

Can Transformer and GNN Help Each Other?

Peiyan Zhang*
Hong Kong University of
Science and Technology
Hong Kong
pzhangao@cse.ust.hk

Yuchen Yan*
School of Intelligence Science and
Technology, Peking University
Beijing, China
2001213110@stu.pku.edu.cn

Chaozhuo Li
Microsoft Research Asia
Beijing, China
cli@microsoft.com

Senzhang Wang
Central South University
China
szwang@csu.edu.cn

Xing Xie
Microsoft Research Asia
Beijing, China
xing.xie@microsoft.com

Sunghun Kim
Hong Kong University of
Science and Technology
Hong Kong
hunkim@cse.ust.hk

ABSTRACT

Although Transformer has achieved great success in natural language process and computer vision, it has difficulty generalizing to medium and large scale graph data for two important reasons: (i) High complexity. (ii) Failing to capture the complex and entangled structure information. In graph representation learning, Graph Neural Networks (GNNs) can fuse the graph structure and node attributes but have limited receptive fields. Therefore, we question that can we combine Transformer and GNNs to help each other? In this paper, we propose a new model named **TransGNN** where the Transformer layer and GNN layer are used alternately to improve each other. Specifically, to expand the receptive field and disentangle the information aggregation from edges, we propose using Transformer to aggregate more relevant nodes' information to improve the message passing of GNNs. Besides, to capture the graph structure information, we utilize positional encoding and make use of the GNN layer to fuse the structure into node attributes, which improves the Transformer in graph data. We also propose to sample the most relevant nodes for Transformer and two efficient samples update strategies to lower the complexity. At last, we theoretically prove that TransGNN is more expressive than GNNs only with extra linear complexity. The experiments on eight datasets corroborate the effectiveness of TransGNN on node and graph classification tasks.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → *Neural networks*; Supervised learning.

KEYWORDS

Graph data, Graph Neural Networks, Transformer

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...\$15.00
<https://doi.org/XXXXXXX.XXXXXXX>

ACM Reference Format:

Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, and Sunghun Kim. 2018. Can Transformer and GNN Help Each Other?. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

In recent years, Transformer has achieved great success in the field of natural language process (NLP) [9] and computer vision (CV) [63]. It is well acknowledged as the most powerful neural network and has displaced the convolutional and recurrent neural networks to become the new de-facto standard among many tasks such as language understanding [7, 9, 54, 67], machine translation [46], and image segmentation [55–57]. Although Transformer has been proven successful in sequence data [26, 39, 48, 49, 62, 66], it has not been generalized to graphs well. Graph data are also very important in our lives, which can describe many kinds of relationships between entities and contain much useful knowledge. For example, we can predict the property of new molecules by analyzing their graph structures and node attributes [38, 58].

The main challenges confronted by Transformer when used in graph data are in two folds: (i) Calculating attention weights on the whole graph data for every central node will result in $O(N^2)$ time and space complexity, which leads to the out-of-memory problem when graph becomes large. (ii) Although Transformer can aggregate information globally and adaptively, it is difficult for Transformer to fully use the graph structure information because the aggregation process does not rely on edges at all.

In graph representation learning, Graph Neural Networks (GNNs) have become the dominant choice [12, 17, 18, 20, 21, 21, 24, 30, 32, 33, 40, 47, 64], and various GNN architectures achieve state-of-the-art performances in many graph-based tasks, such as node classification [29, 45], link prediction [22, 27, 35, 61], and graph classification [4, 15, 28, 51, 53]. Although great improvements have been achieved by GNNs and their variants, there still exist some limitations. On the one hand, the message passing mechanism relies on edges to fuse the graph structures and node attributes, leading to strong bias and noise [3]. Some useful information can not be obtained if there is no connection between nodes. Besides, in real graph data, there are many noisy edges that connect the irrelevant nodes. These bias and noise caused by the message passing

mechanism make the representations learned by GNNs contain incomplete information [41], which compromises the performance of downstream tasks. On the other hand, the receptive field of GNNs is also limited because of the over-smoothing problem [37]. It has been proven that as the GNNs architecture goes deeper and reaches a certain extent, the model will not respond to the training data, and the node representations obtained by such deep models tend to be over-smoothed and also become indistinguishable [2, 6, 11, 36]. These limitations lead to a frustrating compromise of GNNs where shallow GNNs with a limited receptive field can only aggregate incomplete information within the neighborhood while deep GNNs have the over-smoothing problem.

In this paper, we question that can we combine Transformer and GNNs to help each other by fusing their own strengths? With the help of the Transformer, the receptive field of GNN can be expanded to more relevant nodes, which may be far away from the central nodes. On the other hand, GNN can help Transformer capture the complex graph topology information and help aggregate more relevant nodes from the neighborhoods efficiently.

Specifically, we propose a new framework that combines the GNNs and Transformer named **TransGNN**, which is shown in Figure 1. In order to lower down the complexity and filter out the irrelevant nodes influence, we first propose to sample attention nodes for every central node based on semantic and structure information. Then we propose three kinds of positional encodings: (i) shortest path based positional encoding, (ii) degree based positional encoding, (iii) PageRank based positional encoding. These positional encoding can encode the structure topology information into the raw attributes, which helps obtain the simple graph structure information for Transformer. In addition, we design TransGNN module where Transformer and GNNs are used alternately to improve each other. As for the GNN layer, the Transformer can aggregate the attention samples information with low complexity to help expand the receptive field of GNNs to the most relevant nodes. As for Transformer, the message passing mechanism of the GNNs helps fuse representations and graph structure to capture more topology information for Transformer. More relevant nodes information from neighborhoods can also be achieved by the message passing process efficiently. At last, we propose two efficient ways to update the attention samples, which can easily be generalized to medium and large-scale graphs. We analyze the expressive ability and complexity of TransGNN, which shows that TransGNN is more expressive than GNNs only with extra linear complexity. We conduct extensive experiments on many datasets among node and graph classification tasks where TransGNN outperforms state-of-the-art GNN baselines and Transformer baselines in node classification tasks and achieves the competitive performance with Graphormer in graph classification tasks.

Our contribution can be summarized as follows:

- We propose a new model called TransGNN where Transformer and GNNs help each other. Transformer expands the receptive field of GNN and GNN captures the structure information for Transformer. Three positional encodings are used to improve the Transformer by capturing the graph structure.
- In order to lower down the complexity, we propose a sampling strategy together with two efficient ways to update the relevant samples, which can be easily used in medium and large scale graphs.
- We conduct a theoretical analysis on the expressive ability and complexity of TransGNN, which shows that TransGNN is more powerful than GNN with only low extra overhead.
- We conduct extensive evaluations to illustrate that TransGNN can outperform many state-of-the-art baselines on different tasks.

