AUTOGT: AUTOMATED GRAPH TRANSFORMER ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Although Transformer architectures have been successfully applied to graph data with the advent of Graph Transformer, current design of Graph Transformer still heavily relies on human labor and expertise knowledge to decide proper neural architectures and suitable graph encoding strategies at each Transformer layer. In literature, there have been some works on automated design of Transformers focusing on non-graph data such as texts and images without considering graph encoding strategies, which fail to handle the non-euclidean graph data. In this paper, we study the problem of automated graph Transformer, for the first time. However, solving these problems poses the following challenges: i) how can we design a unified search space for graph Transformer, and ii) how to deal with the coupling relations between Transformer architectures and the graph encodings of each Transformer layer. To address these challenges, we propose Automated Graph Transformer (AutoGT), a neural architecture search framework that can automatically discover the optimal graph Transformer architectures by joint optimization of Transformer architecture and graph encoding strategies. Specifically, we first propose a unified graph Transformer formulation that can represent most of state-of-the-art graph Transformer architectures. Based upon the unified formulation, we further design the graph Transformer search space that includes both candidate architectures and various graph encodings. To handle the coupling relations, we propose a novel encoding-aware performance estimation strategy by gradually training and splitting the supernets according to the correlations between graph encodings and architectures. The proposed strategy can provide a more consistent and fine-grained performance prediction when evaluating the jointly optimized graph encodings and architectures. Extensive experiments and ablation studies show that our proposed AutoGT gains sufficient improvement over state-of-the-art hand-crafted baselines on all datasets, demonstrating its effectiveness and wide applicability.

1 INTRODUCTION

Recently, designing Transformer for graph data has attracted intensive research interests (Dwivedi & Bresson, 2020; Ying et al., 2021). As a powerful architecture to extract meaningful information from relational data, the graph Transformers have been successfully applied in natural language processing (Zhang & Zhang, 2020; Cai & Lam, 2020), social networks (Hu et al., 2020b), chemistry (Chen et al., 2019; Rong et al., 2020), and recommendation (Xia et al., 2021) etc. However, developing the state-of-the-art graph Transformer for downstream tasks is still challenging because it heavily relies on the tedious trial-and-error hand-crafted human design, including determining the best Transformer architecture and the choices of proper graph encoding strategies to utilize, etc. In addition, the inefficient hand-crafted design will also inevitably introduce the human bias, which leads to sub-optimal solutions for developing graph Transformer. In literature, there have been works on automatically searching for the architectures of Transformer, which are designed specifically for data in Natural Language Processing (Xu et al., 2021) and Computer Vision (Chen et al., 2021b). These works only focus on non-graph data without considering the graph encoding strategies which are shown to be very important in capturing graph information (Min et al., 2022a), thus failing to handle graph data with non-euclidean properties.

In this paper, we study the problem of automated graph Transformer for the first time. However, previous work (Min et al., 2022a) has demonstrated that a good graph Transformer architecture is

expected to not only select proper neural architectures for every layer, but also utilize appropriate encoding strategies capable of capturing various meaningful graph structure information to boost graph Transformer performance. Therefore, there exist two key challenges for automated graph Transformer:

- **How to design a unified search space appropriate for graph Transformer?** A good graph Transformer needs to handle the non-euclidean graph data, requiring explicit consideration of node relations within the search space, where the architectures as well as the encoding strategies can be incorporated simultaneously.
- **How to conduct encoding-aware architecture search strategy to tackle the coupling relations between Transformer architectures and graph encoding?** Although one simple solution may resort to one-shot formulation enabling efficient searching in vanilla Transformer operation space which can change its functionality during supernet training, the graph encoding strategies differ from vanilla Transformer in containing certain meanings related to structure information. How to train an encoding-aware supernet specifically designed for graph is challenging.

To address these challenges, we propose Automated Graph Transformer, AutoGT, a novel neural architecture search method for graph Transformer. In particular, we propose a unified graph Transformer formulation to cover most of the state-of-the-art graph Transformer architectures in our searching space. Besides the general search space of Transformer with hidden dimension, feed forward dimension, number of attention head, attention head dimension and number of layers, our unified search space introduces two new kinds of augmentation strategies to attain graph information: node attribution augmentation and attention map augmentation. To handle the coupling relations, we further propose a novel encoding-aware performance estimation strategy tailored for graphs. As the encoding strategy and architecture have strong coupling relations when generating results, our AutoGT split the supernet based on the important encoding strategy during evaluation to handle the coupling relations. As such, we propose to gradually train and split the supernets according to the most coupled augmentation, attention map augmentation, using various supernets to evaluate different architectures in our unified searching space, which can provide a more consistent and fine-grained performance prediction when evaluating the jointly optimized architecture and encoding. In summary, we made the following contributions:

- We propose Automated Graph Transformer, AutoGT, a novel neural architecture search framework for graph Transformer, which can automatically discover the optimal graph Transformer architectures for various down-streaming tasks. To the best of our knowledge, AutoGT is the first automated graph Transformer framework.
- We design a unified search space containing both the Transformer architectures and the essential graph encoding strategies, covering most of the state-of-the-art graph Transformer, which can lead to global optimal for structure information excavation and node information retrieval.
- We propose an encoding-aware performance estimation strategy tailored for graph to provide a more accurate and consistent performance prediction without bringing heavier computation costs. The encoding strategy and the Transformer architecture are jointly optimized to discover the best graph Transformers.
- The extensive experiments show that our proposed AutoGT model can significantly outperform the state-of-the-art baselines on graph classification tasks over several datasets with different scales.

