropout rates while we fix the number of layers to two. We implement the GCN and GT models following GPS with hybrid GCN+Transformer aggregation layers but with the latter or former component disabled, respectively. We train all models in full-batch mode using the entire graph as input.

Table 3: Benchmarking of multiple model variants on six heterophilic transductive datasets. Here we report average test accuracy (± SD) over ten random seeds. We follow the dataset protocol of Pei et al. (2020); for additional model comparison; see Luan et al. (2022).

Model (PE/SE type)	Actor	Cornell	Texas	Wisconsin	Chameleon	Squirrel
Geom-GCN (Pei et al., 2020)	31.59 ±1.15	60.54 ±3.67	64.51 ±3.66	66.76 ±2.72	**60.00 ±2.81**	38.15 ±0.92
GCN (no PE/SE)	33.92 ±0.63	53.78 ±3.07	65.95 ±3.67	66.67 ±2.63	43.14 ±1.33	30.70 ±1.17
GCN (LapPE)	34.30 ±1.12	56.22 ±2.65	65.95 ±3.67	66.47 ±1.37	43.53 ±1.45	30.80 ±1.38
GCN (RWSE)	33.69 ±1.07	53.78 ±4.09	62.97 ±3.21	69.41 ±2.66	43.84 ±1.68	31.77 ±0.65
GCN (DEG)	33.99 ±0.91	53.51 ±2.65	66.76 ±2.72	67.26 ±1.53	46.36 ±2.07	34.50 ±0.87
$\text{GPS}^{\text{GCN+Transformer}}$ (LapPE)	37.68 ±0.52	66.22 ±3.87	75.41 ±1.46	74.71 ±2.97	48.57 ±1.02	35.58 ±0.58
$\text{GPS}^{\text{GCN+Transformer}}$ (RWSE)	36.95 ±0.65	65.14 ±5.73	73.51 ±2.65	**78.04 ±2.88**	47.57 ±0.90	34.78 ±1.21
$\text{GPS}^{\text{GCN+Transformer}}$ (DEG)	36.91 ±0.56	64.05 ±2.43	73.51 ±3.59	75.49 ±4.23	52.59 ±1.81	42.24 ±1.09
Transformer (LapPE)	**38.43 ±0.87**	**69.46 ±1.73**	**77.84 ±1.08**	76.08 ±1.92	49.69 ±1.11	35.77 ±0.50
Transformer (RWSE)	**38.13 ±0.63**	**70.81 ±2.02**	**77.57 ±1.24**	**80.20 ±2.23**	49.45 ±1.34	35.35 ±0.75
Transformer (DEG)	37.39 ±0.50	**71.89 ±2.48**	**77.30 ±1.32**	**79.80 ±0.90**	56.18 ±0.83	**43.64 ±0.65**
Graphormer (DEG only)	36.91 ±0.85	68.38 ±1.73	76.76 ±1.79	77.06 ±1.97	54.08 ±2.35	**43.20 ±0.82**
Graphormer (DEG, attn. bias)	36.69 ±0.70	68.38 ±1.73	76.22 ±2.36	77.65 ±2.00	53.84 ±2.32	**43.75 ±0.59**

Table 4: Benchmarking of multiple model variants on the five transductive node classification datasets proposed in (Platonov et al., 2023). Here we report average test accuracy (± SD) for Roman-Empire and Amazon-Ratings and ROC AUC (± SD) for Minesweeper, Tolokers and Questions. Results over 10 splits, following (Platonov et al., 2023). We highlight the **first**, **second** and **third** best results.