2 RELATED WORK

2.1 Graph Neural Networks

Graph Neural Networks (GNNs) have achieved success in many graph representation learning scenarios such as molecules [38, 58], social networks [10] and recommendation systems [16]. Pioneering work was proposed to generalize the CNNs to signals defined on graph structure data, which bases on the theory of graph signal process [65]. Many spectral based methods [5, 13] are proposed to design different **filter** for graph signals. GCN [29] approximates the first-order Chebyshev polynomial filters and simplifies the process of message passing mechanism, which confer greater flexibility to design a GNN model under an easy understanding way. GraphSAGE [23], GAT [45], GIN [50] design different message passing mechanism which makes information aggregation more effective. Although GNNs have achieved state-of-the-art performance in many applications, **the limited receptive field compromises their power. Shallow GNNs can only aggregate nearby information, which shows strong structure bias and noise, while deep GNNs suffer from the over-smoothing problem and aggregate much irrelevant information** [42].

2.2 Transformer

Transformer [44] was first proposed in the field of machine translation, and the kernel idea behind Transformer is **the attention mechanism. Each transformer encoder layer will calculate attention distribution for every token and then aggregate the information of other tokens weighted by the attention scores.** As every token can aggregate information from all the tokens, Transformer can capture the **long-term dependence within the sequence data.** Because of the marvelous effectiveness in representation learning, Transformer quickly displaces the Recurrent Neural Networks and convolution neural networks and becomes the new state-of-the-art model in many other tasks.

Many existing works exert their efforts to generalize the Transformer architecture to graph data. However the main problems they encounter are (1) The design of node-wise positional encoding. (2) The computational expensive calculation of pairwise attention on large graphs. For positional encoding, Laplacian Encoding [15], and Random Walk have been studied both theoretically and empirically. With respect to the scalability problem, some work try to restrict the receptive field from global to local, for example, ADSF [60] introduces random walks to generate high order local receptive field, and the GAT [45] is the extreme scenario where each node only sees its one-hop neighbor. Other work try to scale down the

graph [19]. For example, Coarformer [34] first coarse the graph, then apply the Transformer architecture.

3 METHOD

In this section, we first show the framework of TransGNN, and then we elaborate on the detail of each component of TransGNN. At last, we discuss the expressive ability of TransGNN theoretically and analyze the complexity.

3.1 Model Framework

The framework of TransGNN is shown in Figure 1, which consists of three important components: (1) attention sampling module, (2) positional encoding module, (3) TransGNN module. We first sample the most relevant nodes for each central node by considering the semantic similarity and graph structure information in the attention sampling module. Then in the positional encoding module, we calculate the positional encoding to help the Transformer capture the graph topology information. After these two modules, we use the TransGNN module, which contains three sub-modules in order: (i) Transformer layer, (ii) GNN layer, (iii) samples update sub-module. Among them, the Transformer layer is used to expand the receptive field of the GNN layer and aggregate the attention samples information efficiently, while the GNN layer helps the Transformer layer perceive the graph structure information and obtain more relevant information of neighbor nodes. The samples update sub-module is used to update the attention samples efficiently upon new representations.

3.2 Attention Sampling Module

Calculating attention on the whole graph data has two disadvantages: (i) The attention calculation results in the $O(N^2)$ complexity, which is unrealistic when graphs become large. (ii) Irrelevant nodes are considered under the global attention setting.

In graph data, we argue that there is no need to calculate attention on the whole graph for every node, and it is enough to consider the most relevant ones, which not only decreases complexity but also filters out the noisy nodes' information. Therefore, we propose to sample the most relevant nodes for every central node in the attention sampling module. To achieve this, we first calculate the semantic similarity matrix:

$$S = XX^T, \quad (1)$$

where X is the nodes' attributes. However, through S we can only get the raw semantic similarity without taking the structure influence into consideration. The neighbor nodes' preference also has an influence on the central node because, after the message passing mechanism, the information will be passed to the neighbor nodes. Therefore we also consider the neighbor nodes' preference before sampling. We use the following equation to update the similarity matrix to contain the neighbor nodes preference:

$$S = S + \alpha \hat{A}S, \quad (2)$$

where α is the balance factor, and in this paper, we set α as 0.5. $\hat{A} = A + I$ where A is the adjacent matrix and I is the identity matrix. Based on the new similarity matrix S , for every node $v_i \in \mathcal{V}$ in the input graph, we sample the most relevant nodes as its attention samples as follows:

Attention samples: Given an input graph \mathcal{G} and its similarity matrix S , for node v_i in the graph, we define its attention samples as set $\text{Smp}(v_i) = \{v_j | v_j \in V \text{ and } S(i, j) \in \text{top-}k(S(i, :))\}$ where $S(i, :)$ denotes the i -th row of S and the k works as a hyper-parameter which decides how many nodes should be attended attention.

3.3 Positional Encoding Module

In graph data, the structure information is very important for graph representation learning. However, unlike the grid-like data where the sequence order can be easily captured by Transformer, the graph structure information is more complex. In order to capture the graph structure information for Transformer, we propose three kinds of positional encoding: (i) Shortest path hop based positional encoding. (ii) Degree-based positional encoding. (iii) PageRank based positional encoding. The first two consider the topology distance and local structure, while the last one shows the importance decided by topology in medium and large scale graphs.

3.3.1 Shortest Path Hop Based Positional Encoding. Topology distance is an important kind of structural information in graph data that can be used to describe the topology similarity to some extent. For each central node, the topology distance of different attention samples will have different influences. Therefore we propose to calculate positional encoding based on the shortest path hop to describe the topology distance for Transformer. Specifically, we denote the shortest path hop matrix as P and for each node $v_i \in \mathcal{V}$ and its attention sample node $v_j \in \text{Smp}(v_i)$ the shortest path hop is $P(i, j)$, we calculate the shortest path hop based positional encoding (SPE) for every attention sample node v_j as:

$$\text{SPE}(v_i, v_j) = \text{MLP}(P(i, j)), \quad (3)$$

where $\text{MLP}(\cdot)$ is a two layer neural networks in this paper.