2 Related Work

The Graph Transformer. Graph Transformer, as a category of neural networks, enables Transformer to handle graph data (Min et al., 2022a). Several works (Dwivedi & Bresson, 2020; Ying et al., 2021; Hussain et al., 2021; Zhang et al., 2020; Kreuzer et al., 2021; Shi et al., 2021) propose to pre-calculate some node positional encoding from graph structure and add them to the node attributes after a linear or embedding layer. Some works (Dwivedi & Bresson, 2020; Zhao et al., 2021; Ying et al., 2021; Khoo et al., 2020) also propose to add manually designed graph structural information into the attention matrix in Transformer layers. Others (Yao et al., 2020; Min et al., 2022b) explore the mask mechanism in the attention matrix, masking the influence of non-neighbor nodes. In particular, UniMP (Shi et al., 2021) achieves new state-of-the-art results on Open Graph Benchmark (Hu

et al., 2020a) datasets, Graphormer (Ying et al., 2021) won first place in KDD Cup Challenge on Large-SCale graph classification by encoding various information about graph structures into graph Transformer.

Neural Architecture Search. Neural architecture search has drawn increasing attention in the past few years (Elsken et al., 2019; Zoph & Le, 2016; Ma et al., 2018; Pham et al., 2018). There are many efforts to automate the design of Transformers. (So et al., 2019) propose the first automated framework for Transformer in neural machine translation tasks. AutoTrans (Zhu et al., 2021) improves the search efficiency of the NLP Transformer through a one-shot supernet training. NAS-BERT (Xu et al., 2021) further leverages the neural architecture search for big language model distillation and compression. AutoFormer (Chen et al., 2021b) migrates the automation of Transformer for vision tasks, where they utilize weight-entanglement to improve the consistency of the supernet training. GLiT (Chen et al., 2021a) proposes to search both global and local attention for the Vision Transformer using a hierarchical evolutionary search algorithm. Chen et al., (Chen et al., 2021c) further propose to evolve the search space of the Vision Transformer to solve the exponential explosion problems.

3 Automated Graph Transformer Architecture Search (AutoGT)

To automatically design graph Transformer architectures, we first unify the formulation of current graph Transformers in Section 3.1. Based on the unified formulation, we design the search space tailored for the graph Transformers in Section 3.2. We propose a novel encoding-aware performance estimation strategy in Section 3.3, and introduce our evolutionary search strategy in Section 3.4. The whole algorithm is presented by Figure 2.

3.1 The Unified Graph Transformer Framework

Current representative graph Transformer designs can be regarded as improving the input and attention map in Transformer architecture through various graph encoding strategies. We first introduce the basic Transformer architecture and then show how to combine various graph encoding strategies.

Let $G = (V, E)$ denote a graph where $V = \{v_1, v_2, \cdots, v_n\}$ represents the set of nodes and $E = \{e_1, e_2, \cdots, e_m\}$ represents the set of edges, and denote $n = |V|$ and $m = |E|$ as the number of nodes and edges, respectively. Let $\mathbf{v}_i, i \in \{1, ..., n\}$ represents the features of node v_i, and $\mathbf{e}_j, j \in \{1, ..., m\}$ represents the features of edge e_j.

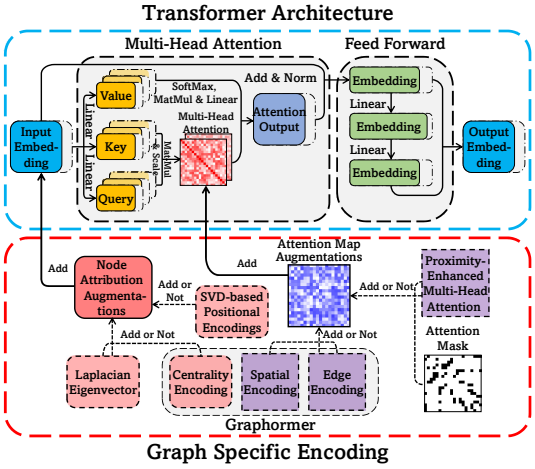

Figure 1: The unified graph Transformer search space. It consists of the Transformer architecture space and the graph specific encoding space. The Transformer architecture search space is detailed in Table 1. The graph specific encoding search space is to decide whether each encoding strategy should be adopted or not and the mask threshold for the attention mask.

3.1.1 Basic Transformer

As shown in Figure 1, a basic Transformer consists of several stacked blocks, with each block containing two modules, namely the multi-head attention (MHA) module and the feed-forward network (FFN) module.

At block l, the node representation $\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d}$ first goes through the MHA module to interact with each other and pass information through self-attention:

$$\mathbf{A}_h^{(l)} = \text{softmax}\left(\frac{\mathbf{Q}_h^{(l)}\mathbf{K}_h^{(l)T}}{\sqrt{d_k}}\right), \mathbf{O}_h^{(l)} = \mathbf{A}_h^{(l)}\mathbf{V}_h^{(l)}, \tag{1}$$

where $\mathbf{A}_h^{(l)} \in \mathbb{R}^{n \times n}$ is the message passing matrix, $\mathbf{O}_h^{(l)}$ is the output of the self-attention mechanism of the hth attention head, $h = 1, 2, \cdots, Head$, $Head$ is the number of attention heads, and $\mathbf{K}_h^{(l)}, \mathbf{Q}_h^{(l)}, \mathbf{V}_h^{(l)} \in \mathbb{R}^{n \times d_k}$ are the key, query, value calculated as:

$$\mathbf{K}_h^{(l)}, = \mathbf{H}^{(l)}\mathbf{W}_{k,h}^{(l)}, \mathbf{Q}_h^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_{q,h}^{(l)}, \mathbf{V}_h^{(l)} = \mathbf{H}^{(l)}\mathbf{W}_{v,h}^{(l)}, \tag{2}$$

where $\mathbf{W}_{k,h}^{(l)}, \mathbf{W}_{q,h}^{(l)}, \mathbf{W}_{v,h}^{(l)} \in \mathbb{R}^{d \times d_k}$ are learnable parameters. Then, the representations of different heads are concatenated and further transformed as:

$$\mathbf{O}^{(l)} = (\mathbf{O}_1^{(l)} \circ \mathbf{O}_2^{(l)} \circ ... \circ \mathbf{O}_{Head}^{(l)})\mathbf{W}_O^{(l)} + \mathbf{H}^{(l)}, \tag{3}$$

where $\mathbf{W}_O^{(l)} \in \mathbb{R}^{(d_k * Head) \times d_t}$ is the parameter and $\mathbf{O}^{(l)}$ is the multi-head result. Then, the attended representation will go through the FFN module to further refine the information of each node:

$$\mathbf{H}^{(l+1)} = \sigma(\mathbf{O}^{(l)}\mathbf{W}_1^{(l)})\mathbf{W}_2^{(l)}, \tag{4}$$

where $\mathbf{O}^{(l)} \in \mathbb{R}^{n \times d_t}$ is the output, $\mathbf{W}_1^{(l)} \in \mathbb{R}^{d_k \times d_h}$, $\mathbf{W}_2^{(l)} \in \mathbb{R}^{d_h \times d}$ are weight matrices.

As for the input of the first block, we concatenate all the node features $\mathbf{H}^{(0)} = [\mathbf{v}_1, ..., \mathbf{v}_n]$. After L blocks, we obtain the final representation of each node $\mathbf{H}^{(L)}$.

3.1.2 Graph Encoding Strategy

From Section 3.1.1, we can observe that directly using the basic Transformer on graphs can only process node attributes, ignoring important edge attributes and graph topology information in the graph. To make the Transformer architecture aware of the graph structure, several works resort to various graph encoding strategies, which can be divided into two kinds of categories: node attribution augmentation and attention map augmentation.

The node attribution augmentations take the whole graph G as input and generate the topology-aware features $Enc_{node}(G)$ for each node to directly improve the node representations:

$$\mathbf{H}_{aug}^{(l)} = \mathbf{H}^{(l)} + Enc_{node}(G). \tag{5}$$

On the other hand, the attention map augmentations generate an additional attention map $Enc_{map}(G)$, which represents the relationships of any two nodes and improves the attention map generated by self-attention in Eq equation 1 as:

$$\mathbf{A}_{h,aug}^{(l)} = \text{softmax}\left(\frac{\mathbf{Q}_h^{(l)}\mathbf{K}_h^{(l)^T}}{\sqrt{d}} + Enc_{map}(G)\right). \tag{6}$$

Combining node attribution augmentations and attention map augmentations together, our proposed framework is as follows:

$$\mathbf{H}^{(l+1)} = \sigma\left(\text{Concat}\left(\text{softmax}\left(\frac{\mathbf{H}_{aug}^{(l)}\mathbf{W}_{q,h}^{(l)}(\mathbf{H}_{aug}^{(l)}\mathbf{W}_{k,h}^{(l)})^T}{\sqrt{d_k}} + Enc_{map}(G)\right)\mathbf{H}_{aug}^{(l)}\mathbf{W}_{v,h}^{(l)}\right)\mathbf{W}_1^{(l)}\right)\mathbf{W}_2^{(l)}. \tag{7}$$

where $\mathbf{H}_{aug}^{(l)} = \mathbf{H}^{(l)} + Enc_{node}(G)$.

3.2 The Graph Transformer Search Space

Based on the unified graph Transformer formulation, we propose our unified search space design, which can be decomposed into two parts, i.e., Transformer Architecture space and graph encoding space. Figure 1 shows the unified graph Transformer search space.

3.2.1 Transformer Architecture Space

Following Section 3.1.1, we automate five key architecture components for graph Transformer as follows: the number of encoder layers L, the dimension d, the intermediate dimension d_t, the hidden dimension d_h, the number of attention heads $Head$, and the attention head dimension d_k in graph Transformer. Notice that these five components already cover the most important designs for Transformer architectures.