Model (PE/SE type)	Roman-Empire	Amazon-Ratings	Minesweeper	Tolokers	Questions
GCN Platonov et al. (2023)	73.69 ±0.74	48.70 ±0.63	89.75 ±0.52	83.64 ±0.67	76.09 ±1.27
GAT Platonov et al. (2023)	80.87 ±0.30	49.09 ±0.63	92.01 ±0.68	83.70 ±0.47	77.43 ±1.20
GCN (LapPE)	83.37 ±0.55	44.35 ±0.36	**94.26 ±0.49**	84.95 ±0.78	**77.79 ±1.34**
GCN (RWSE)	84.84 ±0.55	46.40 ±0.55	**93.84 ±0.48**	**85.11 ±0.77**	**77.81 ±1.40**
GCN (DEG)	84.21 ±0.47	**50.01 ±0.69**	94.14 ±0.50	82.51 ±0.83	76.96 ±1.21
GAT (LapPE)	84.80 ±0.46	44.90 ±0.73	93.50 ±0.54	**84.99 ±0.54**	76.55 ±0.84
GAT (RWSE)	**86.62 ±0.53**	48.58 ±0.41	92.53 ±0.65	**85.02 ±0.67**	77.83 ±1.22
GAT (DEG)	85.51 ±0.56	**51.65 ±0.60**	93.04 ±0.62	84.22 ±0.81	77.10 ±1.23
$\text{GPS}^{\text{GCN+Performer}}$ (LapPE)	83.96 ±0.53	48.20 ±0.67	**93.85 ±0.41**	84.72 ±0.77	**77.85 ±1.25**
$\text{GPS}^{\text{GCN+Performer}}$ (RWSE)	84.72 ±0.65	48.08 ±0.85	92.88 ±0.50	84.81 ±0.86	76.45 ±1.51
$\text{GPS}^{\text{GCN+Performer}}$ (DEG)	83.38 ±0.68	48.93 ±0.47	93.60 ±0.47	80.49 ±0.97	74.24 ±1.18
$\text{GPS}^{\text{GAT+Performer}}$ (LapPE)	**85.93 ±0.52**	48.86 ±0.38	92.62 ±0.79	84.62 ±0.54	76.71 ±0.98
$\text{GPS}^{\text{GAT+Performer}}$ (RWSE)	**87.04 ±0.58**	49.92 ±0.68	91.08 ±0.58	84.38 ±0.91	77.14 ±1.49
$\text{GPS}^{\text{GAT+Performer}}$ (DEG)	85.54 ±0.58	**51.03 ±0.60**	91.52 ±0.46	82.45 ±0.89	76.51 ±1.19
$\text{GPS}^{\text{GCN+Transformer}}$ (LapPE)	OOM	OOM	91.82 ±0.41	83.51 ±0.93	OOM
$\text{GPS}^{\text{GCN+Transformer}}$ (RWSE)	OOM	OOM	91.17 ±0.51	83.53 ±1.06	OOM
$\text{GPS}^{\text{GCN+Transformer}}$ (DEG)	OOM	OOM	91.76 ±0.61	80.82 ±0.95	OOM
$\text{GPS}^{\text{GAT+Transformer}}$ (LapPE)	OOM	OOM	92.29 ±0.61	84.70 ±0.56	OOM
$\text{GPS}^{\text{GAT+Transformer}}$ (RWSE)	OOM	OOM	90.82 ±0.56	84.01 ±0.96	OOM
$\text{GPS}^{\text{GAT+Transformer}}$ (DEG)	OOM	OOM	91.58 ±0.56	81.89 ±0.85	OOM

For the five large datasets, we closely follow the hyper-parameter tuning and training setup described in (Platonov et al., 2023). To this end, we set the embedding size to 512 and only tune the number of layers (1, 2, 3, 4, and 5). Because on these datasets, the number of nodes provides an additional challenge for the transformer, we benchmark GPS also with the Performer module (Choromanski et al., 2022) to study the benefits of linear attention on large-scale graphs.

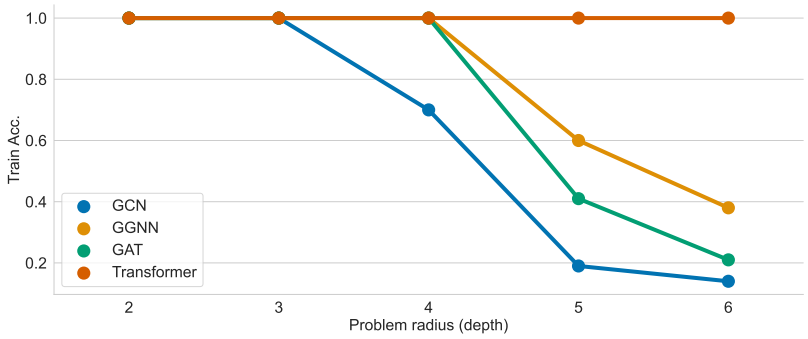

Figure 2: Average train accuracy over ten random seeds of a GT on the NeighborsMatch dataset, compared to models from Alon & Yahav (2021).