3.3.2 Degree Based Positional Encoding. In graph data, the degree of the node can represent the complexity of the neighbor structure around it, whereas the node with a larger degree may have a more complex local structure. The degree distribution in many real-world data has a power-law property that shows the degrees of central nodes and attention samples are likely varied. In order to capture this difference, we propose to use the degree to calculate the positional encoding. Formally, for any node v_i whose degree is deg_i , we calculate the degree based positional encoding (DE) as:

$$\text{DE}(v_i) = \text{MLP}(\text{deg}_i), \quad (4)$$

where $\text{MLP}(\cdot)$ is a two layer neural networks in this paper.

3.3.3 Page Rank Based Positional Encoding. In real-world graphs, different nodes have different importance caused by the topology structure. Page Rank algorithm can label the nodes' importance according to their structural roles in the graph data, where the nodes with higher importance will be labeled with the higher page rank values. In order to obtain the influence of structure importance, we propose to calculate positional encoding based on the page rank value for every node. Formally, for node v_i we denote its page rank value as $\text{Pr}(v_i)$, and we calculate the page rank based positional encoding (PRE) as:

$$\text{PRE}(v_i) = \text{MLP}(\text{Pr}(v_i)), \quad (5)$$

where $\text{MLP}(\cdot)$ is a two layer neural networks in this paper.

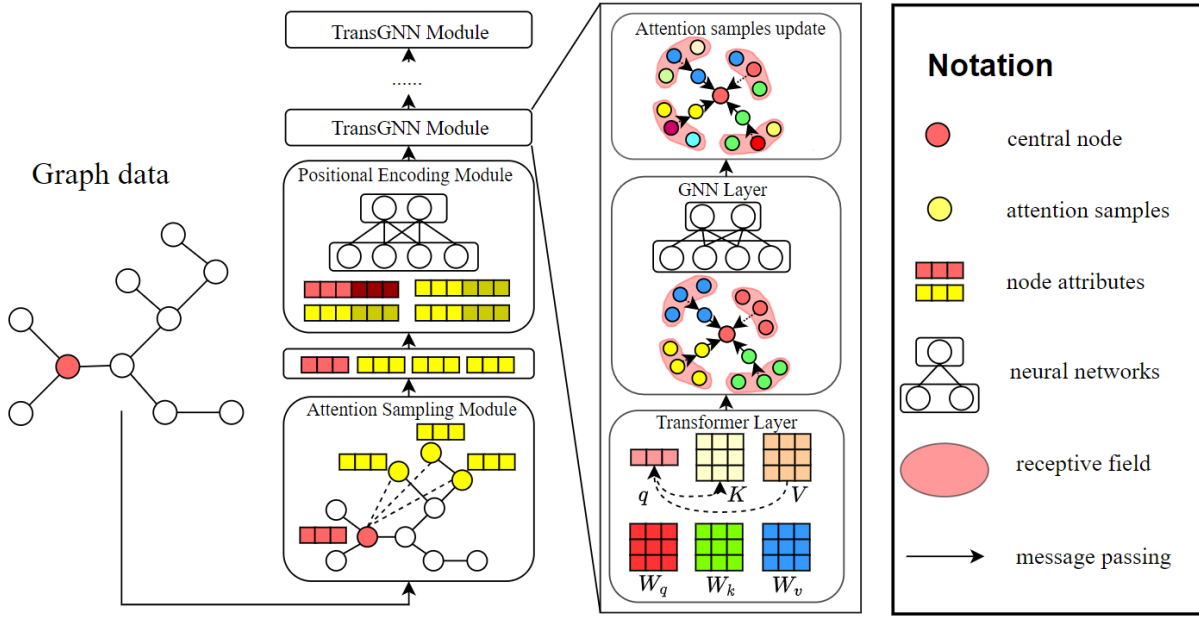


Figure 1: The framework of TransGNN. We first sample relevant nodes for the central nodes, then we calculate positional encoding to enhance the raw attributes by combining the structure information. In the TransGNN module, the Transformer layer and GNN layer improve each other, followed by the samples update sub-module.

Based on the positional encoding defined above, Transformer can capture many different kinds of graph structure information by aggregating them together to the raw node attributes. Specifically, for the central node v_i and its attention samples $\text{Smp}(v_i)$, we aggregate the positional encoding in the following ways:

$$\begin{aligned} h_i &= \text{COMB}(\text{AGG}(\mathbf{x}_i, \text{SPE}(v_i, v_i), \text{DE}(v_i), \text{PRE}(v_i))) \\ h_j &= \text{COMB}(\text{AGG}(\mathbf{x}_j, \text{SPE}(v_i, v_j), \text{DE}(v_j), \text{PRE}(v_j))) \quad v_j \in \text{Smp}(v_i) \end{aligned} \quad (6)$$

where $\mathbf{x}_i, \mathbf{x}_j$ are the raw attributes of v_i, v_j respectively, $\text{AGG}(\cdot)$ is the aggregation function and $\text{COMB}(\cdot)$ is the combination function. In this paper we use two layer neural networks as $\text{COMB}(\cdot)$ and vector concatenation as $\text{AGG}(\cdot)$.

3.4 TransGNN Module

GNNs have limited receptive fields because their message passing mechanism only aggregates information within the neighbor, but deep GNNs have the over-smoothing problem. In the graphs, the relevant nodes may be far from the central node, and the nodes within the neighbor may also contain noise. This compromises the effectiveness of GNNs. However, although Transformer can aggregate the relevant information far away, it is hard for the Transformer to make full use of structure information, and the complexity is also an important problem. In this module, we combine Transformer and GNN to help each other with their strengths, which contains three sub-modules: (i) Transformer layer, (ii) GNN layer, (iii) samples update sub-module.

3.4.1 Transformer Layer. We use the Transformer layer to improve the GNN layer by expanding the receptive field to more relevant nodes which may be far from the neighborhood. In order to lower the complexity and filter out the irrelevant information, we only

consider the most relevant samples for each central node. In the following, we use central node v_i and its attention samples $\text{Smp}(v_i)$ as an example to illustrate the Transformer layer, and for other nodes, this process is the same.