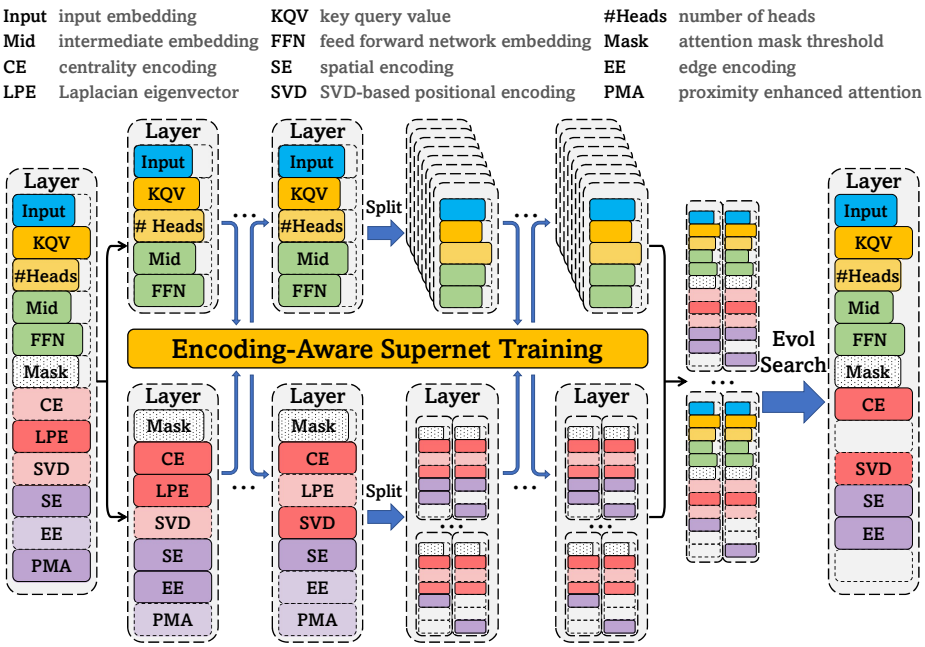

Figure 2: The framework of our work. Firstly, we construct the search space for each layer, consisting of the Transformer architecture space (above) and the graph encoding strategy space (below). Then, we carry out our encoding-aware supernet training method in two stages: before splitting, we train a supernet by randomly sampling architectures from the search space, while after splitting, we train multiple subnets (inheriting the weights from the supernet) by randomly sampling architectures with fixed attention map augmentation strategies (except for the attention mask). Finally, we conduct an evolutionary search based on the subnets and obtain our final architecture and results.

A suitable search space should be not only expressive enough to allow powerful architectures, but also compact enough to enable efficient searches. With this principle in mind, we propose two search spaces for these components with different size ranges. Table 1 gives the detailed search space for these two spaces.

3.2.2 Graph Encoding Space

To fully exploit the potential of the graph encoding strategies, we further determine whether and which graph encoding strategies to use for each layer of the graph Transformer. Specifically, we explore the node attribution augmentations encoding and attention map augmentations encoding as below.

Node Attribution Augmentations:

- **Centrality Encoding** (Ying et al., 2021). Use two node embeddings with the same size representing the in-degree and the out-degree of nodes, i.e.,

$$h_i^{(l)} = x_i^{(l)} + z^-_{deg^-(v_i)} + z^+_{deg^-(v_i)} \tag{8}$$

  where $h_i^{(l)}$ is the input embedding in layer l, x_i is the input attribution of node i in layer l, and z^- and z^+ are the embedding generated by the in-degree and out-degree.

- **Laplacian Eigenvector** (Dwivedi & Bresson, 2020). Conducting spectral decomposition of the graph Laplacian matrix:

$$\mathbf{U}^T \Lambda \mathbf{U} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A}^G \mathbf{D}^{-1/2} \tag{9}$$

  where $\mathbf{A}^G$ is the adjacency matrix of graph G, $\mathbf{D}$ is the diagonal degree matrix, and $\mathbf{U}$ and Λ are the eigenvectors and eigenvalues, respectively. We only select the eigenvectors of the k smallest non-zero eigenvalues as the final embedding and concatenate them to the input node attribute matrix for each layer.

Table 1: The Transformer Architecture Search Space for $\mathrm{AutoGT}_{\mathrm{base}}$ and AutoGT.

	$\mathrm{AutoGT}_{\mathrm{base}}$		AutoGT	
	Choices	Supernet Size	Choices	Supernet Size
#Layers	{2,3,4}	4	{5,6,7,8}	8
Input Dimension d	{24,28,32}	32	{96,112,128}	128
Intermediate Dimension d_t	{24,28,32}	32	{96,112,128}	128
Hidden Dimension d_h	{24,28,32}	32	{96,112,128}	128
#Attention Heads	{2,3,4}	4	{6,7,8}	8
Attention Head Dimension d_k	{6,8}	8	{12,14,16}	16

- **SVD-based Positional Encoding** (Hussain et al., 2021). Conducting singular value decomposition to the graph adjacency matrix:

$$\mathbf{A}^G \overset{\text{SVD}}{\approx} \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = (\mathbf{U}\sqrt{\mathbf{\Sigma}}) \cdot (\mathbf{V}\sqrt{\mathbf{\Sigma}})^T = \hat{\mathbf{U}}\hat{\mathbf{V}}^T \tag{10}$$

  where $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{n \times r}$ contains the left and right singular vectors of the top r singular values in the diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$. Without loss of generality, we only choose $\hat{\mathbf{U}}$ as the final embedding since they are highly correlated for symmetric graphs (with differences in signs, to be specific). Similar to the Laplacian eigenvector, we concatenate it to the input node attribute matrix for each layer.

Attention Map Augmentations Space:

- **Spatial Encoding** (Ying et al., 2021). Spatial encoding is added to the attention result before softmax:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_k}} + b_{\phi(v_i, v_j)} \tag{11}$$

  where $\phi(v_i, v_j)$ is the length of the shortest path from v_i to v_j, and $b \in \mathbb{R}$ is a weight parameter generated by $\phi(v_i, v_j)$.
- **Edge Encoding** (Ying et al., 2021). Edge encoding is also added to the attention result before softmax:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_k}} + \frac{1}{N} \sum_{n=1}^{N} x_{e_n} (w_n^E)^T \tag{12}$$

  where x_{e_n} is the feature of the n-th edge e_n on the shortest path between v_i and v_j, and w_n^E is the n-th learnable embedding vector.
- **Proximity-Enhanced Attention** (Zhao et al., 2021). Edge encoding is also added to the attention result before softmax:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_k}} + \phi_{ij}^T b \tag{13}$$