Apart from Graphormer, we benchmark all models on three different positional/structural encodings, namely LapPE, RWSE, and degree encodings as described in (Ying et al., 2021).

Answering Q2. On the small datasets (Table 3), all transformer-based models outperform a 2-layer GCN, and often the specialized Geom-GCN (Pei et al., 2020), which experimental setup we follow. With the exemption of node degree encodings (DEG), other PE/SE had minimal effect on GCN's performance. Adding global attention to the GCN, i.e., following the GPS model, universally improves the performance. Most interestingly, disabling the local GCN in GPS, i.e., becoming the Transformer model, increases the performance even further. Such results indicate that GNN-like models are unfit for these heterophilic datasets, while the global attention of a transformer empirically facilitates considerably more successful information propagation. Graphormer, which utilizes node degree encodings, performs comparably to the Transformer counterparts. Surprisingly, the attention bias of Graphormer had no impact on its performance. The shortest-path distance bias appears uninformative in these datasets, unlike, e.g., in Zinc, where we observed degradation from 0.12 test MAE to 0.54 when disabling the attention bias.

We conclude that we empirically confirm the expected benefits of global attention, albeit GTs do not achieve overall SOTA performance (e.g., see Luan et al. (2022)), which is a reminder that specialized architectures can achieve similar or higher performance without global attention still.

On the large datasets (Table 4), we observe strong benefits of added positional/structural encodings but also that their effect is highly dataset dependent. At the same time, an added Transformer or Performer module does not yield improvements in general. Specifically, the full Transformer goes out of memory on the three largest datasets while leading to poor performance on Minesweeper and competitive performance on Tolokers. Interestingly, the largest improvements of positional/structural encodings and Transformer modules are achieved on Roman-Empire, the only dataset with a large graph diameter (6824).

Here, we conclude that positional/structural information can lead to large gains in performance, while global attention only provides small or no improvements at all. One possible explanation is that when applied to graphs with 10K or more nodes, the self-attention of graph transformers is subjected to much more noise, which makes learning meaningful long-range interactions difficult.

4.3 Reduced Over-squashing in GTs?

To answer question **Q3**, we evaluate a GT on the NeighborsMatch problem proposed by Alon & Yahav (2021). This synthetic dataset requires long-range interaction between leaf nodes and the root node of a tree graph of depth d. The problem demonstrates GNNs' difficulty in transmitting information through a receptive field growing exponentially with d. We run our experiments with minimal changes to the code of Alon & Yahav (2021) and train our transformer on depths 2 to 6. Note that GNN models fail to perfectly fit the training set of trees with depth 4. Convergence on NeighborsMatch can sometimes take up to 100k epochs for large depths d. Since the structure of the graphs in NeighborsMatch is irrelevant to solving the

problem, we did not need to augment node features with positional/structural encodings or attention bias. Otherwise, we used the same architecture as in Section 4.1.

Answering Q3. Section 4.3 shows that the GT performs exceedingly better than the GNN baselines and can perfectly fit the training set even for depth $d = 6$. However, we note that the NeighborsMatch problem is idealized and has only limited practical implications. The core issue of over-squashing, which is squashing an exponentially growing amount of information into a fixed-size vector representation, is not resolved by transformers. Nonetheless, our results demonstrate that the ability of transformers to model long-range interactions between nodes can circumvent the problem posed by Alon & Yahav (2021).