We denote the input of Transformer layer as $\mathbf{H} \in \mathbb{R}^{N \times d_{\text{in}}}$ and the representation of central node v_i is h_i . We stack the representations of attention samples $\text{Smp}(v_i)$ as the matrix $\mathbf{H}_i^{\text{Smp}} \in \mathbb{R}^{k \times d_{\text{in}}}$. We use three matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ to project the corresponding representations to \mathbf{q}, \mathbf{K} and \mathbf{V} respectively and we aggregate the information based on the attention distribution as:

$$\begin{aligned} \mathbf{q} &= h_i \mathbf{W}_q \\ \mathbf{K} &= \mathbf{H}_i^{\text{Smp}} \mathbf{W}_k \\ \mathbf{V} &= \mathbf{H}_i^{\text{Smp}} \mathbf{W}_v \\ \mathbf{a}_t &= \frac{\mathbf{q} \mathbf{K}^T}{\sqrt{d_{\text{out}}}} \\ h_i &= \text{softmax}(\mathbf{a}_t) \mathbf{V}, \end{aligned} \quad (7)$$

where \mathbf{q} is the representation of the query and \mathbf{K}, \mathbf{V} are the representation of keys and values. \mathbf{a}_t is the attention distribution vector. This process can be expanded to multi-head attention as:

$$\text{MultiHead}(\mathbf{h}_i) = \text{Concat}(\text{head}_1, \dots, \text{head}_m) \mathbf{W}_m, \quad (8)$$

where m is the head number, $\text{Concat}(\cdot)$ denotes the concatenate function and \mathbf{W}_m is the projection matrix, each head is calculated as h_i in Equation 7.

3.4.2 GNN Layer. GNN layer is utilized to fuse the representations and graph structure to help the Transformer layer make better use of the graph structure. Besides, the relevant nodes' information

of the neighborhood can be captured efficiently by the message passing process of the GNN layer. We use node v_i as an example, and the message passing process of the GNN layer can be described as:

$$\begin{aligned} \mathbf{h}_M(v_i) &= \text{Message}(\mathbf{h}_k, \forall v_k \in \mathcal{N}(v_i)) \\ \mathbf{h}_i &= \text{Combine}(\mathbf{h}_i, \mathbf{h}_M(v_i)), \end{aligned} \quad (9)$$

where $\mathcal{N}(v_i)$ is the **neighbor nodes set** of v_i , h_i, h_k are the representations of v_i, v_k respectively. $\text{Message}(\cdot)$ and $\text{Combine}(\cdot)$ are the message passing function and aggregation function defined by GNN layer.

3.4.3 Samples Update Sub-Module. After the Transformer layer and GNN layer, the attention samples should be updated upon new representations. However, calculating the similarity matrix will result in a $O(N^2)$ complexity. We propose two efficient samples update strategies in this sub-module to update the attention samples.

(i) *Random Walk based Update.* Considering the localness of the graph data where similar nodes are more likely to be contained in the neighborhood, we propose to update the attention samples by exploring the attention samples neighborhoods to find the possible new relevant nodes. We use the random walk strategy to **explore the local neighborhood of every sampled node**. Specifically, the transfer probability of the random walk is calculated based on the similarity as:

$$p_{i \rightarrow j} = \begin{cases} \frac{\mathbf{h}_i \mathbf{h}_j^T}{\sum_{l \in \mathcal{N}(v_i)} \mathbf{h}_i \mathbf{h}_l^T}, & \text{if } v_j \in \mathcal{N}(v_i) \\ 0, & \text{if } v_j \notin \mathcal{N}(v_i) \end{cases} \quad (10)$$

The transfer probability can be calculated efficiently in the message passing process. Based on the transfer probability, we walk **a node sequence with length L** for each attention sample, and then we choose the new attention samples among the explored nodes upon new representations.

(ii) *Message Passing based Update.* The random walk-based update strategy has extra overhead. We propose another update strategy that utilizes the message passing of the GNN layer to update the samples without extra overhead. Specifically, we aggregate the attention samples from the neighbor nodes for each central node in the message passing process of the GNN layer. The intuition behind this is that the attention samples of the neighbors may also be the relevant attention samples of the central nodes. We denote the attention samples set of the neighbor nodes as the attention message, which is defined as follows:

$$\text{Attn_Msg}(v_i) = \bigcup \text{Smp}(v_j) \quad \forall v_j \in \mathcal{N}(v_i), \quad (11)$$

thus we choose the new attention samples among $\text{Attn_Msg}(v_i)$ for node v_i based on the new representations.

3.5 Complexity Analysis

We analyze the complexity of TransGNN. The overhead of the attention sampling module and the positional encoding module can be attributed to the **data pre-processing stage**. The complexity of the attention sampling module is $O(N^2)$, and the most complex part of the positional encoding module is the calculation of the shortest hop matrix P . Considering the graphs in real applications

are sparse and have positive edge weights, Johnson algorithm [1] can be used. With the help of the heap optimization, the time and space complexity can be reduced to $O(N(N+E) \log E)$ where N is the nodes number, and E is the edge number. The extra overhead of the TransGNN module compared with GNNs mainly focus on the Transformer layer and attention samples update. The extra complexity caused by the Transformer layer is $O(Nk)$ where k is the attention samples number, and the extra complexity of samples update is $O(Nkd_a)$ for message passing mechanism where d_a is the average degree. (the extra complexity will be $O(NkL)$ if we use random walk-based update). **Therefore, we show that TransGNN has at most $O(N(N+E) \log E)$ data pre-processing complexity and linear extra complexity compared with GNNs because kd_a is a constant and $kd_a \ll N$.**

3.6 Theoretical Analysis

Here we illustrate the expression power of TransGNN. The conclusion is given by the following two theorems with their proofs.

THEOREM 1. *TransGNN has at least the expression ability of GNN, and any GNN can be expressed by TransGNN.*

PROOF. If we add **attention mask** as top 1 mask, the Eqn of Transformer layer will become:

$$\mathbf{H}_{\text{out}} = \frac{1}{\sqrt{d_{\text{out}}}} \mathbf{H}(\mathbf{W}_v + \mathbf{I}) \quad (12)$$

We use the GCN layer as an example and the message passing can be derived as:

$$\begin{aligned} \mathbf{H} &= \sigma(\mathbf{A}\mathbf{H}_{\text{out}}\mathbf{W}) \\ &= \sigma(\mathbf{A} \frac{1}{\sqrt{d_{\text{out}}}} \mathbf{H}(\mathbf{W}_v + \mathbf{I})\mathbf{W}), \end{aligned} \quad (13)$$

where $\sigma(\cdot)$ is the activation function and if we set \mathbf{W}_v as **diagonal matrix with the diagonal value as $\sqrt{d_{\text{out}}} - 1$** and the Equation 13 will become:

$$\mathbf{H} = \sigma(\mathbf{A}\mathbf{H}\mathbf{W}) \quad (14)$$

Therefore, TransGNN has at least the expression ability of GNN. \square

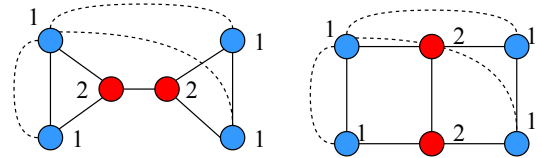


Figure 2: Example illustration. The dotted line denotes the receptive field of the Transformer layer, and the color denotes the similarity relationship.