  where $b \in \mathbb{R}^{M \times 1}$ is a learnable parameter, $\phi_{ij} = \mathrm{Concat}(\Phi_m(v_i, v_j) | m \in 0, 1, \cdots, M-1)$ is the structural encoding generated from: $\Phi_m(v_i, v_j) = \tilde{\mathbf{A}}^m[i, j]$, where $\tilde{\mathbf{A}} = \mathrm{Norm}(\mathbf{A} + \mathbf{I})$ represents the normalized adjacency matrix. Thus the augmentation denotes the reachable probabilities between nodes.
- **Attention Mask** (Min et al., 2022b; Yao et al., 2020). Attention Mask is added to the attention result before softmax:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d_k}} + \mathrm{Mask}_m(v_i, v_j) \tag{14}$$

  where m is the mask threshold, $\mathrm{Mask}_m(v_i, v_j)$ depends on the relationship between m and $\phi(v_i, v_j)$, i.e. the shortest path length between v_i and v_j. When $m \geq \phi(v_i, v_j)$, $\mathrm{Mask}_m(v_i, v_j) = 0$. Otherwise, $\mathrm{Mask}_m(v_i, v_j)$ is $-\infty$, masking the corresponding attention in practical terms.

3.3 Encoding-Aware Supernet Training

We next introduce our proposed encoding-aware performance estimation strategy for efficient training.

Similar to general NAS problems, the graph Transformer architecture search can be formulated as a bi-level optimization problem:

$$a^* = \mathrm{argmax}_{a \in \mathcal{A}} Acc_{val}(\mathbf{W}^*(a), a), \qquad \mathrm{s.t.} \mathbf{W}^*(a) = \mathrm{argmin}_{\mathbf{W}} \mathcal{L}_{train}(\mathbf{W}, a), \tag{15}$$

Table 2: Comparison of our proposed unified framework with state-of-the-art graph Transformer models. CE, LPE, SVD, SE, EE, PMA, Mask denote Centrality Encoding, Laplacian Eigenvector, SVD-based Positional Encoding, Spatial Encoding, Edge Encoding, Proximity-Enhanced Attention, and Attention Mask respectively.

	CE	LPE	SVD	SE	EE	PMA	Mask
EGT (Hussain et al., 2021)			✓				
Gophormer (Zhao et al., 2021)						✓	
Graph Trans (Dwivedi & Bresson, 2020)		✓					✓
Graphormer (Ying et al., 2021)	✓			✓	✓		
Ours	✓	✓	✓	✓	✓	✓	✓

where $a \in \mathcal{A}$ is the architecture in the search space $\mathcal{A}$, Acc_{val} stands for the validation accuracy, $\mathbf{W}$ represents the learnable weights, and a^* and $\mathbf{W}^*(a)$ denotes the optimal architecture and the optimal weights for the architecture a.

Following one-shot NAS methods (Liu et al., 2019; Pham et al., 2018), we encode all candidate architectures in the search space into a supernet and transform Eq. equation 15 into a two-step optimization (Guo et al., 2020):

$$a^* = \text{argmax}_{a \in \mathcal{A}} Acc_{val}(\mathbf{W}^*, a), \qquad \mathbf{W}^* = \text{argmin}_{\mathbf{W}} \mathbb{E}_{a \in \mathcal{A}} \mathcal{L}_{train}(\mathbf{W}, a), \tag{16}$$

where $\mathbf{W}$ denotes the shared learnable weights in the supernet with its optimal value $\mathbf{W}^*$ for all the architectures in the search space.

To further improve the optimization efficiency of the supernet training, we leverage weight entanglement (Guan et al., 2021; Chen et al., 2021b; Guo et al., 2020) to deeply share the weights of architectures with different hidden sizes. Specifically, for every architecture sampled from the supernet, we use a 0-1 mask to discard unnecessary hidden channels instead of maintaining a new set of weights. In this way, the number of parameters in the supernet will remain the same as the largest (i.e., with the most parameters) model in the search space, thus leading to efficient optimization.

Although this strategy is fast and convenient, using the same supernet parameters $\mathbf{W}$ for all architectures will decrease the consistency between the estimation of the supernet and the ground-truth architecture performance. To improve the consistency and accuracy of supernet, we propose an encoding-aware supernet training strategy. Based on the contribution of coupling of different encoding strategies, we split the search space into different sub-spaces based on whether adopting three kinds of attention map augmentation strategies: spatial encoding, edge encoding, and proximity-enhanced attention. Therefore, there are $2^3 = 8$ supernets.

To be specific, we first train a single supernet for certain epochs and split the supernet into 8 subnets according to the sub-spaces afterward. Then, we continuously train the weights in each subnet $\mathbf{W}_i$ by only sampling the architecture from the corresponding subspace A_i. Experiments to support such a design are provided in Section 4.

3.4 Evolutionary Search

Similar to other NAS research, our proposed graph transformer search space is too large to enumerate. Therefore, we propose to utilize the evolutionary algorithm to efficiently explore the search space to obtain the architecture with optimal accuracy on the validation dataset.

Specifically, we first maintain a population consisting of T architectures by random sample. Then, we evolve the architectures through our designed mutation and crossover operations. In the mutation operation, we randomly choose from the top-k architectures with the highest performance in the last generation and change its architecture choices with probabilities. In the crossover operation, we randomly select pairs of architectures with the same number of layers from the remaining architectures, and randomly switch their architecture choices.

Table 3: Comparisons of AutoGT against state-of-the-art hand-crafted baselines. We report the average accuracy (%) and the standard deviation on all the datasets. Out-of-time (OOT) indicates the method cannot produce results in 1 GPU day.