5 Open Challenges and Future Work

Since the area of GTs is a new, emerging field, there are still many open challenges, both from practical and theoretical points of view. On the theoretical side, although it is often claimed that GTs offer better predictive performance over GNNs and are more capable of capturing long-range dependencies and preventing over-smoothing and over-squashing, a principled explanation still needs to be formed. Moreover, there needs to be a clearer understanding of the properties of structural and positional encodings. For example, it has yet to be understood when certain encodings are helpful and how they compare. The first step could be precisely characterizing different encodings in terms of distinguishing non-isomorphic graphs, similar to the Weisfeiler–Leman hierarchy (Morris et al., 2021). Further, understanding GTs generalization performance on larger graphs has yet to be understood on a similar level to GNNs (Yehudai et al., 2021; Zhou et al., 2022).

On the practical side, one major downside of GTs is their quadratic running time in the number of nodes, preventing them from scaling to truly large graphs typical in real-world node-level prediction tasks. Moreover, due to the attention mechanism's nature, it still needs to be determined how best to incorporate edge-feature information into GT architectures. Further, our experimental analysis reveals a disadvantage of local GNN-like models when used in conjunction with transformers as in Rampášek et al. (2022) on heterophilic datasets. Dealing with heterophilic data is thus an open challenge, also for GTs. Moreover, it is crucial to incorporate expert or domain knowledge, e.g., physical or chemical knowledge for molecular prediction, into the attention matrix in a principled manner.

Explaining and interpreting the performance of GTs remains an open research area where we draw parallels with NLP. We posit that studying graph transformers in the graph machine learning community follows a similar path of studying transformer language models in NLP unified under the name of *Bertology* (Rogers et al., 2021; Vulić et al., 2020). Numerous Bertology studies reported that more than dissecting attention matrices and attention scores in transformer layers is needed for understanding how language models work. Thereto, the community converged on designing datasets and tasks tailored to language model features such as co-reference resolution or mathematical reasoning. Therefore, we hypothesize that understanding GTs' performance through attention scores is limited, and the community should focus on designing diverse benchmarks probing particular GTs' properties. Further, studying scaling laws and emergent behavior of GTs and GNNs is still in its infancy, with few examples in chemistry (Frey et al., 2022) and protein representation learning (Lin et al., 2022).

6 Conclusion

We have provided a taxonomy of graph transformers (GTs). To this end, we overviewed GTs' theoretical properties and their connection to structural and positional encodings. We then thoroughly surveyed how GTs can deal with different input features, e.g., 3D information, and discussed the different ways of mapping a graph to a sequence of tokens serving as GTs' input. Moreover, we thoroughly discussed different ways GTs propagate information and recent real-world applications. Most importantly, we showed empirically that different encodings and architectural choices drastically impact GTs' predictive performance. We verified that GTs can deal with heterophilic graphs and prevent over-squashing to some extent. Finally, we proposed open challenges and outlined future work. We hope our survey presents a helpful handbook of graph transformers' methods, perspectives, and limitations and that its insights and principles will help spur and shape novel research results in this emerging field.

References

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *ICLR*, 2021.

Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *ICLR*, 2021.

Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature Reviews Genetics*, 5(2):101–113, 2004.

Pablo Barceló, Egor V. Kostylev, Mikael Monet, Jorge Pérez, Juan Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *ICLR*, 2020.

Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Lió. Directional graph networks. In *ICML*, pp. 748–758, 2021.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *ICLR*, 2023.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and Lehman go cellular: CW networks. *NeurIPS*, pp. 2625–2640, 2021.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *TPAMI*, 2022.

Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between MPNN and graph transformer. In *ICML*, pp. 3408–3430, 2023.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevni Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *JMLR*, pp. 1–64, 2022.

Chaoqi Chen, Yushuang Wu, Qiyuan Dai, Hong-Yu Zhou, Mutian Xu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. A survey on graph neural networks and graph transformers in computer vision: A task-oriented perspective. *ArXiv*, 2022a.

Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. *ICML*, 2022b.

Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. NAGphormer: A tokenized graph transformer for node classification in large graphs. In *ICLR*, 2023.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. In *NeurIPS*, pp. 15868–15876, 2019.

Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, and Thomas Weingarten. From block-Toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In *ICML*, 2022.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

CMU. World Wide Knowledge Base (Web-KB) project. `http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/`, 2001.

Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *Learning on Graphs Conference*, 2022.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 7865–7885. PMLR, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *ArXiv*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *ICLR*, 2022.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *JMLR*, 2023.