THEOREM 2. *TransGNN can be more expressive than 1-WL Test.*

PROOF. With the help of Transformer layer and positional encoding, TransGNN can aggregate more relevant information and structure information to improve the message passing process, which can be more expressive than 1-WL Test [14]. We give an illustration in Figure 2. These two graphs cannot be distinguished by 1-WL Test. However, because the **transformer layer expand the**

receptive field and the positional encoding can capture the local structure, they can be distinguished by TransGNN. For example the node in the top left gets the shortest path information $\{0,1,3,3\}$ and $\{0,1,2,3\}$ respectively. \square

4 EXPERIMENTS

In this section, we conduct experiments on many benchmark datasets among node classification tasks and graph classification tasks.

Datasets	Nodes	Edges	Attributes	Classes
PubMed	19,717	88,648	500	3
Amazon-Photo	7,650	238,162	745	8
Amazon-Computer	13,752	491,722	767	10
OGB-Arxiv	169,343	1,166,243	128	40

Table 1: Statistics of the node classification datasets

Datasets	Graphs	Avg Nodes	Avg Edges	Tasks
Molbase	1,513	34.1	73.7	1
Moltox21	7,831	18.6	38.6	12
Molbbbp	2,039	24.1	51.9	1
Molhiv	41,127	25.5	54.9	1

Table 2: Statistics of the graph classification datasets

4.1 Datasets and Metrics

We use four datasets for the node classification task, and we use four datasets for the graph classification task. The datasets are described as follows:

- **Nodes classification datasets:** We use PubMed [52], Amazon-Photo, Amazon-Computer [43], OGB-Arxiv [25] for nodes classification task. For PubMed, Amazon-Photo, Amazon-Computer, we randomly split 10% nodes as training nodes, 70% nodes as test nodes, and 10% nodes as validation nodes. And for OGB-Arxiv, we use the public split by [25] to generate training/test/validation set. For all the datasets, We use accuracy as the metric. We use four datasets for the node classification task, and we use four datasets for the graph classification task. The datasets are described as follows:
- **Graph classification datasets:** We use Molbase, Moltox21, Molbbbp, Molhiv for graph classification task. For all these datasets, we adopt the public split and we use RDKit¹ to pre-process them as their official settings. We use ROC-AUC as the metric for all these datasets.

The information of the node classification datasets and graph classification datasets can be found in Table 3 and Table 4 respectively.

¹Open-source cheminformatics; <http://www.rdkit.org>

4.2 Baselines

We compare our method with many state of the art baselines, which can be divided into two categories: GNN based baselines and Transformer based baselines. For GNN based baselines we compare with the following models:

- **GCN:** We use two layers GCN [29] with hidden dimension is 64 for all the datasets, and we train GCN in an end-to-end manner in the training set, and validation set is used for early stop with patience is 10. For the node classification task, we report the accuracy on the test nodes directly, and for the graph classification task, we use the mean pooling after getting the nodes' representations and classify the graph with a linear head.
- **GraphSAGE:** We use two layers GraphSAGE [23] model, and we sample 10, 10 neighbor nodes for each layer. The hidden dimension is 64 for all the datasets, and we train GraphSAGE in the same way as GCN. We use the mean pooling and linear classification head for the graph classification task.
- **GAT:** We use two layers GAT [45] model, and we use multi heads attention with the head number is 8 for each layer. The hidden dimension is 64 for all the datasets. The training strategy, pooling function, and classification head are the same as the GCN model.
- **GCNII:** On PubMed, we apply the setting suggested by [8], and for other datasets, we train a 64 layer model, and each of the dimensions is 64. We also apply an early stop mechanism with a patience of 10.
- **APPNP:** The experiment setting on PubMed is set according to [31], and for other datasets, we search the hyperparameter α within $\{0.1, 0.2, 0.5, 0.9\}$.
- **GPRGNN:** We conduct the experiment on Pubmed according to [11]. For other datasets, We set the length of the random walk path to $K = 10$ and adopt a 2-layer (MLP) with 64 hidden units for the Neural Network component. For the GPR weights, we choose α within $\{0.1, 0.2, 0.5, 0.9\}$.

For Transformer based baselines we compare with the following models:

- **Graph-Bert:** We use the official implementation of Graph-Bert [59], and for PubMed, we set the sub-graph size as 30, which is the official setting, and for other datasets, we use the grid search to find the best sub-graph size. Other settings are the same as the official implementation. We use the same training strategy as GCN.
- **Graphormer:** We compare our method with Graphormer [53] only in graph classification task because nodes classification task in medium and large scale graphs will result in out-of-memory for Graphormer.

4.3 Setting of TransGNN

For PubMed and OGB-Arxiv, we set attention samples number k as 15, and for Amazon-Photo and Amazon-computer, we set k as 5. We use three Transformer layers with two GNN layers sandwiched between them. For the transformer layer, multi-head attention is used, and we set the head number as 8 for all the datasets. For the GNN layer, we test GCN, GraphSAGE, GAT, GIN separately (we denote these backbones as TransGCN, TransSAGE, TransGAT, and

	PubMed	Amazon-Photo	Amazon-Computer	OGB-Arxiv
GCN	83.79±0.34%	90.46±0.17%	87.20±0.23%	71.74±0.29%
GraphSAGE	83.81±0.25%	90.63±0.31%	86.93±0.37%	71.21±0.27%
GAT	83.02±0.12%	90.69±0.08%	87.44±0.21%	71.49±0.12%
SGCN	82.79±0.63%	90.02±0.43%	87.31±0.17%	71.95±0.21%
GCNII	84.89±0.49%	91.91±0.27%	88.12±0.08%	72.74±0.16%
APPNP	85.02±0.35%	91.66±0.46%	87.01±0.21%	71.78±0.19%
GPRGNN	84.56±0.45%	92.91±0.26%	88.29±0.44%	71.76±0.23%
Graph-Bert	83.91±0.37%	91.88±0.40%	86.80±0.36%	71.32±0.34%
TransSAGE-RW	86.36±0.51%	93.63±0.49%	88.88±0.42%	72.12±0.41%
TransSAGE-MP	86.92±0.45%	93.89±0.36%	90.20±0.35%	72.33±0.22%
TransGCN-RW	86.23±0.26%	92.95±0.58%	89.34±0.33%	72.61±0.34%
TransGCN-MP	87.53±0.37%	94.05±0.29%	90.07±0.47%	73.45±0.41%
TransGAT-RW	86.58±0.35%	92.16±0.37%	89.92±0.61%	71.59±0.26%
TransGAT-MP	87.19±0.25%	93.36±0.25%	89.66±0.28%	71.87±0.23%

Table 3: Performance of node classification task. The best result is bolded.

	ogbg-molbace	ogbg-moltox21	ogbg-molbbbp	ogbg-molhiv
GCN	78.12	72.98	65.23	76.06
GraphSAGE	78.95	73.10	64.58	75.35
GAT	77.78	72.41	65.31	74.79
GIN	79.43	73.32	66.99	75.58
Graphormer	85.12	76.81	71.32	80.51
TransGCN	82.56	74.89	70.02	78.78
TransGAT	81.62	74.31	69.56	78.41
TransGIN	83.41	75.51	69.91	79.45

Table 4: Performance of graph classification task. The best two results are bold.

