Subgraphormer: Subgraph GNNs meet Graph Transformers

Guy Bar-Shalom
Department of Computer Science
Technion - Israel Institute of Technology
guy.b@cs.technion.ac.il

Beatrice Bevilacqua
Department of Computer Science
Purdue University
bbevilac@purdue.edu

Haggai Maron
Department of Electrical & Computer Engineering
Technion - Israel Institute of Technology, Nvidia Research
haggaimaron@ef.technion.ac.il

Abstract

In the realm of Graph Neural Network (GNNs), two intriguing research directions have recently emerged: Subgraph GNNs and Graph Transformers. These approaches have distinct origins – Subgraph GNNs aim to address the limitations of message passing, while Graph Transformers seek to build on the success of sequential transformers in language and vision tasks. In this paper, we propose a model that integrates both approaches, dubbed `Subgraphormer`, which combines the message passing and global aggregation schemes from Subgraph GNNs with attention mechanisms and positional and structural encodings, which are arguably the most important components in Graph Transformers. Our preliminary experimental results demonstrate significant performance improvements over both Subgraph GNNs and Graph Transformers.

1 Introduction

Due to their scalability and elegant architectural design, Message Passing Neural Networks (MPNNs) have emerged as the gold standard type of Graph Neural Networks (GNNs) for processing graph data. Nonetheless, message passing suffers from a set of limitations, including limited expressive power [23, 35], and issues like over-squashing [1] and over-smoothing [31]. Over the past few years, multiple GNN-based architectures have been proposed to mitigate these problems.

In this paper, we focus on two different enhanced GNN architectures: Subgraph GNNs and Graph Transformers. In Subgraph GNNs [39, 7, 41, 2], an MPNN is applied to a bag (multiset) of subgraphs, which is generated from the original graph (for example the set of subgraphs obtained by deleting one node in the original graph). Notably, these subgraph-based architectures are provably more expressive than the traditional Message Passing (MP) algorithms applied directly to the original graph. In parallel, Transformers [32] have demonstrated outstanding performance across a wide range of applications, including natural language processing (NLP) [32, 13, 15], computer vision [14, 8, 18], and, more recently, graph-based tasks [36, 38, 29]. In particular, Graph Transformers (GTs) have achieved impressive empirical results, as evidenced in tasks like molecular prediction [17]. As demonstrated by Müller et al. [24], these achievements are often attributed to the GTs' ability to overcome some of the inherent limitations of traditional GNNs, such as the issues of over-smoothing [31] and over-squashing [1] mentioned earlier.

In an effort to design a model that enjoys the benefits of both architectural paradigms, we introduce `Subgraphormer`, a transformer-based architecture that combines Graph Transformers with subgraph methodologies; To the best of our knowledge, this represents the first attempt to harmonize these two paradigms. To elaborate, we construct a bag of subgraphs and implement attention-based aggregation

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

methods, thereby allowing individual nodes to refine their representations by selectively attending to particular nodes from different subgraphs. More specifically, we incorporate the inductive biases that are intrinsic to subgraph-based aggregations. Additionally, we propose a subgraph-based positional and structural encoding scheme, enriching each node with information from subgraphs.

Our preliminary experiments confirm that our architecture delivers performance improvements on the ZINC12k dataset [30, 12, 9], achieving state-of-the-art results. Furthermore, we address the potential computational burden of operating on extensive bags of subgraphs, demonstrating that the performance of stochastic bag sampling improves dramatically when using `Subgraphormer` compared to other Subgraph GNNs.

To summarize, the contributions of this paper are: (1) `Subgraphormer`, a novel architecture which combines the strengths of both transformer-based and subgraph-based methodologies; (2) A positional and structural encoding scheme tailored to subgraphs, enabling each node to integrate information from multiple subgraphs; and (3) An empirical study demonstrating significant improvements of `Subgraphormer` compared to existing baselines in both full bag and stochastic bag sampling setups.