David Easley and Jon Kleinberg. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

Nathan Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael Gomez-Bombarelli, Connor Coley, and Vijay Gadepally. Neural scaling of deep chemical models. *chemRxiv*, 2022.

Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. SE(3)-Transformers: 3D roto-translation equivariant attention networks. In *NeurIPS*, 2020.

Fabian B. Fuchs, Edward Wagstaff, Justas Dauparas, and Ingmar Posner. Iterative SE(3)-Transformers. In *Geometric Science of Information*, pp. 585–595, 2021.

Johannes Gasteiger, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. *NeurIPS*, pp. 6790–6802, 2021.

Simon Geisler, Yujia Li, Daniel Mankowitz, Ali Taylan Cemgil, Stephan Günnemann, and Cosmin Paduraru. Transformers meet directed graphs. In *ICML*, 2023.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, pp. 1263–1272, 2017.

Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *TPAMI*, 2022.

Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of ViT/MLP-mixer to graphs. *ArXiv*, 2022.

Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv preprint*, 2016.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. In *NeurIPS: Datasets and Benchmarks Track*, 2021.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *WWW*, pp. 2704–2710, 2020.

Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *KDD*, pp. 655–665, 2022.

Paras Jain, Zhanghao Wu, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *NeurIPS*, 2021.

Chaitanya K. Joshi, Cristian Bodnar, Simon V Mathis, Taco Cohen, and Pietro Lio. On the expressive power of geometric graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pp. 15330–15355. PMLR, 2023.

Xuan Kan, Wei Dai, Hejie Cui, Zilong Zhang, Ying Guo, and Carl Yang. Brain network transformer. *ArXiv*, 2022.

Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. Interpretable rumor detection in microblogs by attending to user interactions. In *AAAI*, pp. 8783–8790, 2020.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *NeurIPS*, 2022.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In *NeurIPS*, pp. 4202–4212, 2019.

Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. GOAT: A global transformer on large-scale graphs. In *ICML*, 2023.

Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *NeurIPS*, 2021.

Weirui Kuang, Zhen Wang, Yaliang Li, Zhewei Wei, and Bolin Ding. Coarformer: Transformer for large graph via graph coarsening. *OpenReview*, 2022.

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. In *NeurIPS*, 2020.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pp. 3538–3545, 2018.

Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3D atomistic graphs. In *ICLR*, 2023.

Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *ArXiv*, 2022.

Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *ICCV*, pp. 12919–12928, 2021a.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *ArXiv*, 2021b.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. In *NeurIPS*, 2022.

Shengjie Luo, Tianlang Chen, Yixian Xu, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. One transformer can understand both 2D & 3D molecular data. *ArXiv*, 2022a.

Shengjie Luo, Shanda Li, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. Your transformer may not be as powerful as you expect. *ArXiv*, 2022b.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *ICML*, 2023.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, pp. 2153–2164, 2019.

Dominic Masters, Josef Dean, Kerstin Klaser, Zhiyi Li, Sam Maddrell-Mander, Adam Sanders, Hatem Helal, Deniz Beker, Ladislav Rampášek, and Dominique Beaini. GPS++: an optimised hybrid MPNN/transformer for molecular property prediction. *ArXiv*, 2022.

Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. GraphiT: Encoding graph structure in transformers. *ArXiv*, 2021.

Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *ArXiv*, 2022a.

Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Da Luo, Kangyi Lin, Junzhou Huang, Sophia Ananiadou, and Peilin Zhao. Neighbour interaction based click-through rate prediction via graph-masked transformer. In *SIGIR*, pp. 353–362, 2022b.

C. Morris, Y. L., H. Maron, B. Rieck, N. M. Kriege, M. Grohe, M. Fey, and K. Borgwardt. Weisfeiler and Leman go machine learning: The story so far. *ArXiv*, 2021.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, pp. 4602–4609, 2019.

Ryan L. Murphy, Balasubramaniam Srinivasan, Vinayak A. Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *ICML*, pp. 4663–4673, 2019.

Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, and Seung won Hwang. GRPE: Relative positional encoding for graph transformer. *ArXiv*, 2022.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *ICLR*, 2023.

Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. In *NeurIPS*, 2022.

Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *TACL*, 8:842–866, 2021.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Kristof Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *ICML*, pp. 9377–9388, 2021.

Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *ArXiv*, 2022.

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Scaling graph transformers with expander graphs. In *ICML*, 2023.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, pp. 29–38, 2017.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In *KDD*, 2009.

Philipp Thölke and Gianni De Fabritiis. Equivariant transformers for neural network based molecular potentials. In *ICLR*, 2022.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *ICLR*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pp. 5998–6008, 2017.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Ivan Vulić, Edoardo Maria Ponti, Robert Litschko, Goran Glavaš, and Anna Korhonen. Probing pretrained language models for lexical semantics. In *EMNLP*, pp. 7222–7240, 2020.

Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. In *ICLR*, 2022.

Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *NeurIPS*, 2022.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Keqiang Yan, Yi Liu, Yuchao Lin, and Shuiwang Ji. Periodic graph transformers for crystal material property prediction. In *NeurIPS*, 2022.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. Heterogeneous graph transformer for graph-to-sequence learning. In *ACL*, pp. 7145–7154, Online, 2020.

Gilad Yehudai, Ethan Fetaya, Eli A. Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In *ICML*, pp. 11975–11986, 2021.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *NeurIPS*, 2021.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big Bird: Transformers for longer sequences. In *NeurIPS*, 2020.

Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of GNNs via graph biconnectivity. In *ICLR*, 2023.

Yangze Zhou, Gitta Kutyniok, and Bruno Ribeiro. OOD link prediction generalization capabilities of message-passing GNNs in larger test graphs. In *NeurIPS*, 2022.

A Dataset details

Here, we describe the datasets used in our experiments; see Table 5 for an overview over the dataset statistics.

Edges We derive the Edges dataset from Zinc (Dwivedi et al., 2023). Zinc comprises 12K molecules from the ZINC database with heavy atoms as nodes and bonds as edges. The original task of Zinc is to predict the constrained solubility of a given molecule. Note that we ignore these regression targets in our link prediction task.