TransGIN, respectively). The two update strategies are denoted as -RW(random walk) and -MP(message passing), respectively. The learning rate is set as 0.001. We use Adam to optimize our model with a weight decay of 0. For the node classification task, we train TransGNN in an end-to-end manner, and a validation set is used for early stop with a patience of 10. For the graph classification task, we calculate full attention for every central node without sampling because the graphs are relatively small. We use mean pooling and linear classification heads to classify the graph.

4.4 Node Classification Task

We first show the performance on nodes classification task, the results can be found in Table 3. From the results we have the following observations:

- TransGNN achieves the best performance across all the datasets on the node classification task, demonstrating that the representations got by the TransGNN are more distinguishable than the state-of-the-art baselines.
- Compared with the corresponding GNN models, TransGNN gets better performance, demonstrating that Transformer can help improve the GNNs by expanding the receptive field of GNNs to focus on more relevant nodes.

- Compared with other Transformer models in the graph, TransGNN can be used in medium- and large-scale graphs because of the efficient sampling and update strategies. Besides, TransGNN can achieve better performance because, with the help of GNNs and positional encoding, Transformer can make better use of graph structure information.
- By comparing different sampling update strategies, we find that the random walk-based update strategy is not as good as the message passing based update strategy. The possible reason is that the noise in the edges may lead to walking to the irrelevant nodes.

4.5 Graph Classification Task

We then show the performance on graph classification task, the results can be found in Table 4. From the results we have the following observations:

- TransGNN achieves improved performance than GNN baselines across all the datasets. This demonstrates that the Transformer layer can help improve the graph representations obtained by the GNN model.
- Compared with Graphormer, TransGNN can achieve competitive results on all the datasets. The most important reason why TransGNN performs not as well as Graphormer is:

Method	Accuracy
TransSAGE-MP(No PE)	85.75
TransSAGE-MP(No SAGE)	83.02
TransSAGE-MP(No Trans)	84.60
TransSAGE(No MP)	84.97
TransSAGE-MP	86.69

Table 5: Influence of different modules. No PE means no positional encoding. No SAGE means no GNN layer. No Trans means no Transformer layer. No MP means no message passing update.

Graphormer can make better use of edge information which is very important for molecular graph representation learning. However, with the help of the GNN layer, TransGNN can be easily generalized to medium and large-scale graphs, while Graphormer will suffer from the out-of-memory problem.

- The performance on different GNN backbones of TransGNN shows that the GNN layer has a nontrivial influence on the graph representations.

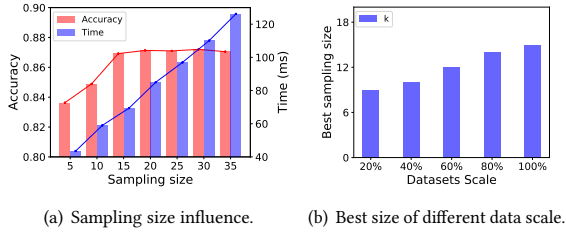


Figure 3: Different sampling size influence and best sampling size for different graph scale.

4.6 Attention Sampling Study

We study the influence of **attention sampling size** using PubMed as an example. The results can be found in Figure 3. In Figure 3(a) we vary the k from 5 to 35 and we find that sampling fewer nodes will compromise the performance while excessive nodes sampling will result in extra overhead but no improvement. A relative small sampling size is enough to get a good performance. In Figure 3(b) we change the graph scale and show the best sampling size. We find that we only need to increase a small number of samples when the graph becomes large greatly.

4.7 Ablation Study

In order to show the influence of different modules, we use PubMed as an example and GraphSAGE as the backbone GNN to report the performance when ignoring different modules. All the settings are the same as the default. The results can be found in Table 5. From the results, we have the following observations:

- Without positional encoding, TransGNN achieves compromised results because the Transformer layer can not capture structure information directly.
- No GNN layer nor Transformer layer will result in a significant decrease in performance. It illustrates that the GNN layer and Transformer layer can be combined properly and help each other.
- Without message passing update strategy TransGNN also has a drop in the performance because there are irrelevant nodes after the change of the representations. Besides, the static attention samples are also incomplete.

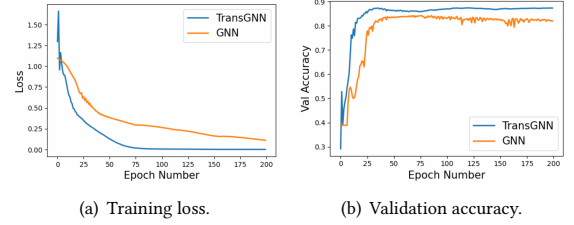


Figure 4: Training process of TransGNN and GNN.

4.8 Convergence of TransGNN

We study the training process of TransGNN and GNN on the PubMed dataset. GraphSAGE is used as the backbone of TransGNN, and the result is shown in Figure 4. From the result, we find that compared with GNN, the loss of TransGNN decreases faster, and the number of epochs required for convergence is less. This demonstrates that the training of TransGNN can converge faster than the training process of GNN.

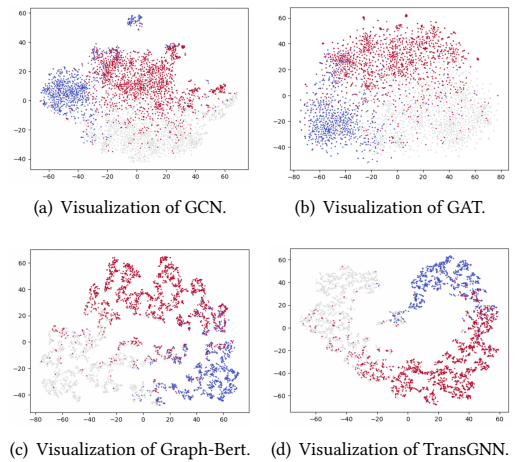


Figure 5: Visualization of different baselines.