2 Previous Work and Preliminaries

Notation. Let $G = (A, X)$ denote an undirected graph, which belongs to the family $\mathcal{G}$ of finite, simple, node-attributed graphs[1]. The adjacency matrix $A \in \mathbb{R}^{n \times n}$ represents the graph connectivity while the feature matrix, $X \in \mathbb{R}^{n \times d}$, maintains the node features. We denote the sets of nodes and edges as V and E, respectively. Let $\mathcal{B}_G$ be the set of all possible bags of subgraphs derived from G. For a given multiset (or bag) $B_G \in \mathcal{B}_G$ comprising S subgraphs of G, the adjacency and feature matrices associated with the subgraphs in B_G are structured as tensors: $\mathcal{A} \in \mathbb{R}^{S \times n \times n}$ and $\mathcal{X} \in \mathbb{R}^{S \times n \times d}$, respectively; we define $\mathcal{T}_{\mathcal{A}} \subseteq \mathbb{R}^{S \times n \times n}$ to be the set of all possible such adjacency tensors. We use the superscript notation ${}^{(t);s}$ to denote subgraph s at layer t, while the underscore notation ${}_v$ represents the v-th node; e.g., $x_v^{(0);s}$ denotes the feature of node v in subgraph s, at the 0-th layer.

2.1 Subgraph GNNs

Subgraph GNNs represent a graph as a multiset of subgraphs and then process it using a permutation-equivariant architecture. The bag of subgraphs is generated through the application of a predefined selection policy to the original graph. While several selection polices have been proposed [2], in this paper we focus on node-based policies [11], in which each subgraph is associated with a unique node in the graph. For example, the simplest implementation of the node-marking (NM) policy generates the i-th subgraph in the bag by copying the original graph and marking its i-th node. Since there is a bijection between nodes and subgraphs, then subgraph i is associated to node i, which we will refer to as the root of the subgraph. These subgraphs are subsequently processed using an architecture that applies GNNs to the subgraphs, and in some cases, shares information between them.

Although extensive research has been conducted on Subgraph GNNs [39, 7, 26, 41, 25, 11, 28] our method draws particular inspiration from Zhang et al. [37], which constructed a comprehensive hierarchy of Subgraph Weisfeiler-Lehman Tests (SWL). The authors proposed a corresponding family of Subgraph GNN architectures that leverage various atomic operations: single-point, global, and local aggregations. Let $N_{G(v)}$ denote the neighbours of v in a graph G, these operations are formally defined in Equations (1), (2), and (3), respectively, as follows,

$$\begin{aligned}
&\text{Single-point: } x_v^{(t+1);s} \leftarrow x_v^{(t);s} \text{ or } x_v^{(t+1);s} \leftarrow x_s^{(t);v} \\
&\qquad \text{or } x_v^{(t+1);s} \leftarrow x_s^{(t);s} \text{ or } x_v^{(t+1);s} \leftarrow x_v^{(t);v},
\end{aligned} \tag{1}$$

$$\text{Global: } x_v^{(t+1);s} \leftarrow \sum_{s'=1}^{S} x_v^{(t);s'} \text{ or } x_v^{(t+1);s} \leftarrow \sum_{v'=1}^{V} x_{v'}^{(t);s}, \tag{2}$$

$$\text{Local: } x_v^{(t+1);s} \leftarrow \sum_{v' \in N_{G(v)}} x_{v'}^{(t);s} \text{ or } x_v^{(t+1);s} \leftarrow \sum_{s' \in N_{G(s)}} x_v^{(t);s'}. \tag{3}$$

[1]The consideration of edge-features is omitted for simplicity.

