Graph External Attention Enhanced Transformer

Anonymous Authors[1]

Abstract

The Transformer architecture has recently gained considerable attention in the field of graph representation learning, as it naturally overcomes several limitations of Graph Neural Networks (GNNs) with customized attention mechanisms or positional and structural encodings. Despite making some progress, existing works tend to overlook external information of graphs, specifically the correlation between graphs. Intuitively, graphs with similar structures should have similar representations. Therefore, we propose Graph External Attention (GEA) — a novel attention mechanism that leverages multiple external node/edge key-value units to capture inter-graph correlations implicitly. On this basis, we design an effective architecture called Graph External Attention Enhanced Transformer (GEAET), which integrates local structure and global interaction information for more comprehensive graph representations. Extensive experiments on benchmark datasets demonstrate that GEAET achieves state-of-the-art empirical performance.

1. Introduction

Graph representation learning has attracted widespread attention in the past few years. It plays a crucial role in various applications, such as social network analysis (Pal et al., 2020), drug discovery (Gaudelet et al., 2021), protein design (Ingraham et al., 2019), medical diagnosis (Li et al., 2020b) and so on.

Early research in graph representation learning primarily focuses on Graph Neural Networks (GNNs). A milestone example is GCN (Defferrard et al., 2016; Kipf & Welling, 2017). It performs convolution operations on the graph. Based on the framework of message-passing GNNs (Gilmer et al., 2017), GraphSage (Hamilton et al., 2017), GatedGCN (Bresson & Laurent, 2017) and GIN (Xu et al., 2019) adapt to complex graph data by employing different message-passing strategies. While message-passing GNNs have recently emerged as prominent methods for graph representation learning, there still exist some critical limitations, including the limited expressiveness (Xu et al., 2019; Morris et al., 2019), over-smoothing (Li et al., 2018; Chen et al., 2020; Oono & Suzuki, 2020), over-squashing (Alon & Yahav, 2021) and poor long-range dependencies.

Instead of aggregating local neighborhood, graph Transformer captures interaction information between any pair of nodes through a single self-attention layer. Some of the existing works focus on customizing specific attention mechanisms or positional encodings (Dwivedi & Bresson, 2020; Ying et al., 2021; Kreuzer et al., 2021; Hussain et al., 2022; Ma et al., 2023), while others combine message-passing GNNs to design hybrid architectures (Wu et al., 2021; Chen et al., 2022; Rampášek et al., 2022). These methods enable nodes to interact with all other nodes within a graph, facilitating the direct modeling of long-range relations. This may address typical issues such as over-smoothing in GNNs.

While the above-mentioned methods have achieved impressive results, their focus tends to be confined to internal information within the graph, specifically, different nodes or structures in the graph, neglecting potential correlations with other graphs. Indeed, in numerous practical scenarios, such as molecular graph data, strong correlations frequently exist between different graphs (molecules). This is exemplified by the three molecular graphs in Figure 1, all of which possess a benzene ring structure. Under such circumstances, integrating inter-graph correlations into the model can improve both the effectiveness and expressiveness of graph representation learning.

In this work, we address the critical question of how to incorporate external information into graph representation learning. Our principal contribution is to introduce a novel Graph External Attention (GEA) mechanism, which implicitly learns inter-graph correlations with the external key-value units. Moreover, we design Graph External Attention Enhanced Transformer (GEAET), combining inter-graph correlations with both local structure and global interaction information. This enables the acquisition of more comprehensive graph representations, in contrast to most existing

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

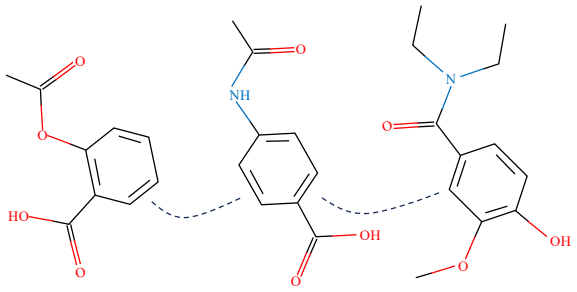


Figure 1. Three molecular graphs from the ZINC dataset are correlated to the benzene ring structure.

methods. Our contributions are listed as follows.

- We introduce GEA to implicitly learn correlations between all graphs. The complexity scales linearly with the number of nodes and edges.
- We propose GEAET, which uses GEA to learn external information and integrates local structure and global interaction information, resulting in more comprehensive and expressive graph representations.
- We demonstrate that GEAET achieves state-of-the-art performance. Furthermore, we highlight the significance of GEA, emphasizing its superior interpretability and reduced dependence on positional encoding compared to self-attention.

2. Related Work

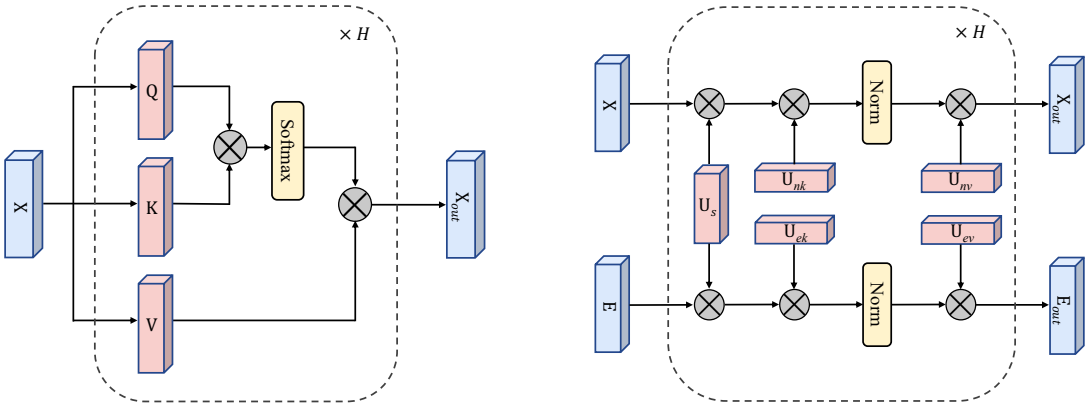
Message-Passing Graph Neural Networks. Early developments include GCN (Defferrard et al., 2016; Kipf & Welling, 2017), GraphSage (Hamilton et al., 2017), GIN (Xu et al., 2019), GAT (Veličković et al., 2018), GatedGCN (Bresson & Laurent, 2017) and others. These methods are based on a message-passing architecture (Gilmer et al., 2017) that generally faces the challenges of limited expressivity. Recent advancements include various works attempting to enhance GNNs to improve expressivity. Examples include some works that add features to distinguish nodes (Murphy et al., 2019; Sato et al., 2021; Qiu et al., 2018; Bouritsas et al., 2023; Dwivedi et al., 2022a). Others focus on altering the message-passing rule (Beaini et al., 2021) or modifying the underlying graph structure for message-passing (Morris et al., 2019; Bodnar et al., 2021) to further exploit the graph structure.