4.9 Visualization

We use PubMed as an example to visualize the representations of different baselines. For GNNs and Graph-Bert, we chose the output of the second layer as the representations, and for TransGNN(TransSAGE), we chose the output of the second GNN layer as representations. We randomly choose 20% of the nodes for visualization. The results are shown in Figure 5. From the results, we can find that the cluster property of the representations produced by GNNs is not as good as that of Transformer models. The possible reasons are: (i) GNNs rely too much on edges, where noisy edges will lead to irrelevant information. (ii) Some relevant nodes outside the receptive field are not considered by the central nodes. Besides, we also find that compared with the Graph-Bert, TransGNN can cluster the relevant nodes better, because: (i) TransGNN can aggregate the most relevant nodes information. (ii) With the help of the GNN layer, TransGNN can make use of structure information better than Graph-Bert.

5 CONCLUSION

In this paper, we propose TransGNN to help GNN expand its receptive field with low overhead. We first use three kinds of **positional encoding to capture the structure information** for the transformer layer. Then the Transformer layer and GNN layer are used alternately to **make every node focus on the most relevant samples**. Two efficient sample update strategies are proposed for medium- and large-scale graphs. Experiments on eight datasets show the effectiveness of TransGNN compared to the state-of-the-art baselines.

REFERENCES

- [1] Hamid Allaoui and Abdelhakim Artiba. 2009. Johnson’s algorithm: A key to solve optimally or approximately flow shop scheduling problems with unavailability periods. *International Journal of Production Economics* 121, 1 (2009), 81–87.
- [2] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. <https://openreview.net/forum?id=i80OPhOCVH2>
- [3] Muhammet Balcilar, Pierre Héroux, Benoit Gaüzère, Pascal Vasseur, Sébastien Adam, and Paul Honeine. 2021. Breaking the Limits of Message Passing Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 599–608. <http://proceedings.mlr.press/v139/balcilar21a.html>
- [4] Rémy Brossard, Oriel Frigo, and David Dehaene. 2020. Graph convolutions that can finally model local structure. *CoRR abs/2011.15069* (2020). [arXiv:2011.15069](https://arxiv.org/abs/2011.15069) <https://arxiv.org/abs/2011.15069>
- [5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations*.
- [6] Chen Cai and Yusu Wang. 2020. A Note on Over-Smoothing for Graph Neural Networks. *CoRR abs/2006.13318* (2020). [arXiv:2006.13318](https://arxiv.org/abs/2006.13318) <https://arxiv.org/abs/2006.13318>
- [7] Changyu Chen, Xiting Wang, Yiqiao Jin, Victor Ye Dong, Li Dong, Jie Cao, Yi Liu, and Rui Yan. 2023. Semi-Offline Reinforcement Learning for Optimized Text Generation. In *ICML*.
- [8] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1725–1735.
- [9] Nuo Chen, Fenglin Liu, Chenyu You, Peilin Zhou, and Yuexian Zou. 2021. Adaptive bi-directional attention: Exploring multi-granularity representations for machine reading comprehension. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 7833–7837.
- [10] Justin Cheng, Jon M. Kleinberg, Jure Leskovec, David Liben-Nowell, Bogdan State, Karthik Subbian, and Lada A. Adamic. 2018. Do Diffusion Protocols Govern Cascade Growth?. In *ICWSM*. AAAI Press, 32–41.
- [11] Eli Chien, Jianhao Peng, Pan Li, and Olga Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.
- [12] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. 2020. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems* 33 (2020), 13260–13271.
- [13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.
- [14] Brendan L. Douglas. 2011. The weisfeiler-lehman method and graph isomorphism testing. *arXiv preprint arXiv:1101.5211* (2011).
- [15] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699* (2020).
- [16] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, Ling Liu, Ryan W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (Eds.). ACM, 417–426. <https://doi.org/10.1145/3308558.3313488>
- [17] Jiayan Guo, Lun Du, Wendong Bi, Qiang Fu, Xiaojun Ma, Xu Chen, Shi Han, Dongmei Zhang, and Yan Zhang. 2023. Homophily-oriented Heterogeneous Graph Rewiring. In *Proceedings of the ACM Web Conference 2023*. 511–522.
- [18] Jiayan Guo, Lun Du, Xu Chen, Xiaojun Ma, Qiang Fu, Shi Han, Dongmei Zhang, and Yan Zhang. 2023. On Manipulating Signals of User-Item Graph: A Jacobi Polynomial-based Graph Collaborative Filtering. *arXiv preprint arXiv:2306.03624* (2023).
- [19] Jiayan Guo, Lun Du, and Hengyu Liu. 2023. GPT4Graph: Can Large Language Models Understand Graph Structured Data? An Empirical Evaluation and Benchmarking. *arXiv preprint arXiv:2305.15066* (2023).
- [20] Jiayan Guo, Shangyang Li, and Yan Zhang. 2023. An Information Theoretic Perspective for Heterogeneous Subgraph Federated Learning. In *International Conference on Database Systems for Advanced Applications*. Springer, 745–760.
- [21] Jiayan Guo, Shangyang Li, Yue Zhao, and Yan Zhang. 2022. Learning robust representation through graph adversarial contrastive learning. In *International Conference on Database Systems for Advanced Applications*. Springer, 682–697.
- [22] Jiayan Guo, Peiyan Zhang, Chaozhao Li, Xing Xie, Yan Zhang, and Sunghun Kim. 2022. Evolutionary Preference Learning via Graph Nested GRU ODE for Session-based Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 624–634.
- [23] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems*.
- [24] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.
- [25] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. 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*, Hugo Larochelle, Marc’Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). <https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cf84fd0-Abstract.html>
- [26] Juyong Jiang, Jae Boum Kim, Yingtao Luo, Kai Zhang, and Sunghun Kim. 2022. AdaMCT: adaptive mixture of CNN-transformer for sequential recommendation. *arXiv preprint arXiv:2205.08776* (2022).
- [27] Yiqiao Jin, Yunsheng Bai, Yanqiao Zhu, Yizhou Sun, and Wei Wang. 2022. Code Recommendation for Open Source Software Developers. In *Proceedings of the ACM Web Conference 2023*.
- [28] Yiqiao Jin, Yeon-Chang Lee, Kartik Sharma, Meng Ye, Karan Sikka, Ajay Divakaran, and Srijan Kumar. 2023. Predicting Information Pathways Across Online Communities. In *KDD*.
- [29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*.
- [30] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997* (2018).
- [31] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR (Poster)*. OpenReview.net.
- [32] Johannes Klicpera, Shankari Giri, Johannes T Margraf, and Stephan Günnemann. 2020. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115* (2020).
- [33] Johannes Klicpera, Janek Groß, and Stephan Günnemann. 2020. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123* (2020).
- [34] Weirui Kuang, Zhen WANG, Yaliang Li, Zhewei Wei, and Bolin Ding. 2022. Coarformer: Transformer for large graph via graph coarsening. https://openreview.net/forum?id=flkjO_FKVzw