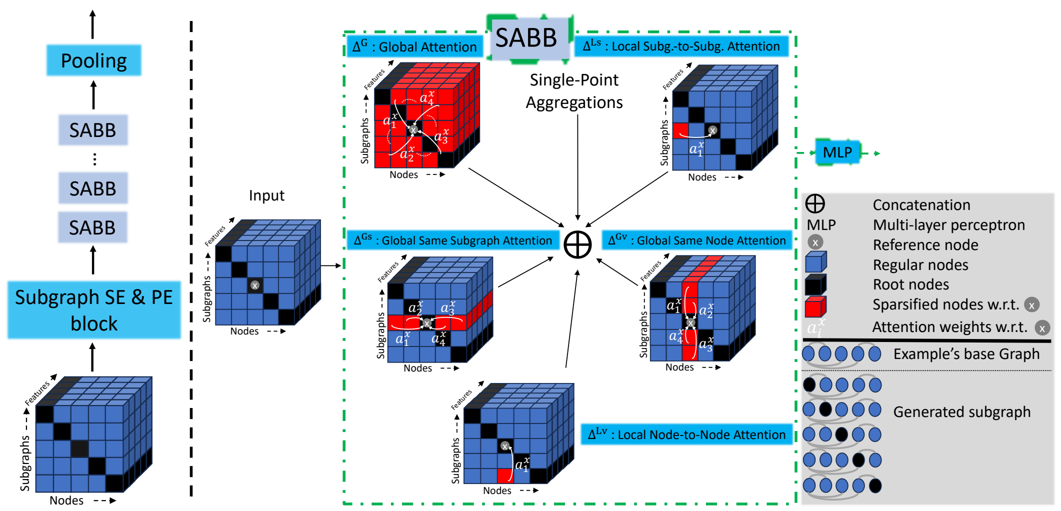

Figure 1: On the left, the complete architecture; on the right, the Subgraph Attention-Based Block (SABB) composed of both attention-based aggregations and single-point aggregations – we visualize each of the attention-based aggregations. A specific graph is selected for visualization, as indicated in the right legend (bottom part). We show each sparsified tensor related to the sparsification function Δ (red cubes represent the nodes selected by the sparsification function Δ) w.r.t. a reference node, denoted by a white 'X' (refer to the legend for details).

In the equations above, indices associated with s and v, are rendered in red and blue, respectively. As demonstrated by Zhang et al. [37], employing these aggregation methods or even just a specific subset of them, results in the most expressive Subgraph GNN proposed to date, surpassing the expressive capability of the 1-WL test.

2.2 Graph Transformers

Transformers have achieved notable success in both natural language processing (NLP) [32, 13, 15] and computer vision [14, 8, 18]. The success of transformers can be largely attributed to their key component, the attention mechanism [32]. Building on the success of transformers, researchers have recently introduced Graph-Transformers (GTs), a specialized adaptation of the transformer architecture tailored for graph-structured data. In particular, Graphormer [36] and Graphormer-GD [38], effectively integrates graph-induced biases into the attention mechanism. Rampášek et al. [29] proposed GPS, a hybrid architecture that employs both MP and transformer-based layers. Additionally, K-Subgraph SAT [6] enhances the self-attention mechanism with structural awareness. Specifically, it integrates structural information into the original self-attention mechanism by first extracting a subgraph representation rooted at each node.

3 Subgraphormer

Our architecture, `Subgraphormer`, is composed of a structural and positional encoding layer, a stacking of Subgraph Attention-Based Blocks (SABBs), and a pooling layer, as illustrated in Figure 1 (left). In the following, we describe these building blocks in detail.

3.1 Subgraph Transformer Architecture

In this subsection, we introduce the structure of the Subgraph Attention-Based Blocks (SABBs), designed to update the representation of each node v within every subgraph s. In every SABB, we leverage a combination of single-point aggregations and attention-based aggregations. The SABB is visualized in Figure 1 (right).

Overview. Within each SABB, a node's representation is updated by considering a list of possible single-point and attention-based aggregations. For each attention-based aggregation, only specific nodes from the subgraphs within B_G influence the update equation for the given node, while others are disregarded. As we elaborate later, the criteria for selecting these subsets are determined by a sparsification function Δ, chosen from a predefined set of functions. Different choices for Δ enable attention mechanisms that operate on distinct subsets of nodes. For example, employing Δ^{Ls} for the update operation effectively sparsifies the graph, and the representation of the targeted node is updated based solely on its immediate neighbors in the original graph. Conversely, Δ^{Gs} allows updates of the node's representation by considering all nodes within the same subgraph. Figure 1 (right) contains a visual representation of all the different Δs. The attention coefficient function α, defined in Equation (7), quantifies the strength of the attention between nodes, serving as a weighting factor in the update equation, determining the influence of different nodes on each other. Together, Δ and α offer a flexible framework in modeling node interactions. In what follows, we provide a rigorous mathematical formulation of our architecture. We begin by discussing single-point aggregations, followed by attention-based aggregations; a complexity analysis is provided in Appendix A.2, Table 3.

Single-point aggregations. We adopt the single-point aggregations from Zhang et al. [37], $\text{Ag}_{\text{sv}}(\mathcal{X}, s, v) \triangleq x_v^s$, $\text{Ag}_{\text{vs}}(\mathcal{X}, s, v) \triangleq x_s^v$, $\text{Ag}_{\text{ss}}(\mathcal{X}, s, v) \triangleq x_s^s$, $\text{Ag}_{\text{vv}}(\mathcal{X}, s, v) \triangleq x_v^v$.