Despite achieving state-of-the-art performance, GNNs face over-smoothing and over-squashing due to their constrained receptive field. Over-smoothing happens when all node representations converge to a constant after deep layers, whereas over-squashing occurs when messages from distant nodes fail to propagate effectively. It is crucial to design new architectures beyond neighborhood aggregation to address these issues.

Graph Transformers. The Transformer with self-attention, a dominant approach in natural language processing (Vaswani et al., 2017; Devlin et al., 2018), has shown competitiveness in computer vision (Dosovitskiy et al., 2021). Given the remarkable achievements of Transformers and their capability to address crucial challenges of GNNs, graph Transformer has been proposed, attracting increasing attention. Existing works primarily focus on designing tailored attention mechanisms or positional and structural encodings, or combining message-passing GNNs, enabling models to capture complex structures.

A number of works embed topology information into graph nodes by designing tailored attention mechanisms or positional and structural encodings without message-passing GNNs. GT (Dwivedi & Bresson, 2020) introduces the pioneering example of a graph Transformer by integrating Laplacian positional encoding. In the following years, a series of works spring up. SAN (Kreuzer et al., 2021) incorporates both sparse and global attention mechanisms in each layer, utilizing Laplacian positional encodings for the nodes. Graphormer (Ying et al., 2021) attains state-of-the-art performance in graph-level prediction tasks with centrality encoding, spatial encoding and edge encoding. EGT (Hussain et al., 2022) introduces the edge channel into attention mechanisms and adopts an SVD-based positional encoding instead of Laplacian positional encoding. While self-attention is commonly constrained by quadratic complexity, our proposed GEA demonstrates a linear computational complexity with respect to both the number of nodes and edges.

Furthermore, some works introduce hybrid architectures incorporating message-passing GNNs. For instance, GraphTrans (Wu et al., 2021) utilizes a stack of GNN layers before establishing full connectivity attention within the graph. Focusing on kernel methods, SAT (Chen et al., 2022) introduces a structure-aware attention mechanism using GNNs to extract a subgraph representation rooted at each node before computing the attention. A recent breakthrough emerged with the introduction of GraphGPS (Rampášek et al., 2022), the first parallel framework that combines local message-passing and a global attention mechanism with various positional and structural encodings. While these methods have achieved competitive performance, they overlook external information in the graph. Therefore, to alleviate this issue, we design GEAET, which inherits the merits of the GEA network, message-passing GNN and Transformer, leverages inter-graph correlations, local structure and global interaction information.



(a) Transformer with self-attention (b) Graph external attention network

Figure 2. Transformer versus graph external attention network. For simplicity, we omit skip connections and FFNs.

3. Method

In the following, we denote a graph as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ represents the edges. The graph has $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. We denote the node features for a node $i \in \mathcal{V}$ as x_i and the features for an edge between nodes i and j as $e_{i,j}$. All node features and edge features are stored in matrices $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{E} \in \mathbb{R}^{m\times d}$, respectively.

3.1. Graph External Attention

We first revisit the Transformer, as illustrated in Figure 2a. Transformer consists of two blocks: a self-attention module and a feed-forward network (FFN). Specifically, self-attention regards the graph as a fully connected graph and computes the attention of each node to every other node. With the input node features $\mathbf{X}$, self-attention linearly projects the input into three matrices: a query matrix ($\mathbf{Q}$), a key matrix ($\mathbf{K}$) and a value matrix ($\mathbf{V}$), where $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$ and $\mathbf{V} = \mathbf{X}\mathbf{W}_V$. Then self-attention can be formulated as:

$$\begin{aligned} \mathbf{A}_{Self} &= \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_{out}}}) \in \mathbb{R}^{n\times n}, \\ \text{Self-Attn}(\mathbf{X}) &= \mathbf{A}_{Self}\mathbf{V} \in \mathbb{R}^{n\times d_{out}}, \end{aligned} \tag{1}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ are trainable parameters and d_{out} denotes the dimension of $\mathbf{Q}$. The output of the self-attention is followed by both a skip connection and a FFN.

Self-attention on a graph can be viewed as employing a linear combination of node features within a single graph to refine node features. However, it exclusively focuses on the correlations among nodes within a single graph, overlooking implicit connections between nodes in different graphs, which may potentially limit its capacity and adaptability.

Thus, inspired by (Guo et al., 2022), we introduce a novel method called GEA, as shown in Figure 2b. It calculates attention between the node features of the input graph and the external units, using:

$$\begin{aligned} \mathbf{A}_{GE} &= \text{norm}(\mathbf{X}\mathbf{U}^T) \in \mathbb{R}^{n\times S}, \\ \text{GE-Attn}(\mathbf{X}) &= \mathbf{A}_{GE}\mathbf{U} \in \mathbb{R}^{n\times d}, \end{aligned} \tag{2}$$

where $\mathbf{U} \in \mathbb{R}^{S\times d}$ is a learnable parameter independent of the input graph, it can be viewed as an external unit with S nodes, serving as shared memory for all input graphs. In self-attention, $\mathbf{A}_{Self}$ represents the similarities between the nodes of the input graph, while in GEA, $\mathbf{A}_{GE}$ denotes the similarities between the nodes of the input graph and the external unit. Considering the sensitivity of the attention matrix to the scale of input features, we apply a double-normalization technique (Guo et al., 2021) on $\mathbf{A}_{GE}$. The double-normalization process normalizes both columns and rows separately, it can be expressed as:

$$\begin{aligned} \tilde{\alpha}_{i,j} &= (\mathbf{X}\mathbf{U}^T)_{i,j}, \\ \hat{\alpha}_{i,j} &= \exp(\tilde{\alpha}_{i,j}) / \sum_{k=0}^{n} \exp(\tilde{\alpha}_{k,j}), \\ \alpha_{i,j} &= \hat{\alpha}_{i,j} / \sum_{k=0}^{S} \hat{\alpha}_{i,k}. \end{aligned} \tag{3}$$