Dataset	COX2_MD	BZR_MD	PTC_FM	DHFR_MD	PROTEINS	DBLP
GIN	$45.82_{14.35}$	$59.68_{14.65}$	$57.87_{8.86}$	$62.88_{8.26}$	$73.76_{4.61}$	$91.18_{0.42}$
DGCNN	$54.81_{18.51}$	$62.74_{20.59}$	$62.17_{3.62}$	$63.89_{5.91}$	$72.68_{3.75}$	$91.57_{0.54}$
DiffPool	$51.45_{14.28}$	$65.01_{14.74}$	$60.16_{5.87}$	$61.06_{9.42}$	$73.31_{3.75}$	OOT
GraphSAGE	$49.59_{12.80}$	$57.43_{13.50}$	$64.17_{3.28}$	$66.92_{2.35}$	$67.19_{6.97}$	$51.01_{0.02}$
Graphormer	$56.39_{15.03}$	$63.94_{12.58}$	$64.88_{7.58}$	$64.88_{7.58}$	$75.29_{3.10}$	$89.36_{2.31}$
GT(ours)	$54.44_{16.84}$	$63.33_{11.67}$	$64.18_{2.60}$	$65.68_{5.64}$	$73.94_{3.78}$	$90.67_{1.01}$
AutoGT(ours)	$\mathbf{59.72}_{23.26}$	$\mathbf{65.92}_{10.00}$	$\mathbf{65.60}_{3.71}$	$\mathbf{68.22}_{5.02}$	$\mathbf{77.17}_{3.40}$	$\mathbf{91.66}_{0.79}$

Table 4: Comparisons of AutoGT against state-of-the-art hand-crafted baselines. We report the area under the curve (AUC) [%] and the standard deviation on all the datasets.

Dataset	OGBG-MolHIV	OGBG-MolBACE	OGBG-MolBBBP
GIN	$71.11_{2.57}$	$70.42_{4.78}$	$63.37_{1.81}$
DGCNN	$69.97_{2.16}$	$75.62_{2.64}$	$60.92_{1.78}$
DiffPool	$74.58_{1.71}$	$73.87_{4.50}$	$66.68_{6.08}$
GraphSAGE	$67.82_{3.67}$	$72.91_{1.24}$	$64.19_{3.50}$
Graphormer	$71.89_{2.66}$	$76.42_{1.67}$	$66.52_{0.74}$
AutoGT(ours)	$\mathbf{74.95}_{1.02}$	$\mathbf{76.70}_{1.42}$	$\mathbf{67.29}_{1.46}$

4 EXPERIMENTS

In this section, we present detailed experimental results as well as the ablation studies to empirically show the effectiveness of our proposed AutoGT.

Datasets and Baselines. We first consider six graph classification datasets from Deep Graph Kernels Benchmark((Yanardag & Vishwanathan, 2015)) and TUDataset (Morris et al., 2020), namely COX2_MD, BZR_MD, PTC_FM, DHFR_MD, PROTEINS, and DBLP. We also adopt three datasets from Open Graph Benchmark (OGB) (Hu et al., 2020a), including OGBG-MolHIV, OGBG-MolBACE, and OGBG-MolBBBP. The task is to predict the label of each graph using node/edge attributes and graph structures. The detailed statistics of the datasets are shown in Table 6 in the appendix.

We compare AutoGT with state-of-the-art hand-crafted baselines, including GIN (Xu et al., 2019), DGCNN (Zhang et al., 2018), DiffPool (Ying et al., 2018), GraphSAGE (Hamilton et al., 2017), and Graphormer (Ying et al., 2021). Notice that Graphormer is a state-of-the-art graph Transformer architecture that won first place in the graph classification task of KDD Cup 2021 (OGB-LSC).

For all the datasets, we follow Errica et al., (Errica et al., 2020) to utilize 10-fold cross-validation for all the baselines and our proposed method. All the hyper-parameters and training strategies of baselines are implemented according to the publicly available codes (Errica et al., 2020)[1].

Implementation Details. Recall that our proposed architecture space has two variants, a larger $\mathrm{AutoGT}(L = 8, d = 128)$ and a smaller $\mathrm{AutoGT}_{base}(L = 4, d = 32)$. In our experiments, we adopt the smaller search space for five relatively small datasets, i.e., all datasets except DBLP, and the larger search space for DBLP. We use the Adam optimizer, and the learning rate is $3e - 4$. For the smaller/larger datasets, we set the number of iterations to split (i.e., T_s in Algorithm 1 in Appendix) as 50/6 and the maximum number of iterations (i.e., T_m in Algorithm 1) as 200/50. The batch size is 128. The hyperparameters of these baselines are kept consistent with our method for a fair comparison.

We also report the results of our unified framework in Section 3.1, i.e. mixing all the encodings in our search space with the supernet but without the search part, denoted as GT(Graph Transformer).

[1]https://github.com/diningphil/gnn-comparison

Experimental Results. We report the results of AutoGT and baselines in Table 3. We can make the following observations. First, AutoGT consistently outperforms all the existing hand-crafted methods on all datasets, demonstrating the effectiveness of our proposed method. Graphormer shows remarkable performance and achieves the second-best results on three datasets, showing the great potential of Transformer architectures in processing graph data. However, since Graphormer is a manually designed architecture and cannot adapt to different datasets, it fails to be as effective as our proposed automatic solution. Lastly, GT, our proposed unified framework, fails to show strong performance in most cases. The results indicate that simply mixing different graph Transformers cannot produce satisfactory results, demonstrating the importance of searching for effective architectures to handle different datasets.