For a given instance of single-point aggregation, denoted by Ag^P and chosen from the set $\{\text{Ag}_{\text{sv}}, \text{Ag}_{\text{vs}}, \text{Ag}_{\text{ss}}, \text{Ag}_{\text{vv}}\}$, we employ a GIN base encoder [35], that is,

$$x_v^{(t+1);s;\text{Ag}^P} = (1 + \epsilon^{(t+1;\text{Ag}^P)})x_v^{(t);s} + \text{Ag}^P, \tag{4}$$

Attention-Based Aggregations. In the following, we present the attention-based aggregations within the SABBs. Each attention component serves to update a node v within a specific subgraph s. This update is performed with respect to a distinct subset of nodes contained in the subgraphs within the bag B_G. More precisely, each node index pair (s, v) attends to a subset of nodes, determined by a sparsification function $\Delta \in \mathcal{D}$, defined as follows:

$$\Delta : V^2 \times V^2 \times \mathcal{T}_{\mathcal{A}} \times \mathcal{G} \to \{0, 1\}, \tag{5}$$

where by $\mathcal{D}$ we denote the set of all possible sparsification functions. The function Δ takes four inputs: two tuples of indices, the adjacency matrices of the subgraphs, $\mathcal{A}$, and the original graph, G. It returns either 0 or 1, where 0 signifies that attention should *not* be applied between the nodes, and 1 signifies that it *should*. This function essentially determines which nodes should participate in the attention mechanism of a given node in a given subgraph.

We provide five specific instantiations of Δ allowing various attention mechanisms tailored for different scenarios, all visualized in Figure 1 (right). In particular, the *Global Same Subgraph Attention* directs attention exclusively within individual subgraphs; it "turns on" attention only when nodes belong to the same subgraph. Conversely, the *Global Same Node Attention* prompts nodes to attend to their own representations across different subgraphs. Additionally, we introduce locality-sensitive mechanisms. The *Local Subgraph-to-Subgraph Attention* permits nodes within the same subgraph to attend to each other, but only if they are also neighbors in the original graph G. Similarly, the *Local Node-to-Node Attention* allows for inter-subgraph attention, but it is restricted to nodes in subgraphs whose root nodes are neighbors in G. Lastly, we introduce a *Global Attention* mode, which disregards subgraph and node identities, enabling attention across all nodes in all subgraphs.

Each of these attention mechanisms is represented by a specific instance of the sparsification function Δ, as formalized below:

$$\Delta \in \{\Delta^{\text{Gs}}, \Delta^{\text{Gv}}, \Delta^{\text{Ls}}, \Delta^{\text{Lv}}, \Delta^{\text{G}}\}, \tag{6}$$

where,

$$\text{Global Same Subgraph Attention: } \Delta^{\text{Gs}}\Big((s, v), (s', v'), \mathcal{A}, G\Big) = \delta_{ss'}$$

$$\text{Global Same Node Attention: } \Delta^{\text{Gv}}\Big((s, v), (s', v'), \mathcal{A}, G\Big) = \delta_{vv'}$$

$$\text{Local Subg.-to-Subg. Attention: } \Delta^{\text{Ls}}\Big((s, v), (s', v'), \mathcal{A}, G\Big) = \begin{cases} \delta_{ss'} & \text{if } v \text{ and } v' \text{ are neighbors in } G, \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Local Node-to-Node Attention: } \Delta^{\text{Lv}}\Big((s, v), (s', v'), \mathcal{A}, G\Big) = \begin{cases} \delta_{vv'} & \text{if } s \text{ and } s' \text{ are neighbors in } G, \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Global Attention: } \Delta^{\text{G}}\Big((s, v), (s', v'), \mathcal{A}, G\Big) = 1$$

By δ we are referring to the Kronecker delta.

We introduce the *attention coefficient function* α, defined as,

$$\alpha : \mathcal{D} \times \mathbb{R}^d \times \mathbb{R}^d \times V^2 \times V^2 \times \mathcal{B} \times \mathcal{G} \to \mathbb{R}^+. \tag{7}$$

The function α takes as input: an attention sparsification function Δ from the set $\mathcal{D}$, two vectors representing the features of the nodes under consideration and their corresponding indices, a bag of subgraphs $B_G \in \mathcal{B}$, and the original graph $G \in \mathcal{G}$. Based on these inputs, α calculates the normalized attention score between the nodes. Precisely, if Δ yields a value of 0, then α also returns 0, indicating that no attention should be applied. On the contrary, if Δ returns 1, α computes attention scores for the corresponding elements and normalizes them using a softmax function. Our specific implementation of α is inspired by GATv2 [5]; for further details, see Appendix A.