In practical applications, for boosting the capability of the network, we utilize two distinct external units for the key and value. Furthermore, to leverage the edge information within the input graph, we employ additional external units for edge features and a shared unit to store the connections between edges and nodes:

$$\begin{aligned} \mathbf{X}_{out} &= \text{norm}(\mathbf{X}\mathbf{U}_s\mathbf{U}_{nk}^T)\mathbf{U}_{nv}, \\ \mathbf{E}_{out} &= \text{norm}(\mathbf{E}\mathbf{U}_s\mathbf{U}_{ek}^T)\mathbf{U}_{ev}, \end{aligned} \tag{4}$$

where $\mathbf{U}_s \in \mathbb{R}^{d\times d}$ is a shared unit to store the connections between edges and nodes; $\mathbf{U}_{nk}, \mathbf{U}_{nv} \in \mathbb{R}^{S\times d}$ are external key-value units for nodes, while $\mathbf{U}_{ek}, \mathbf{U}_{ev} \in \mathbb{R}^{S\times d}$ are external key-value units for edges.

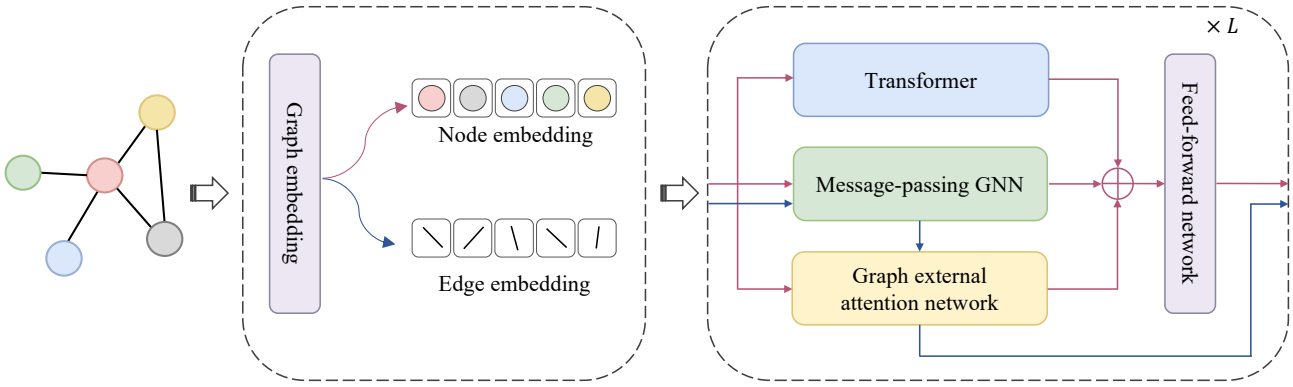


Figure 3. Overall architecture of GEAET. It consists of a graph embedding layer and L feature extraction layers. The graph embedding layer transforms graph data into node embeddings $\mathbf{X}$ and edge embeddings $\mathbf{E}$. It computes positional encodings, which are added to the node embeddings as inputs to the feature extraction layers. Each feature extraction layer consists of a graph external attention network, a message-passing GNN and a Transformer to extract inter-graph correlations, local structures and global interaction information. Finally, this information is integrated using a feed-forward network (FFN) and then employed on the output embeddings for various graph tasks.

Within the Transformer architecture, self-attention is computed across various input channels in multiple instances, a technique referred to as multi-head attention. Multi-head attention can capture diverse node relations, enhancing the ability of the attention mechanism. Similarly, take nodes as an example, the relations between nodes within the graph and external units are various. Therefore, we adopt an analogous approach, it can be written as:

$$\begin{aligned} h_i &= \text{GE-Attn}(\mathbf{X}_i, \mathbf{U}_{nk}, \mathbf{U}_{nv}), \\ \mathbf{X}_{out} &= \text{MultiHeadGEA}(\mathbf{X}, \mathbf{U}_{nk}, \mathbf{U}_{nv}) \\ &= \text{Concat}(h_1, ..., h_H)\mathbf{W}_o, \end{aligned} \tag{5}$$

where h_i represents the i-th head, H is the total number of heads, $\mathbf{W}_o$ is a linear transformation matrix, $\mathbf{U}_{nk}, \mathbf{U}_{nv} \in \mathbb{R}^{S\times d}$ serve as shared memory units for different heads. Finally, the output of the GEA is followed by a skip connection forming a Graph External Attention Network (GEANet).

3.2. Graph External Attention Enhanced Transformer

Figure 3 illustrates an overview of the proposed GEAET framework. GEAET consists of two components: graph embedding and feature extraction layers.

Graph Embedding. For each input graph, we initially perform a linear projection of the input node features $\alpha_i \in \mathbb{R}^{d_\alpha}$ and edge features $\beta_{i,j} \in \mathbb{R}^{d_\beta}$, resulting in d-dimensional hidden features:

$$\begin{aligned} \tilde{x_i^0} &= \mathbf{W}_x^0 \alpha_i + u^0 \in \mathbb{R}^d, \\ e_{ij}^0 &= \mathbf{W}_e^0 \beta_{i,j} + v^0 \in \mathbb{R}^d, \end{aligned} \tag{6}$$

where $\mathbf{W}_x^0 \in \mathbb{R}^{d\times d_\alpha}$, $\mathbf{W}_e^0 \in \mathbb{R}^{d\times d_\beta}$ and $u^0, v^0 \in \mathbb{R}^d$ are learnable parameters. Then, we use positional encoding to enhance the input node features, thereby improving the expressivity of the network:

$$x_i^0 = \mathbf{T}^0 p_i + \tilde{x_i^0}, \tag{7}$$

where $\mathbf{T}^0 \in \mathbb{R}^{d\times k}$ is a learnable matrix and $p_i \in \mathbb{R}^k$ is positional encoding. It is noteworthy that the advantages of different positional encodings are dependent on the dataset.