We also conduct experiments on Open Graph Benchmark (OGB) (Hu et al., 2020a). On the three binary classification datasets of OGB, we report the AUC score of our method and all the baselines. The results also show that our method outperforms all the hand-crafted baselines on these datasets.

Time Cost. We further show the time comparison of our proposed AutoGT with hand-crafted graph transformer Graphormer. On OGBG-MolHIV dataset, both Graphormer and AutoGT cost 2 minutes for one epoch on single GPU. The default Graphormer is trained for 300 epochs, which costs 10 hours to obtain the result of one random seed. For AutoGT, we train shared supernet for 50 epochs, 8 supernets inherit, and continue to train for 150 epochs. So the training process costs totally 1250 epochs with 40 hours. And on the evolutionary search stage, we evaluate 2000 architectures' inheriting weight performances, which costs about 900 epochs with 30 hours. In summary, the total time cost for our proposed AutoGT is only 7 times total time cost for a hand-crafted graph transformer Graphormer.

Ablation Studies. We verify the effectiveness of the proposed encoding-aware supernet training strategy by reporting the results on the PROTEINS dataset, while other datasets show similar patterns.

To show the importance of considering encoding strategies when training the supernet, we design two variants of AutoGT and compare the results:

- **One-Shot.** We only train a single supernet and use it to evaluate all the architectures.
- **Positional-Aware.** We also split up the supernet into 8 subnets but based on three node attribute augmentations instead of the three attention map augmentation as in AutoGT.

The results of AutoGT and two variants are shown in Table 5. From the table, we can observe that, compared with the result of one-shot NAS, positional-aware and AutoGT methods achieve different levels of improvement. Further comparing the accuracy gain, we find that the result of AutoGT (1.25%) is nearly 5 times larger than the result of positional-aware (0.27%), even though both methods adopt 8 subnets. We attribute the significant difference in accuracy gain from supernet splitting to the different degrees of coupling of graph encoding strategies with the Transformer architecture. For example, the dimensionality of node attribution augmentation is the same as the number of nodes, while the attention map augmentation has a quadratic dimensionality, resulting in different coupling degrees. Our proposed encoding-aware performance estimation based on three attention map augmentation strategies is shown to be effective in practice.

Table 5: The ablation study on the effectiveness of the proposed encoding-aware supernet training strategy. We report the average accuracy[%] with the variance on PROTEINS.

Method	Accuracy
One-Shot	$75.92_{3.10}$
Positional-Aware	$76.19_{3.42}$
AutoGT	$77.17_{3.40}$

5 Conclusion

In this paper, we propose AutoGT, a neural architecture search framework for graph Transformer. We design a search space tailored for graph Transformer architectures, and an encoding-aware supernet training strategy to provide reliable graph Transformer supernets considering various graph encoding strategies. Our method integrates the existing graph Transformer into a unified framework, where different Transformer encoding can enhance each other. Extensive experiments on six datasets demonstrate that our proposed AutoGT consistently outperforms state-of-the-art baselines on all datasets, demonstrating its strength on various graph tasks.

REFERENCES

Deng Cai and Wai Lam. Graph transformer for graph-to-sequence learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7464–7471, 2020.

Benson Chen, Regina Barzilay, and Tommi Jaakkola. Path-augmented graph transformer network. *arXiv preprint arXiv:1905.12712*, 2019.

Boyu Chen, Peixia Li, Chuming Li, Baopu Li, Lei Bai, Chen Lin, Ming Sun, Junjie Yan, and Wanli Ouyang. Glit: Neural architecture search for global and local image transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12–21, October 2021a.

Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12270–12280, October 2021b.

Minghao Chen, Kan Wu, Bolin Ni, Houwen Peng, Bei Liu, Jianlong Fu, Hongyang Chao, and Haibin Ling. Searching the search space of vision transformer. *CoRR*, abs/2111.14725, 2021c.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Chaoyu Guan, Yijian Qin, Zhikun Wei, Zeyang Zhang, Zizhao Zhang, Xin Wang, and Wenwu Zhu. One-shot neural channel search: What works and what's next. *In CVPR Workshop on NAS*, 2021.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVI*, volume 12361 of *Lecture Notes in Computer Science*, pp. 544–560. Springer, 2020.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pp. 2704–2710. ACM / IW3C2, 2020b.

Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv preprint arXiv:2108.03348*, 2021.

Ling Min Serena Khoo, Hai Leong Chieu, Zhong Qian, and Jing Jiang. Interpretable rumor detection in microblogs by attending to user interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8783–8790, 2020.

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

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

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 preprint arXiv:2202.08455*, 2022a.

Erxue Min, Yu Rong, Tingyang Xu, Yatao Bian, Peilin Zhao, Junzhou Huang, Da Luo, Kangyi Lin, and Sophia Ananiadou. Masked transformer for neighhourhood-aware click-through rate prediction. *arXiv preprint arXiv:2201.13311*, 2022b.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4092–4101. PMLR, 2018.

Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying Wei, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 1548–1554. ijcai.org, 2021.

David R. So, Quoc V. Le, and Chen Liang. The evolved transformer. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5877–5886. PMLR, 2019.

Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4486–4493, 2021.

Jin Xu, Xu Tan, Renqian Luo, Kaitao Song, Jian Li, Tao Qin, and Tie-Yan Liu. NAS-BERT: task-agnostic and adaptive-size BERT compression with neural architecture search. In Feida Zhu, Beng Chin Ooi, and Chunyan Miao (eds.), *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1933–1943. ACM, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7145–7154, 2020.