Given a source node x_v^s, and a specific sparsification function Δ, its new representation is computed via the following equation:

$$x_v^{(t+1),\Delta;s} = \sum_{s'=1}^{S} \sum_{v'=1}^{n} \alpha^{(t)} \Big(\Delta, \Big(x_v^{(t);s}, x_{v'}^{(t);s'} \Big), \Big((s,v), (s',v') \Big), B_G, G \Big) \cdot F_V \Big(x_{v'}^{(t);s'} \Big), \tag{8}$$

where $F_V : \mathbb{R}^d \to \mathbb{R}^d$ is a value transformation.

In each SABB, the final representation of a node is determined by applying a MLP to the concatenation of the four single-point aggregations, and the five attention-based aggregations, as illustrated in Figure 1 (right).

Pooling. The final pooling layer, ρ is implemented as follows:

$$\rho(B_G^{(T)}) = \frac{1}{S} \sum_{s=1}^{S} \texttt{MLP} \left(\sum_{v=1}^{N} x_v^{(T);s} \right), \tag{9}$$

where the superscript T refers to the representations at the final layer, when $t = T$.

3.2 Positional and Structural Encodings for Subgraph Transformers

Previous works have demonstrated the effectiveness of both positional and structural encodings in various settings [10, 34, 19, 37], and specifically in the context of GTs [29, 27]. In the following, we provide a formal definition of our Subgraph Structural Encoding (SE) and Positional Encoding (PE) block, as depicted in Figure 1 (left).

Node-marking. We employ the node-marking strategy used in Zhang et al. [37]. In particular, a special mark is added to each node x_v^s, as follows:

$$x_v^{(0);s;NM} \leftarrow x_v^{(0);s} + z_{\text{dist}(s,v)}. \tag{10}$$

The vector $z \in \mathbb{R}^d$ is a learnable embedding specified by the shortest path $z_{\text{dist}}(s, v)$ between the *root* node s of subgraph s, and any node v within that same subgraph[2].

Positional Encoding for Graphs. Positional encodings for graphs aim to represent the relative positions of nodes within a graph. To establish a suitable positional encoding in the context of subgraphs, we build upon the widely recognized Graph Laplacian, $L := D - A = U^T \Lambda U$, where A represents the **original** adjacency matrix of G, $D_{ii} := \sum_{j=1}^{n} A_{ij}$, and Λ, U denote respectively to the eigenvalues and eigenvectors of L. Let $\mathbf{p}_i := [U_{i1}, \ldots, U_{ik}]$ (k is a hyperparameter), then, using the node embeddings of both root node and current node, our positional encoding is defined as follows[3]:

$$x_v^{(0);s;PE} \leftarrow \mathbf{W_1^{PE}} \text{LeakyReLU}\big(\mathbf{W_2^{PE}} [\mathbf{p}_s \oplus \mathbf{p}_v] \big). \tag{11}$$

Structural Encoding for Graphs. Structural encoding (SE) techniques have proven highly effective in enhancing the performance of GNNs, as demonstrated in Ma et al. [22]. The fundamental idea involves augmenting each node in the graph with additional information that enhances its knowledge of the surrounding structural context. In our work, we make use of the random walk operator utilized in previous research [10, 20] for constructing SE for graphs. This operator is formally defined

[2]We assign a unique mark if the two nodes are unreachable from each other, e.g., dist$(s, v) = \infty$.

[3]Bias weights are also included but omitted here for simplicity

Table 1: Performance comparison for different architectures. Transformer-based and Subgraph-based architectures are in gray and light blue, respectively. Best method is bolded.

Model	ZINC (Test MAE ↓)
GSN [4]	0.101 ± 0.010
CIN (small) [3]	0.094 ± 0.004
GIN [35]	0.252 ± 0.017
SAN [16]	0.139 ± 0.006
URPE [21]	0.086 ± 0.007
GPS [29]	0.070 ± 0.004
Graphormer [36]	0.122 ± 0.006
Graphormer-GD [38]	0.081 ± 0.009
K-Subgraph SAT [6]	0.094 ± 0.008
NGNN [20]	0.111 ± 0.003
SUN [11]	0.083 ± 0.003
ESAN [2]	0.102 ± 0.003
OSAN [28]	0.154 ± 0.008
GNN-AK [40]	0.105 ± 0.010
GNN-AK+ [40]	0.091 ± 0.002
GNN-SSWL [37]	0.082 ± 0.003
GNN-SSWL+ [37]	0.070 ± 0.005
Subgraphormer	0.064 ± 0.001
Subgraphormer + SE	0.066 ± 0.003
Subgraphormer + PE	$\mathbf{0.062 \pm 0.002}$
Subgraphormer + SE + PE	0.067 ± 0.002

Table 2: Results for the stochastic sampling approach, where each model sees 100%, 50%, 20%, and 5% of subgraphs for each graph, uniformly sampled at each training epoch. Best method for each percentage is bolded.