Feature Extraction Layer. At each layer, external feature information is captured by the GEANet and then aggregated with intra-graph information to update node features. The intra-graph information is obtained through a combination of message-passing GNN and Transformer. This process can be formulated as:

$$\begin{aligned} \mathbf{X}_M^{l+1}, \mathbf{E}_M^{l+1} &= \text{MPNN}^l(\mathbf{X}^l, \mathbf{E}^l, \mathbf{A}), \\ \mathbf{X}_T^{l+1} &= \text{TLayer}^l(\mathbf{X}^l), \\ \mathbf{X}_G^{l+1}, \mathbf{E}_G^{l+1} &= \text{GEANet}^l(\mathbf{X}^l, \mathbf{E}_M^{l+1}), \end{aligned} \tag{8}$$

where GEANet refers to the graph external attention network introduced in Section 3.1, TLayer represents the Transformer layer with self-attention, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, MPNN is an instance of a message-passing GNN to update node and edge representations as follows:

$$\begin{aligned} x_i^{l+1} &= f_{node}(x_i^l, \{x_j^l \mid j \in \mathcal{N}(i)\}, e_{i,j}^l), \\ e_{i,j}^{l+1} &= f_{edge}(x_i^l, x_j^l, e_{i,j}^l), \end{aligned} \tag{9}$$

where $x_i^{l+1}, x_i^l, e_{i,j}^{l+1}, e_{i,j}^l \in \mathbb{R}^d$, l is the layer index, i, j denotes the node index, $\mathcal{N}(i)$ is the neighborhood of the i-th node and the functions f_{node} and f_{edge} with learnable parameters define any arbitrary message-passing GNN architecture (Kipf & Welling, 2017; Bresson & Laurent, 2017;

Table 1. Comparison of GEAET to baselines on various tasks, including three tasks in Benchmarking GNNs (graph classification for CIFAR10 and MNIST, node classification for PATTERN), as well as three tasks in LRGB (node classification for PascalVOC-SP and COCO-SP, graph regression for Peptides-struct). Best results are colored: **first**, **second**, **third**.

Model	CIFAR10 Accuracy(%) ↑	MNIST Accuracy(%) ↑	PATTERN Accuracy(%) ↑	Peptides-Struct MAE ↓	PascalVOC-SP F1 score ↑	COCO-SP F1 score ↑
GCN (Kipf & Welling, 2017)	55.710 ± 0.381	90.705 ± 0.218	71.892 ± 0.334	0.3496 ± 0.0013	0.1268 ± 0.0060	0.0841 ± 0.0010
GINE (Xu et al., 2019)	–	–	–	0.3547 ± 0.0045	0.1265 ± 0.0076	0.1339 ± 0.0044
GIN (Xu et al., 2019)	55.255 ± 1.527	96.485 ± 0.252	85.387 ± 0.136	–	–	–
GAT (Veličković et al., 2018)	64.223 ± 0.455	95.535 ± 0.205	78.271 ± 0.186	–	–	–
GatedGCN (Bresson & Laurent, 2017)	67.312 ± 0.311	97.340 ± 0.143	85.568 ± 0.088	0.3357 ± 0.0006	0.2873 ± 0.0219	0.2641 ± 0.0045
PNA (Corso et al., 2020)	70.350 ± 0.630	97.940 ± 0.120	–	–	–	–
DGN (Beaini et al., 2021)	72.838 ± 0.417	–	86.680 ± 0.034	–	–	–
DRew (Gutteridge et al., 2023)	–	–	–	0.2536 ± 0.0015	0.3314 ± 0.0024	–
CRaW1 (Toenshoff et al., 2021)	69.013 ± 0.259	97.944 ± 0.050	–	–	–	–
GIN-AK+ (Zhao et al., 2022)	72.190 ± 0.130	–	86.850 ± 0.057	–	–	–
SAN (Kreuzer et al., 2021)	–	–	86.581 ± 0.037	0.2545 ± 0.0012	0.3230 ± 0.0039	0.2592 ± 0.0158
K-Subgraph SAT (Chen et al., 2022)	–	–	86.848 ± 0.037	–	–	–
EGT (Hussain et al., 2022)	68.702 ± 0.409	98.173 ± 0.087	86.821 ± 0.020	–	–	–
GraphGPS (Rampášek et al., 2022)	72.298 ± 0.356	98.051 ± 0.126	86.685 ± 0.059	0.2500 ± 0.0005	**0.3748 ± 0.0109**	**0.3412 ± 0.0044**
LGI-GT (Yin & Zhong, 2023)	–	–	**86.930 ± 0.040**	–	–	–
GPTrans-Nano (Gutteridge et al., 2023)	–	–	86.731 ± 0.085	–	–	–
Graph-ViT/MLPMixer (He et al., 2023)	73.960 ± 0.330	**98.460 ± 0.090**	–	**0.2449 ± 0.0016**	–	–
GRIT (Ma et al., 2023)	**76.468 ± 0.881**	98.108 ± 0.111	**87.196 ± 0.076**	**0.2460 ± 0.0012**	–	–
Exphormer (Shirzad et al., 2023)	**74.754 ± 0.194**	**98.414 ± 0.038**	86.734 ± 0.008	0.2481 ± 0.0007	**0.3966 ± 0.0027**	**0.3430 ± 0.0008**
GEAET (ours)	**76.634 ± 0.427**	**98.513 ± 0.086**	**86.993 ± 0.026**	**0.2445 ± 0.0013**	**0.4585 ± 0.0087**	**0.3895 ± 0.0050**

Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019; Hu et al., 2020).