- [35] Yuxuan Lei, Xiaolong Chen, Defu Lian, Peiyan Zhang, Jianxun Lian, Chaozhao Li, and Xing Xie. 2023. Practical Content-aware Session-based Recommendation: Deep Retrieve then Shallow Rank. In *Amazon KDD Cup 2023 Workshop*.
- [36] Guohao Li, Matthias Müller, Ali K. Thabet, and Bernard Ghanem. 2019. Deep-GCNs: Can GCNs Go As Deep As CNNs?. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 9266–9275. <https://doi.org/10.1109/ICCV.2019.00936>
- [37] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3538–3545.
- [38] Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S. Zemel. 2019. Lanczos-Net: Multi-Scale Deep Graph Convolutional Networks. In *ICLR (Poster)*. OpenReview.net.
- [39] Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149v2* (2023).
- [40] Xiaojun Ma, Qin Chen, Yuanyi Ren, Guojie Song, and Liang Wang. 2022. Meta-Weight Graph Neural Network: Push the Limits Beyond Global Homophily. In *Proceedings of the ACM Web Conference 2022*. 1270–1280.
- [41] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. 2019. Provably Powerful Graph Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 2153–2164. <https://proceedings.neurips.cc/paper/2019/hash/bb04af0f7ecae4aa62035497da1387-Abstract.html>
- [42] Kenta Oono and Taiji Suzuki. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- [43] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).
- [44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*.
- [45] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *Proceedings of the 6th International Conference on Learning Representations*.
- [46] Haohan Wang, Peiyan Zhang, and Eric P Xing. 2020. Word shape matters: Robust machine translation with visual embedding. *arXiv preprint arXiv:2010.09997* (2020).
- [47] Junshan Wang, Guojie Song, Yi Wu, and Liang Wang. 2020. Streaming Graph Neural Networks via Continual Learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1515–1524.
- [48] Yueqi Xie, Jingqi Gao, Peilin Zhou, Qichen Ye, Yining Hua, Jaeboum Kim, Fangzhao Wu, and Sunghun Kim. 2023. Rethinking Multi-Interest Learning for Candidate Matching in Recommender Systems. *arXiv preprint arXiv:2302.14532* (2023).
- [49] Yueqi Xie, Peilin Zhou, and Sunghun Kim. 2022. Decoupled Side Information Fusion for Sequential Recommendation. In *International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.
- [50] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. 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.
- [51] Ruichao Yang, Xiting Wang, Yiqiao Jin, Chaozhao Li, Jianxun Lian, and Xing Xie. 2022. Reinforcement subgraph reasoning for fake news detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2253–2262.
- [52] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.), Vol. 48.
- [53] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanning Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation? *Advances in Neural Information Processing Systems* 34 (2021).
- [54] Chenyu You, Nuo Chen, Fenglin Liu, Shen Ge, Xian Wu, and Yuexian Zou. 2022. End-to-end spoken conversational question answering: Task, dataset and model. *arXiv preprint arXiv:2204.14272* (2022).
- [55] Chenyu You, Weicheng Dai, Yifei Min, Fenglin Liu, Xiaoran Zhang, Chen Feng, David A Clifton, S Kevin Zhou, Lawrence Hamilton Staib, and James S Duncan. 2023. Rethinking semi-supervised medical image segmentation: A variance-reduction perspective. *arXiv preprint arXiv:2302.01735* (2023).
- [56] Chenyu You, Weicheng Dai, Yifei Min, Lawrence Staib, and James S Duncan. 2023. Implicit Anatomical Rendering for Medical Image Segmentation with Stochastic Experts. *arXiv preprint arXiv:2304.03209* (2023).
- [57] Chenyu You, Jinlin Xiang, Kun Su, Xiaoran Zhang, Siyuan Dong, John Onofrey, Lawrence Staib, and James S Duncan. 2022. Incremental learning meets transfer learning: Application to multi-site prostate mri segmentation. In *International Workshop on Distributed, Collaborative, and Federated Learning*. Springer, 3–16.
- [58] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. 2018. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*.
- [59] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations. *CoRR* abs/2001.05140 (2020). [arXiv:2001.05140](https://arxiv.org/abs/2001.05140) <https://arxiv.org/abs/2001.05140>
- [60] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. 2020. Adaptive Structural Fingerprints for Graph Attention Networks. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=BJxWx0NYPr>
- [61] Peiyan Zhang, Jiayan Guo, Chaozhao Li, Yueqi Xie, Jae Boum Kim, Yan Zhang, Xing Xie, Haohan Wang, and Sunghun Kim. 2023. Efficiently leveraging multi-level user intent for session-based recommendation via atten-mixer network. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 168–176.
- [62] Peiyan Zhang and Sunghun Kim. 2023. A Survey on Incremental Update for Neural Recommender Systems. *arXiv preprint arXiv:2303.02851* (2023).
- [63] Peiyan Zhang, Haoyang Liu, Chaozhao Li, Xing Xie, Sunghun Kim, and Haohan Wang. 2023. Foundation Model-oriented Robustness: Robust Image Model Evaluation with Pretrained Models. *arXiv preprint arXiv:2308.10632* (2023).
- [64] Peiyan Zhang, Yuchen Yan, Chaozhao Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. 2023. Continual Learning on Dynamic Graphs via Parameter Isolation. *arXiv preprint arXiv:2305.13825* (2023).
- [65] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1 (2020), 57–81. <https://doi.org/10.1016/j.aiopen.2021.01.001>
- [66] Peilin Zhou, Jingqi Gao, Yueqi Xie, Qichen Ye, Yining Hua, and Sunghun Kim. 2022. Equivariant Contrastive Learning for Sequential Recommendation. *arXiv preprint arXiv:2211.05290* (2022).
- [67] Peilin Zhou, Zhiqi Huang, Fenglin Liu, and Yuexian Zou. 2021. PIN: A novel parallel interactive network for spoken language understanding. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2950–2957.