Model		ZINC (Test MAE ↓)
ESAN [2]	(100%)	0.102 ± 0.003
ESAN [2]	(50%)	0.155 ± 0.007
ESAN [2]	(20%)	0.166 ± 0.005
ESAN [2]	(5%)	0.179 ± 0.001
Subgraphormer	(100%)	0.064 ± 0.001
Subgraphormer	(50%)	$\mathbf{0.079 \pm 0.050}$
Subgraphormer	(20%)	0.129 ± 0.010
Subgraphormer	(5%)	0.217 ± 0.008
Subgraphormer + SE	(100%)	0.065 ± 0.002
Subgraphormer + SE	(50%)	0.081 ± 0.005
Subgraphormer + SE	(20%)	0.121 ± 0.014
Subgraphormer + SE	(5%)	$\mathbf{0.143 \pm 0.001}$
Subgraphormer + PE	(100%)	$\mathbf{0.062 \pm 0.002}$
Subgraphormer + PE	(50%)	0.082 ± 0.005
Subgraphormer + PE	(20%)	0.130 ± 0.003
Subgraphormer + PE	(5%)	0.227 ± 0.012
Subgraphormer + SE + PE	(100%)	0.067 ± 0.002
Subgraphormer + SE + PE	(50%)	0.081 ± 0.006
Subgraphormer + SE + PE	(20%)	$\mathbf{0.114 \pm 0.005}$
Subgraphormer + SE + PE	(5%)	0.164 ± 0.005

as, RW $:= D^{-1}A$, where A and D denote the adjacency and degree matrices of a given graph G, respectively.

To adapt SE to subgraphs, we take the following steps. First, we construct the RW operator corresponding to the **original** graph G. Then, we define a random walk vector $\mathbf{r}_i$ for each node i as $\mathbf{r}_i = [\text{RW}_{ii}, \text{RW}^2_{ii}, \dots, \text{RW}^{k'}_{ii}]$. Finally, we obtain the the structural encoding for subgraphs based on these random walk vectors as follows,

$$x_v^{(0);s;SE} \leftarrow \mathbf{W}_{\mathbf{1}}^{\text{SE}} \text{LeakyReLU}\left(\mathbf{W}_{\mathbf{2}}^{\text{SE}} \left[\mathbf{r}_s \oplus \mathbf{r}_v\right]\right). \tag{12}$$

The three vectors $x_v^{(0);s;NM}, x_v^{(0);s;PE}, x_v^{(0);s;SE}$ are then concatenated and passed through an `MLP` with one hidden layer, along with a residual connection with $x_v^{(0);s;NM}$. Notably, the framework is also flexible to support cases where only PE or SE is used, or even cases where neither is applied. The output of this SE & PE block serves as the input to the SABB, as depicted in Figure 1 (left).

4 Experiments

In this section, we conduct a preliminary evaluation of `Subgraphormer` using the ZINC12k [30, 12, 9] molecular benchmark. To assess the effectiveness of our `Subgraphormer` model, we compare it against several baselines, including the two natural ones, Graph Transformers (GTs) and Subgraph GNNs. We also analyze the performance of our model when using stochastic sampling.

Table 1 clearly demonstrates that all variations of our `Subgraphormer` model improve over all baselines, including both subgraph-based architectures and GTs, by a large margin. More specifically, `Subgraphormer` + PE achieves the best *mean absolute error* (MAE) – 0.062, a significant improvement of 0.008 from the best baseline, namely, GNN-SSWL+, which achieved a MAE of 0.070. Notably, this improvement holds with a statistical significance within a 1-σ error bar.

In Table 2, we evaluate the performance of our model when utilizing only a subset of the subgraphs; Due to space constraints, we refer the reader to Appendix A.1 for an in-depth explanation of this variation for our model. By employing stochastic sampling to select a subset of the subgraphs, we balance performance with complexity. Our model is benchmarked against ESAN [2], which also supports stochastic sampling. Impressively, even when utilizing a reduced subset of subgraphs, both `Subgraphormer` + SE and `Subgraphormer` + SE + PE variants consistently outperform ESAN across all sampling percentages: $\{5\%, 20\%, 50\%, 100\%\}$. This observation underscores the significant role played by structural and positional encodings in compensating for the limited subgraph usage. Notably, all variants of `Subgraphormer` demonstrate a significant performance leap, requiring only 50% of the subgraphs to surpass the full capacity performance of ESAN (100%).