Finally, we employ an FFN block to aggregate node information to obtain the node representations $\mathbf{X}^{l+1}$. Additionally, we employ $\mathbf{E}_G^{l+1}$ as the edge features for the $l+1$-th layer:

$$\begin{aligned} \mathbf{X}^{l+1} &= \mathrm{FFN}^l(\mathbf{X}_G^{l+1} + \mathbf{X}_T^{l+1} + \mathbf{X}_M^{l+1}), \\ \mathbf{E}^{l+1} &= \mathbf{E}_G^{l+1}, \end{aligned} \tag{10}$$

where $\mathbf{X}_T^{l+1}, \mathbf{X}_M^{l+1} \in \mathbb{R}^{n\times d}$ are the outputs of l-layer Transformer and message-passing GNN, $\mathbf{X}_G^{l+1}, \mathbf{E}_G^{l+1} \in \mathbb{R}^{n\times d}$ are the outputs of l-layer GEANet.

See Appendix C for the complexity and expressivity analysis of GEAET.

4. Experiments

In this section, we evaluate the empirical performance of GEANet and GEAET on a variety of graph datasets with graph prediction and node prediction tasks, including CIFAR10, MNIST, PATTERN, CLUSTER and ZINC from Benchmarking GNNs (Dwivedi et al., 2020), as well as PascalVOC-SP, COCO-SP, Petides-Struct, Petides-Func and PCQM-Contact from Long Range Graph Benchmark (LRGB; Dwivedi et al., 2022b), and the TreeNeighbourMatch dataset (Alon & Yahav, 2021). Detailed information is provided in the Appendix A.

We first compare our main architecture, GEAET, with the latest state-of-the-art models. In addition, we integrate GEANet with the message-passing GNNs and compare it with the corresponding network to demonstrate the role of GEANet. Furthermore, we conduct a series of comparative experiments with Transformer, including visualization experiments on attention in molecular graphs, experiments varying the number of attention heads and positional encoding experiments. Finally, we conduct ablation studies on each component of GEANet, including the external node unit, the external edge unit and the shared unit, to confirm the effectiveness of each component. More details on the experimental setup and hyperparameters are provided in the Appendix B, additional results are given in the Appendix C.

In summary, our experiments reveal that: (a) our GEAET architecture outperforms existing state-of-the-art methods on various datasets, (b) GEANet can be seamlessly integrated with some basic GNNs, significantly enhancing the performance of GNNs and (c) GEANet shows better interpretability than Transformer and is less dependent on positional encoding.

4.1. Comparison with SOTAs

We compare our methods with several recent SOTA graph Transformers, including Exphormer, GRIT, Graph-ViT/MLPMixer, and numerous popular graph representation learning models, such as well-known message-passing GNNs (GCN, GIN, GINE, GAT, GatedGCN, PNA), and graph Transformers (SAN, SAT, EGT, GraphGPS, LGI-GT, GPTrans-Nano). Additionally, we consider other recent methods with SOTA performance, such as DGN, DRew, CRaW1 and GIN-AK+.

As shown in Table 1, for the three tasks from Benchmarking GNNs (Dwivedi et al., 2020), we observe that our GEAET achieves SOTA results on CIFAR10 and MNIST and ranks second on the PATTERN dataset. For the three tasks on LRGB (Dwivedi et al., 2022b), GEAET achieves the best results. It is noteworthy that the GEAET achieves F1 scores

Table 2. Comparison of the classical message-passing GNN baseline to its variant augmented by GEANet (without positional encoding for fairness) across five LRGB tasks: (from left to right) node classification, node classification, graph regression, graph classification and link prediction.

Model	PascalVOC-SP F1 score ↑	COCO-SP F1 score ↑	Peptides-Struct MAE ↓	Peptides-Func AP ↑	PCQM-Contact MRR ↑
GCN	0.1268 ± 0.0060	0.0841 ± 0.0010	0.3496 ± 0.0013	0.5930 ± 0.0023	0.3234 ± 0.0006
+GEANet	**0.2250 ± 0.0103**	**0.2096 ± 0.0041**	**0.2512 ± 0.0003**	**0.6722 ± 0.0065**	**0.3244 ± 0.0007**
GINE	0.1265 ± 0.0076	0.1339 ± 0.0044	0.3547 ± 0.0045	0.5498 ± 0.0079	0.3180 ± 0.0027
+GEANet	**0.2742 ± 0.0032**	**0.2410 ± 0.0028**	**0.2544 ± 0.0012**	**0.6509 ± 0.0021**	**0.3276 ± 0.0012**
GatedGCN	0.2873 ± 0.0219	0.2641 ± 0.0045	0.3420 ± 0.0013	0.5864 ± 0.0077	0.3242 ± 0.0008
+GEANet	**0.3933 ± 0.0027**	**0.3219 ± 0.0052**	**0.2547 ± 0.0009**	**0.6485 ± 0.0035**	**0.3321 ± 0.0008**

Table 3. Comparison of the classical message-passing GNN baseline to its variant augmented by GEANet (without positional encoding for fairness) across five Benchmarking GNNs tasks: (from left to right) node classification, node classification, graph classification, graph classification and graph regression.

Model	PATTERN Accuracy(%) ↑	CLUSTER Accuracy(%) ↑	MNIST Accuracy(%) ↑	CIFAR10 Accuracy(%) ↑	ZINC MAE ↓
GCN	71.892 ± 0.334	68.498 ± 0.976	90.705 ± 0.218	55.710 ± 0.381	0.367 ± 0.011
+GEANet	**85.323 ± 0.128**	**74.015 ± 0.124**	**96.465 ± 0.054**	**61.925 ± 0.271**	**0.240 ± 0.008**
GIN	85.387 ± 0.136	64.716 ± 1.553	96.485 ± 0.252	55.255 ± 1.527	0.526 ± 0.051
+GEANet	**85.527 ± 0.015**	**66.370 ± 2.145**	**96.845 ± 0.097**	**62.320 ± 0.221**	**0.193 ± 0.001**
GatedGCN	85.568 ± 0.088	73.840 ± 0.326	97.340 ± 0.143	67.312 ± 0.311	0.282 ± 0.015
+GEANet	**85.607 ± 0.038**	**77.013 ± 0.224**	**98.315 ± 0.097**	**73.857 ± 0.306**	**0.218 ± 0.011**

of 0.4585 on PascalVOC-SP and 0.3895 on COCO-SP, surpassing other models by a significant gap.