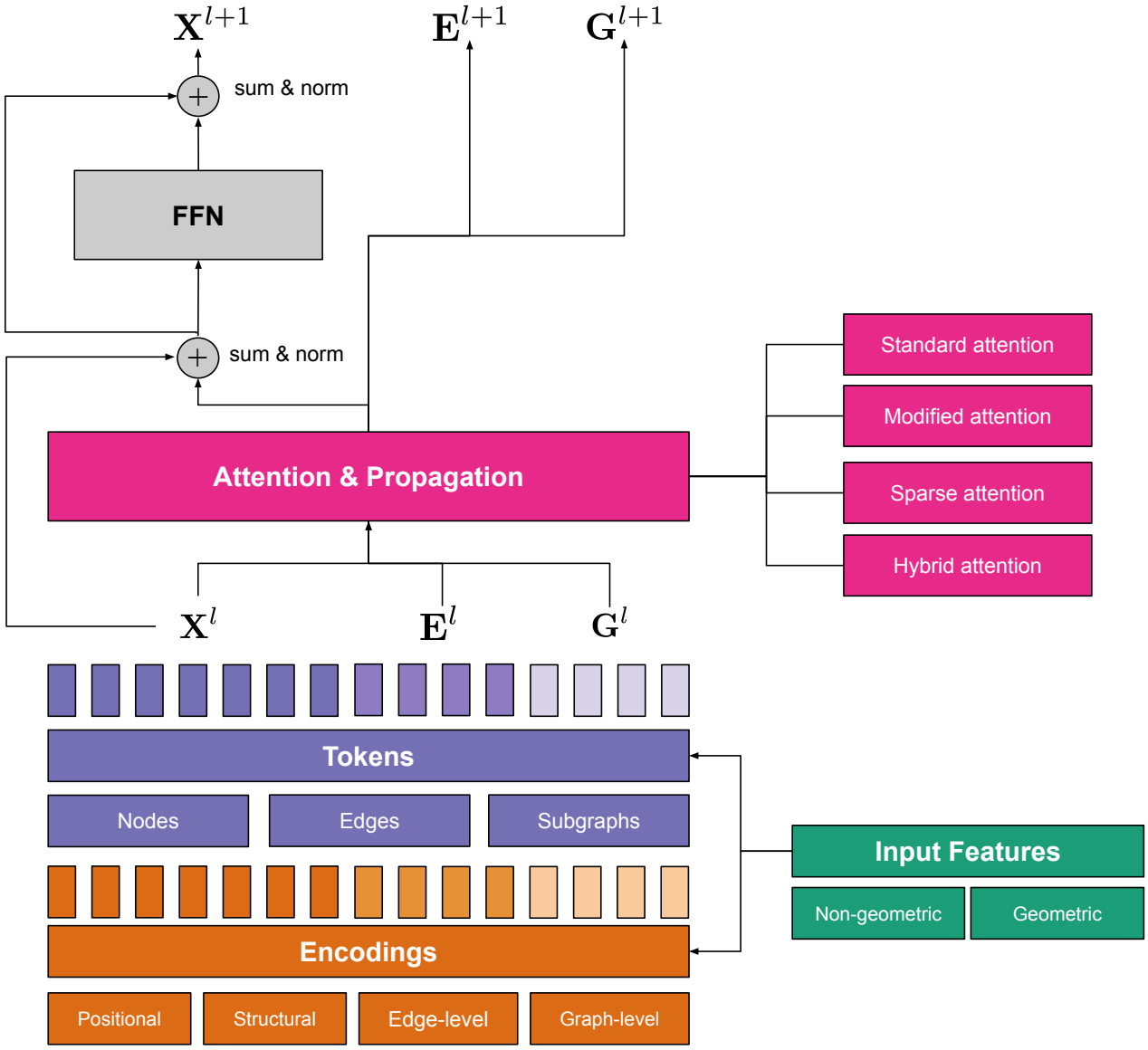

Figure 3: Overview of how the different branches of our taxonomy affect the original transformer architecture.

Triangles The TRIANGLES dataset contains synthetic graphs with the goal of predicting the number of triangle subgraphs contained in a given graph (Knyazev et al., 2019).

CSL The CSL dataset contains synthetic regular graphs, i.e., graphs where each node has the same number of neighbors (Dwivedi et al., 2023). The nodes of each graph initially form a cycle and are then additionally equipped with additional edges, so called skip links, connecting nodes on the cycle that are some fixed length L apart. Different graphs are constructed for different skip link lengths L and the task of CSL is to classify L.

Actor The ACTOR dataset is derived from the film-director-actor-writer network, a heterogeneous graph resulting from crawled Wikipedia pages and proposed in Tang et al. (2009). Specifically, ACTOR is a graph where actors are the nodes and two actors are connected by an edge if their names occur on the same Wikipedia page. The task is to classify the actors in to one of five most occuring categories in the category information on the respective actor's Wikipedia page.

Cornell, Texas, Wisconsin The CORNELL, TEXAS and WISCONSIN are three of the four webpage networks collected in WebKB (CMU, 2001). Each network contains crawled webpages as nodes and hyperlinks connecting the pages as edges. The task is to predict which category of student, project, course, staff and faculty a webpage belongs to.

Chameleon, Squirrel The CHAMELEON and SQUIRREL are datasets constructed from Wikipedia pages of their respective topic (that of Chameleons and that of Squirrels, respectively), where the nodes are articles and edges represent links from one article to another (Rozemberczki et al., 2021). The node features are based on the occurrence of particular nouns on the respective article and the task to predict the average monthly traffic on the site.

Roman-Empire The ROMAN-EMPIRE dataset is based on the Wikipedia article of the Roman Empire (Platonov et al., 2023), where nodes are the words occurring in the article and an edge between two words indicates that either the words follow each other in a sentence or if one word syntactically depends on the other . The task is predict the syntactic role of the work according to spaCy (Honnibal et al., 2020).

Amazon-Ratings The AMAZON-RATINGS dataset is derived from Amazon product co-purchasing metadata (Platonov et al., 2023), where the nodes are products and an edge between two products indicates that the products are frequently bought together. The task is to predict the average rating of the product from one to five stars.