Acknowledgments

HM is the Robert J. Shillman Fellow, and is supported by the Israel Science Foundation through a personal grant (ISF 264/23) and an equipment grant (ISF 532/23).

References

[1] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

[2] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. *arXiv preprint arXiv:2110.02910*, 2021.

[3] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

[4] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. arxiv 2020. *arXiv preprint arXiv:2006.09252*, 2006.

[5] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[6] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[7] Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[10] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

[11] Fabrizio Frasca, Beatrice Bevilacqua, Michael Bronstein, and Haggai Maron. Understanding and extending subgraph gnns by rethinking their symmetries. *Advances in Neural Information Processing Systems*, 35:31376–31390, 2022.

[12] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[13] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542*, 2021.

[14] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54 (10s):1–41, 2022.

[15] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[16] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[17] Han Li, Dan Zhao, and Jianyang Zeng. Kpgt: knowledge-guided pre-training of graph transformer for molecular property prediction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 857–867, 2022.

[18] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4804–4814, 2022.

[19] 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 preprint arXiv:2202.13013*, 2022.

[20] Renming Liu, Semih Cantürk, Olivier Lapointe-Gagné, Vincent Létourneau, Guy Wolf, Dominique Beaini, and Ladislav Rampášek. Graph positional and structural encoder. *arXiv preprint arXiv:2307.07107*, 2023.

[21] Shengjie Luo, Shanda Li, Shuxin Zheng, Tie-Yan Liu, Liwei Wang, and Di He. Your transformer may not be as powerful as you expect. *Advances in Neural Information Processing Systems*, 35: 4301–4315, 2022.

[22] 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. *arXiv preprint arXiv:2305.17589*, 2023.

[23] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[24] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampášek. Attending to graph transformers. *arXiv preprint arXiv:2302.04181*, 2023.

[25] Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. *CoRR*, abs/2201.12884, 2022.

[26] Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34:21997–22009, 2021.

[27] Wonpyo Park, Woonggi Chang, Donggeon Lee, Juntae Kim, and Seung-won Hwang. Grpe: Relative positional encoding for graph transformer. *arXiv preprint arXiv:2201.12787*, 2022.

[28] Chendi Qian, Gaurav Rattan, Floris Geerts, Mathias Niepert, and Christopher Morris. Ordered subgraph aggregation networks. *Advances in Neural Information Processing Systems*, 35: 21030–21045, 2022.

[29] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[30] Teague Sterling and John J Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

[31] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[34] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *arXiv preprint arXiv:2203.00199*, 2022.

[35] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[36] 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? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[37] Bohang Zhang, Guhao Feng, Yiheng Du, Di He, and Liwei Wang. A complete expressiveness hierarchy for subgraph gnns via subgraph weisfeiler-lehman tests. *arXiv preprint arXiv:2302.07090*, 2023.

[38] Bohang Zhang, Shengjie Luo, Liwei Wang, and Di He. Rethinking the expressive power of gnns via graph biconnectivity. *arXiv preprint arXiv:2301.09505*, 2023.

[39] Muhan Zhang and Pan Li. Nested graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[40] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

[41] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

A Implementation Details

In this section we elaborate on our specific implementation of our attention mechanism.

Implementation of Attention Mechanism We employ an attention layer inspired by *GATv2* [5] to determine the attention parameter α, as specified in Equation (7). In this context, α signifies the sparsified softmax normalization applied to an unnormalized attention coefficient, denoted by e. In particular, the unnormalized attention coefficient $e : (\mathbb{R}^d \times \mathbb{R}^d) \to \mathbb{R}$ is calculated between two elements, x_v^s and $x_{v'}^{s'}$, as follows:

$$e(x_v^s, x_{v'}^{s'}) \triangleq e_{(s,v),(s',v')} = \mathbf{a}^T \cdot \sigma\left(\mathbf{W}[x_v^s \oplus x_{v'}^{s'}]\right), \tag{13}$$

where $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters. Importantly, we compute e only for pairs for which the specific sparsification function, as described in Equation (5), yields a non-zero value. More specifically, given a sparsification function Δ, from the sparsification set, recall Equation (6), and let $\mathcal{S}_{(s,v)} = \{(s'', v'') | \Delta((s,v),(s'',v''),\mathcal{A},G) = 1\}$. Then, α is computed as follows,

$$\begin{aligned} &\alpha_{(s,v),(s',v')}(\Delta, \left(x_v^{(t);s}, x_{v'}^{(t);s'}\right), \left((s,v),(s',v')\right), \mathcal{A}^{(t)}, G) = \\ &\begin{cases} \frac{\exp\left(e_{(s,v),(s',v')}\right)}{\sum_{(s'',v'')\in\mathcal{S}_{(s,v)}} \exp\left(e_{(s,v),(s'',v'')}\right)} & \text{if } \Delta\left((s,v),(s',v'),\mathcal{A},G\right) = 1, \\ -\infty & \text{otherwise.} \end{cases} \end{aligned} \tag{14}$$

In our implementation, we also support the standard multi-head attention [5, 32, 33].