4.2. Comparison with GNNs

To clearly demonstrate the performance improvement of GEANet on graph representation learning models, we integrate GEANet with some commonly used message-passing GNNs, providing the models with the ability to learn graph external information. In our comparison, we evaluate our approach against the corresponding GNNs, with GNN results sourced from Dwivedi et al. (2020) or Dwivedi et al. (2022b). To maintain a fair comparison, our trained models strictly adhere to the parameter constraints without incorporating any positional encoding, consistent with Dwivedi et al. (2020) and Dwivedi et al. (2022b). As depicted in Table 2 and Table 3, GEANet significantly improves the performance of all base message-passing GNNs, including GCN (Kipf & Welling, 2017), GatedGCN (Bresson & Laurent, 2017), GIN (Xu et al., 2019) and GINE (Hu et al., 2020), on various datasets simply by combining the output of GEANet with the output of the GNN. This is achieved without any additional modifications, validating that GEANet can effectively alleviate issues in message-passing GNNs.

4.3. Comparison with Self-Attention

Attention Interpretation. To better explain the attention mechanism, we conduct visualization experiments on the ZINC dataset. As shown in Figure 4, both molecular graphs have a benzene ring structure consisting of six light blue carbon atoms. We observe that GEANet captures this correlation and assigns low attention scores with light orange to

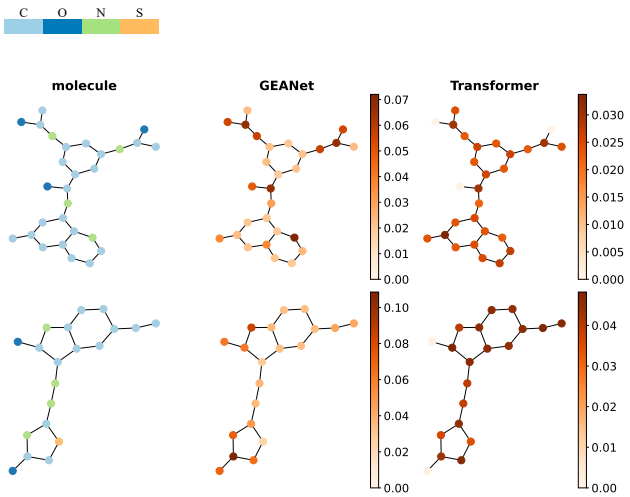


Figure 4. Attention visualization of GEANet and Transformer on ZINC molecular graphs. The left column shows two original molecular graphs, while the middle and right columns show the visualization results of attention scores with GEANet and Transformer, respectively.

the benzene rings of both graphs. This allows the model to focus more on other unique atoms or discriminative motifs. The attention distribution of GEANet is similar to the structural distribution of the original molecular graphs, which helps to represent features better. In contrast, Transformer does not utilize inter-graph correlations. It assigns orange attention scores to the benzene ring in the first molecular graph and deep orange attention scores to that in the second molecular graph. This leads to significant differences

between the attention distribution and the structural distribution, which limits the expressive power. More results are provided in the Appendix E.

Impact of Attention Heads. We investigate the impact on the number of attention heads with two attention mechanisms. Figure 5 shows the MAE values with either GCN + Transformer or GCN + GEANet on the Peptides-Struct dataset. For a fair comparison, both models use Laplacian positional encoding, with the same number of layers and approximately total number of parameters (about 500k). Heads = 0 corresponds to a pure GCN network without attention. The introduction of attention mechanism significantly improves performance. GEANet achieves the best performance with 8 heads. In contrast, Transformer performs best with one head, suggesting that multiple self-attention heads do not enhance performance. Notably, GEANet consistently outperforms self-attention across various numbers of heads.

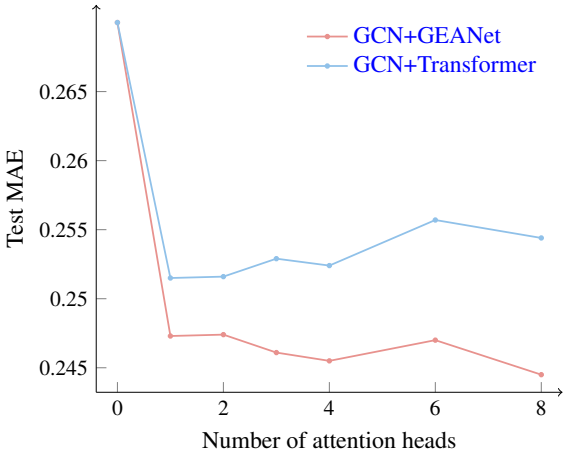


Figure 5. Test MAE with different number of attention heads.

Impact of Positional Encoding. We conduct an ablation study on the Peptides-Struct dataset to assess the impact of positional encoding on GEANet and Transformer. Similar to the experiments with attention heads, we utilize a parallel architecture consisting of an GCN block and an attention block. Figure 6 shows the MAE values obtained with and without positional encoding, including Random Walk Positional Encoding (RWPE; Li et al., 2020a; Dwivedi et al., 2022a) and Laplacian Positional Encoding (LapPE; Dwivedi & Bresson, 2020; Kreuzer et al., 2021). The Transformer with self-attention performs poorly without positional encoding. The utilization of LapPE and RWPE improves performance to some extent. In contrast, GEANet achieves an good performance without positional encoding. For GEANet, we observe that LapPE can enhance performance, while RWPE decreases performance. On the whole, GEANet is less dependent on positional encoding compared to Transformer.

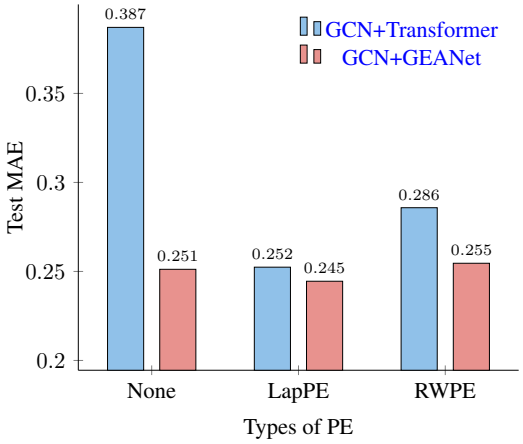


Figure 6. Test MAE with different Positional Encodings (PEs).