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 *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=OeWooOxFwDa.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Haopeng Zhang and Jiawei Zhang. Text graph transformer for document classification. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In Sheila A. McIlraith and Kilian Q. Weinberger (eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 4438–4445. AAAI Press, 2018.

Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

Wei Zhu, Xiaoling Wang, Yuan Ni, and Guotong Xie. Autotrans: Automating transformer design via reinforced architecture search. In *Natural Language Processing and Chinese Computing - 10th CCF International Conference, NLPCC 2021, Qingdao, China, October 13-17, 2021, Proceedings, Part I*, volume 13028 of *Lecture Notes in Computer Science*, pp. 169–182. Springer, 2021.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A TRAINING PROCEDURE

We list the training procedure of our method in Algorithm 1.

Algorithm 1 Our proposed encoding-aware supernet training strategy

Initialize. Supernet weights $\mathbf{W}$, subnet weights $\mathbf{W}_i$, search space $\mathcal{A}$, subspace $\mathcal{A}_i$, the split iteration T_s, the max iteration T_m, a dataset $\mathcal{D}$.
for $t = 1 : T_s$ **do**
  Sample architecture and encoding strategies $a \in \mathcal{A}$.
  Sample a batch of graph data $\mathcal{D}_s \subset \mathcal{D}$.
  Calculate the training loss $\mathcal{L}_{train}$ over the sampled data.
  Update the supernet weights $\mathbf{W}$ through gradient descents.
end for
for $i = 1 : n$ **do**
  Let subnet $\mathbf{W}_i$ inherit the weights from supernet $\mathbf{W}$.
  for $t = T_s : T_m$ **do**
    Sample architectures and encoding strategies from the subspace $a \in \mathcal{A}_i$.
    Sample a batch of graph data $\mathcal{D}_s \in \mathcal{D}$.
    Calculate the training loss $\mathcal{L}_{train}$ over the sampled batch.
    Update the subnet weights $\mathbf{W}_i$ through gradient descents.
  end for
end for
Output. Subnets with weights $\mathbf{W}_i$.

B DATASET

We provide the statistics of the adopted datasets in Table 6 and Table 7.

Table 6: Statistics of graph classification datasets (precision) used to compare AutoGT with baselines. We adopt five datasets with relatively small numbers of graphs (upper part) and one dataset with a larger size (lower part) to demonstrate the efficiency of the proposed AutoGT.

Dataset	#Graph	#Class	#Avg. Nodes	#Avg. Edges	# Node Feature	# Edge Feature
COX2_MD	303	2	26.28	335.12	7	5
BZR_MD	306	2	21.3	225.06	8	5
PTC_FM	349	2	14.11	14.48	18	4
DHFR_MD	393	2	23.87	283.01	7	5
PROTEINS	1,133	2	39.06	72.82	3	0
DBLP	19,456	2	10.48	19.65	41,325	3

Table 7: Statistics of graph classification datasets (AUC) used to compare AutoGT with baselines. We adopt two datasets with relatively small numbers of graphs (upper part) and one dataset with a larger size (lower part) to demonstrate the efficiency of the proposed AutoGT.

Dataset	#Graph	#Class	#Avg. Nodes	#Avg. Edges	# Node Feature	# Edge Feature
OGBG-MolHIV	1,513	2	25.51	27.47	9	3
OGBG-MolBACE	2,039	2	34.09	36.86	9	3
OGBG-MolBBBP	41,127	2	24.06	25.95	9	3

C ADDITIONAL EXPERIMENTS

In Table 3, the results on COX2_MD and BZR_MD show larger standard deviations than other datasets. One plausible reason is that the number of graphs of these two datasets are relatively small, so that the results of the model can be sensitive to dataset splits. To obtain more convincing results on these two datasets, we conduct additional experiments by still utilizing 10-fold cross-validation for all

Table 8: Comparisons of AutoGT against state-of-the-art hand-crafted baselines. We report the average accuracy (%) and the standard deviation on all the datasets.

Dataset	COX2_MD	BZR_MD
GIN	$57.22_{9.74}$	$62.64_{8.23}$
DGCNN	$60.33_{7.56}$	$64.91_{9.42}$
DiffPool	$59.52_{8.20}$	$64.84_{8.51}$
GraphSAGE	$53.62_{6.95}$	$55.83_{8.27}$
Graphormer	$59.22_{7.04}$	$64.53_{9.43}$
AutoGT(ours)	$\mathbf{63.45}_{8.04}$	$\mathbf{67.18}_{9.87}$

Table 9: Comparisons of AutoGT with the different number of supernets. We report the average accuracy (%) and the standard deviation on the datasets.

Dataset	PROTEINS
1 supernet	$75.92_{3.10}$
2 supernet	$76.73_{3.25}$
4 supernet	$76.91_{3.35}$
8 supernet	$77.17_{3.40}$
16 supernet	$77.27_{3.65}$

the baselines and our proposed method, and repeat the 10-fold cross-validation with 10 random seeds. We report the results in Table 8. The results are consistent with Table 3, i.e., our method consistently outperforms other baselines, while the standard deviations are considerably smaller by adopting more repeated experiments.

In addition, to further explore how the number of supernets affects our proposed method, we carry out experiments with 1, 2, 4, 8, 16 supernets on the PROTEINS dataset, and report our results in Table 9. We can observe that as the number of subnets increases, the performance of our method increases. One possible reason is that more well-trained subnets can bring more consistent performance estimation results, which improves performance.