Finally, we note that in our experiments we used only the following three aggregations: Δ^{Ls}, Δ^{Lv}, and Ag_{vv}. This selection was based on their superior performance compared to other options.

A.1 Stochastic sampling

To enhance the scalability of our model, we embrace the stochastic sampling technique, as suggested in Bevilacqua et al. [2]. Specifically, during each training iteration, we stochastically omit individual subgraphs from the bag B_G based on a pre-defined rate p, with p ranging in our experiments from $\{0.95, 0.8, 0.5\}$, corresponding to $\{0.05\%, 0.2\%, 0.5\%\}$ of subgraphs which remain. For the inference phase, we perform $l = 5$ runs, each independently performing the stochastic omission of subgraphs using the same probability p. The final output is determined by a majority vote across these l inferences. Importantly, our attention mechanism is adapted to ignore subgraphs that were not sampled stochastically. Recalling Figure 1 (right), this means that arrows originating from nodes in unsampled subgraphs are effectively nullified.

A.2 Complexity analysis

In this section, we analyze the computational complexity associated with each of the proposed aggregation methods, as depicted in Figure 1 (right).

Let E be the edges and $d(\cdot)$ denote the degree of a node in the **original** graph. In the case of stochastic sampling (recall Appendix A.1), we use $\tilde{S}$ to denote the set of indices corresponding to subgraphs chosen stochastically. Thus, when no sampling is applied, we have $\tilde{S} = S$, and, using a slight abuse of notation, $d(s)$ represents the degree of node s based solely on sampled subgraphs. For clarification, consider Δ^{Gv} in Figure 1 (right) as an example. Arrows originating at cubes, which correspond to nodes in subgraphs that were not sampled, simply do not exist.

A detailed analysis is provided in Table 3.

A.3 Training details

All experiments were conducted on an NVIDIA RTX A6000 GPU. The training parameters are outlined in Table 4, while Table 5 details the hyperparameters explored.

Table 3: Complexity analysis of our proposed aggregations.

Aggregation Type	Complexity
Δ^{Gs}	$\sum_{s\in\tilde{S}}\sum_{v\in V}\lvert V\rvert = \lvert V\rvert^2\cdot\lvert\tilde{S}\rvert$
Δ^{Gv}	$\sum_{s\in\tilde{S}}\sum_{v\in V}\lvert\tilde{S}\rvert = \lvert\tilde{S}\rvert^2\cdot\lvert V\rvert$
Δ^{Ls}	$\sum_{s\in\tilde{S}}\sum_{v\in V}(1+d(v)) = \lvert\tilde{S}\rvert\cdot(\lvert V\rvert+E)$
Δ^{Lv}	$\sum_{v\in V}\sum_{s\in\tilde{S}}(1+d(s))$
Δ^{G}	$\sum_{s\in\tilde{S}}\sum_{v\in V}\lvert\tilde{S}\rvert\cdot\lvert V\rvert = \lvert\tilde{S}\rvert^2\cdot\lvert V\rvert^2$
Point Aggregations	$\sum_{s\in\tilde{S}}\sum_{v\in V}1 = \lvert\tilde{S}\rvert\cdot\lvert V\rvert$

Table 4: Training Parameters.

	ZINC
Number of layers	6
Optimizer	Adam
Scheduler	ReduceLROnPlateau
Patience	20
Learning rate	0.0005
Embedding size	96
Epochs	400
Batch size	128
Drop ratio	0.0
Weight decay	0.0

Table 5: Hyperparameters Search.

	ZINC
Number of layers	$\{6\}$
Optimizer	{Adam}
Scheduler	{ReduceLROnPlateau}
Patience	{ 20 }
Learning rate	$\{0.0005, 0.0001, 0.001\}$
Embedding size	$\{96\}$
Epochs	$\{400\}$
Batch size	$\{128\}$
Drop ratio	$\{0.0\}$
Weight decay	$\{0.0\}$