4.4. GEAET Mitigates Over-Squashing

The TreeNeighbourMatch dataset (Alon & Yahav, 2021), serves as a synthetic dataset to highlight the issue of over-squashing intuitively. On this dataset, each example represents a binary tree of depth r. Figure 7 shows the performance comparison of the standard message-passing GNNs (i.e., GCN, GINE, GAT, GIN and GatedGCN) and our proposed GEAET on the TreeNeighbourMatch dataset. The results indicate that our GEAET generalizes well on the dataset up to $r = 7$, effectively alleviating the issue of over-squashing. In contrast, the five GNN methods fail to generalize effectively from $r = 4$. This is consistent with the perspective of Alon & Yahav (2021) that GNN suffers from over-squashing due to squashing information of the graph into a fixed-length embedding. In contrast, our method addresses this problem by utilizing graph external attention mechanisms to transmit long-range information without squashing.

4.5. Ablation Experiments

To assess the practicality of our model design choices, we conduct multiple ablation experiments on MNIST and PATTERN. The results are shown in Table 4. We notice that the removal of either external node units, external edge units, or external shared units all leads to poorer performance, which demonstrates the soundness of our architectural decisions.

5. Conclusion

Observing that existing graph representation learning methods are confined to internal information, we argue for the importance of inter-graph correlations. Drawing inspira-

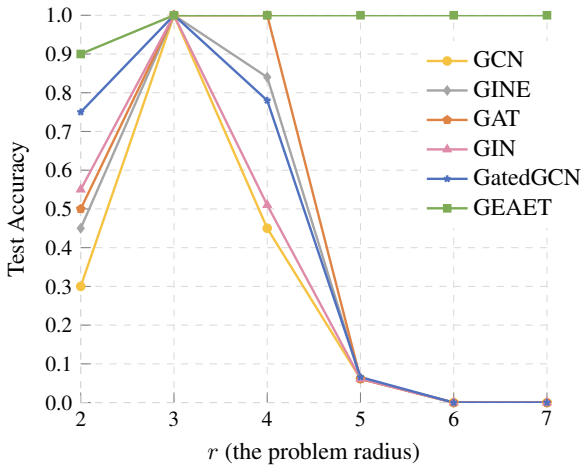


Figure 7. Test accuracy across problem radius (tree depth) on the TreeNeighbourMatch dataset.

Table 4. Ablation study on MNIST and PATTERN datasets for GEA components. The metric is the Accuracy(%).

Node	Edge	Share	MNIST	PATTERN
–	–	–	98.274 ± 0.011	86.882 ± 0.028
✓	–	–	98.474 ± 0.013	86.956 ± 0.038
✓	✓	–	98.488 ± 0.082	86.959 ± 0.024
✓	✓	✓	**98.513 ± 0.086**	**86.993 ± 0.026**

tion from the idea that graphs with similar structures ought to have analogous representations, we propose GEA, a novel lightweight yet effective attention mechanism to exploit inter-graph correlations. On this basis, we introduce GEAET to exploit local structure and global interaction information. GEAET achieves state-of-the-art performance on various graph datasets, highlighting the significance of inter-graph correlations. Nevertheless, GEAET is not the final chapter of our work; future efforts will focus on reducing the high memory cost and time complexity, as well as addressing the lack of upper bounds on expressive power.

References

Abbe, E. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Lió, P. Directional graph networks. In *International Conference on Machine Learning*, pp. 748–758, 2021.

Bodnar, C., Frasca, F., Otter, N., Wang, Y., Liò, P., Montufar, G. F., and Bronstein, M. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, 26, 2013.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2023.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., and Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3438–3445, 2020.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pp. 3469–3489, 2022.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 33: 13260–13271, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, 29, 2016.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022a.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022b.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88: 303–338, 2010.

Gaudelet, T., Day, B., Jamasb, A. R., Soman, J., Regep, C., Liu, G., Hayter, J. B., Vickers, R., Roberts, C., Tang, J., et al. Utilizing graph machine learning within drug discovery and development. *Briefings in Bioinformatics*, 22(6), 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., and Hu, S.-M. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

Guo, M.-H., Liu, Z.-N., Mu, T.-J., and Hu, S.-M. Beyond self-attention: External attention using two linear layers for visual tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5436–5447, 2022.

Gutteridge, B., Dong, X., Bronstein, M. M., and Di Giovanni, F. Drew: Dynamically rewired message passing with delay. In *International Conference on Machine Learning*, pp. 12252–12267, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

He, X., Hooi, B., Laurent, T., Perold, A., Lecun, Y., and Bresson, X. A generalization of ViT/MLP-mixer to graphs. In *International Conference on Machine Learning*, pp. 12724–12745, 2023.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations*, 2020.

Hussain, M. S., Zaki, M. J., and Subramanian, D. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 655–665, 2022.

Ingraham, J., Garg, V., Barzilay, R., and Jaakkola, T. Generative models for graph-based protein design. *Advances in Neural Information Processing Systems*, 32, 2019.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: a free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, pp. 21618–21629, 2021.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020a.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Li, Y., Qian, B., Zhang, X., and Liu, H. Graph neural network-based diagnosis prediction. *Big Data*, 8(5):379–390, 2020b.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Ma, L., Lin, C., Lim, D., Romero-Soriano, A., Dokania, P. K., Coates, M., Torr, P., and Lim, S.-N. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pp. 23321–23337, 2023.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 4602–4609, 2019.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673, 2019.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In

International Conference on Learning Representations, 2020.

Pal, A., Eksombatchai, C., Zhou, Y., Zhao, B., Rosenberg, C., and Leskovec, J. Pinnersage: Multi-modal user embedding framework for recommendations at pinterest. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2311–2320, 2020.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pp. 459–467, 2018.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining*, pp. 333–341, 2021.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pp. 31613–31632, 2023.

Singh, S., Chaudhary, K., Dhanda, S. K., Bhalla, S., Usmani, S. S., Gautam, A., Tuknait, A., Agrawal, P., Mathur, D., and Raghava, G. P. Satpdb: a database of structurally annotated therapeutic peptides. *Nucleic Acids Research*, 44, 2016.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph learning with 1d convolutions on random walks. *arXiv:2102.08786*, 2021.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J. E., and Stoica, I. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Yin, S. and Zhong, G. Lgi-gt: graph transformers with local and global operators interleaving. In *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 4504–4512, 2023.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022.