Minesweeper The MINESWEEPER dataset is constructed from a grid graph, where each node represents a cell in a Minesweeper game and the task is to predict for each cell whether it contains a mine (Platonov et al., 2023).

Tolokers The TOLOKERS dataset contains data from the Toloka crowd-sourcing platform (Platonov et al., 2023). Nodes represent workers and an edge between two workers indicates that the workers have worked together on one of 13 selected projects. The task is to predict which workers have been banned from a project.

Questions The QUESTIONS dataset contains data from a question-answering wbesite (Platonov et al., 2023). Nodes represent users answering questions on "medicine" and an edge between two users indicates that the users have answered the same questions within some fixed time span. The task is to predict which users remained active at the end of the time span.

B Experimental details

Here, we describe the details of our experiments. All experiments were run on a single A100 NVIDIA GPU with 80GB of GPU RAM.

Dataset	Num. graphs	Num. nodes	Num. edges	Task	Metric
Edges	12,000	277,864	597,970	Link prediction	Cross entropy
Triangles	45,000	938,438	2,947,024	10-way graph classification	Cross entropy
CSL	150	6,150	24,600	10-way graph classification	Cross entropy
Actor	1	7,600	30,019	5-way node classification	Cross entropy
Cornell	1	183	298	5-way node classification	Cross entropy
Texas	1	183	325	5-way node classification	Cross entropy
Wisconsin	1	251	515	5-way node classification	Cross entropy
Chameleon	1	2,277	36,101	5-way node classification	Cross entropy
Squirrel	1	5,201	217,073	5-way node classification	Cross entropy
Roman-Empire	1	22,662	32,927	18-way node classification	Cross entropy
Amazon-Ratings	1	24,492	93,050	5-way node classification	Cross entropy
Minesweeper	1	10,000	39,402	2-way node classification	Cross entropy
Tolokers	1	11,758	519,000	2-way node classification	Cross entropy
Questions	1	48,921	153,540	2-way node classification	Cross entropy

Table 5: Statistics of the datasets used in our experiments (Dwivedi et al., 2023; Knyazev et al., 2019; Tang et al., 2009; CMU, 2001; Rozemberczki et al., 2021; Platonov et al., 2023)

We base our implementation on GraphGPS (Rampášek et al., 2022), which is available at `https://github.com/rampasek/GraphGPS`, which also includes an implementation of Graphormer (Ying et al., 2021), except for the over-squashing experiment, where we use the implementation by Alon & Yahav (2021) to stay as close as possible to the original implementation. All model layers first apply a convolution/attention, followed by a feed-forward network. The resulting embeddings are then fed into a final MLP head to make a prediction. For all models we use a GELU non-linearity (Hendrycks & Gimpel, 2016) in the feed-forward network. Further, we apply residual connections for all models, one after the convolution/attention and one after the feed-forward network.

For the structural awareness experiments in Section 4.1, we follow the hyper-parameters detailed in Table 1. In addition, for all models, we use 6 layers of convolution/attention. For the GIN, Transformer and GPS we use batch normalization, mean pooling and three layers for the final MLP head. For Graphormer we use layer normalization, `[graph]` token readout and a linear layer, preceded by a final layer norm for the final MLP head, following (Ying et al., 2021).

For the six small datasets Actor, Cornell, Texas, Wisconsin, Chameleon and Squirrel, we select hyper-parameters with a grid search over the hidden dimension (32, 64, 96), dropout (0.0, 0.2, 0.5, 0.8) and, where applicable, attention dropout (0.0, 0.2, 0.5). For the grid search we repeat each experiments 10 times and select the hyper-parameters leading to the best average validation performance. In all models, we use 2 layers of convolution/attention and a linear layer for the final MLP head. We use batch normalization for the GCN, GPS and Transformer and layer normalization for Graphormer. Again, for Graphormer, we apply a final layer norm before the final MLP head.

For the five large datasets in Platonov et al. (2023), we follow their hyper-parameter selection exactly to enable a fair comparison. As a result, we only tune the number of layers (1,2,3,4,5). We use sum pooling and a linear layer for the final MLP head.