A. Dataset Descriptions

We evaluate our method on diverse datasets, including five graph benchmark datasets from Benchmarking GNNs, five long-range dependency graph datasets from LRGB and the TreeNeighbourMatch dataset. Below, we provide descriptions of the datasets and present summary statistics in Table 5.

MNIST and CIFAR10. MNIST and CIFAR10 (Dwivedi et al., 2020) represent the graphical counterparts of their respective image classification datasets of the same name. A graph is formed by creating an 8-nearest neighbor graph of the SLIC superpixels of the images. These datasets pose 10-class graph classification challenges. For MNIST, the resulting graphs have sizes ranging from 40 to 75 nodes, while for CIFAR10, the graphs vary between 85 and 150 nodes. The classification tasks involve the standard dataset splits of 55K/5K/10K for MNIST and 45K/5K/10K for CIFAR10, corresponding to train/validation/test graphs. These datasets serve as sanity checks, with an expectation that most GNNs would achieve close to 100% accuracy for MNIST and satisfactory performance for CIFAR10.

PATTERN and CLUSTER. PATTERN and CLUSTER (Dwivedi et al., 2020) are synthetic datasets designed for node classification tasks, both derived from the Stochastic Block Model (SBM; Abbe, 2018) used to model communities. In PATTERN, the objective is to determine whether a node is part of one of the 100 predefined subgraph patterns, while CLUSTER focuses on classifying nodes into six distinct clusters with identical distributions. The unique feature of PATTERN involves recognizing nodes belonging to randomly generated sub-graph patterns, while CLUSTER entails inferring the cluster ID for each node in graphs composed of 6 SBM-generated clusters. We use the splits as is used in (Dwivedi et al., 2020).

ZINC. ZINC (Dwivedi et al., 2020) is a graph regression dataset derived from a subset of molecular graphs (12K out of 250K) sourced from a freely available database of commercially accessible compounds (Irwin et al., 2012). The molecular graphs in ZINC range from 9 to 37 nodes, where each node represents a heavy atom (with 28 possible atom types) and each edge signifies a bond (with 3 possible types). The primary task is to regress a molecular property known as constrained solubility. The dataset includes a predefined train/validation/test split of 10K/1K/1K instances.

PascalVOC-SP and COCO-SP. PascalVOC-SP and COCO-SP (Dwivedi et al., 2022b) are graph-based versions of image datasets with larger images and involve the task of node classification, specifically the semantic segmentation of superpixels. These datasets respectively derived from the Pascal VOC 2011 image dataset (Everingham et al., 2010) and the MS COCO image dataset through SLIC superpixelization, present a more intricate node classification challenge compared to CIFAR10 and MNIST. Each superpixel node is associated with a specific object class, making them node classification datasets with a focus on regions of images belonging to particular classes.

Peptides-Func and Peptides-Struct. Peptides-Func and Peptides-Struct (Dwivedi et al., 2022b) are derived from 15,535 peptides with a total of 2.3 million nodes sourced from SAT-Pdb (Singh et al., 2016). The graphs exhibit large sizes, averaging 150.94 nodes per graph and a mean graph diameter of 56.99. Specifically suited for benchmarking graph Transformers or expressive GNNs capable of capturing long-range interactions. Peptides-func involves multi-label graph classification into 10 nonexclusive peptide functional classes, while Peptides-struct focuses on graph-level regression predicting 11 3D structural properties of the peptides.

PCQM-Contact. PCQM-Contact (Dwivedi et al., 2022b) is derived from PCQM4Mv2 and corresponding 3D molecular structures, where the task is a binary link prediction task. This dataset contains 529,434 graphs with a total of 15 million nodes, where each graph represents a molecular graph with explicit hydrogens. All graphs in PCQM-Contact are extracted from the PCQM4M training set, specifically those with available 3D structure and filtered to retain only those with at least one contact.

TreeNeighbourMatch. TreeNeighbourMatch is a synthetic dataset introduced by Alon & Yahav (2021) to illustrate the challenge of over-squashing in GNNs. It features binary trees of controlled depth that simulate an exponentially-growing receptive field with a problem radius r. The task is to predict a label for the target node, situated in one of the leaf nodes, necessitating information propagation from all leaves to the target node. This setup exposes the issue of over-squashing at the target node due to the need for comprehensive long-range signal incorporation before label prediction.

Table 5. Summary statistics of datasets used in this study.

Dataset	Graphs	Avg. nodes	Avg. edges	Prediction Level	Task	Metric
MNIST	70,000	70.6	564.5	graph	10-class classif.	Accuracy
CIFAR10	60,000	117.6	941.1	graph	10-class classif.	Accuracy
PATTERN	14,000	118.9	3,039.3	inductive node	binary classif.	Accuracy
CLUSTER	12,000	117.2	2,150.9	inductive node	6-class classif.	Accuracy
ZINC	12,000	23.2	24.9	graph	regression	MAE
PascalVOC-SP	11,355	479.4	2,710.5	inductive node	21-class classif.	F1
COCO-SP	123,286	476.9	2,693.7	inductive node	81-class classif.	F1
PCQM-Contact	529,434	30.1	61.0	inductive link	link ranking	MRR
Peptides-Func	15,535	150.9	307.3	graph	10-class classif.	Avg. Precision
Peptides-Struct	15,535	150.9	307.3	graph	regression	MAE
TreeNeighbourMatch(r=2)	96	7	6	inductive node	4-class classif.	Accuracy
TreeNeighbourMatch(r=3)	32,000	15	14	inductive node	8-class classif.	Accuracy
TreeNeighbourMatch(r=4)	64,000	31	30	inductive node	16-class classif.	Accuracy
TreeNeighbourMatch(r=5)	128,000	63	62	inductive node	32-class classif.	Accuracy
TreeNeighbourMatch(r=6)	256,000	127	126	inductive node	64-class classif.	Accuracy
TreeNeighbourMatch(r=7)	512,000	255	254	inductive node	128-class classif.	Accuracy

B. Hyperparameter Choices and Reproducibility

Hyperparameter Choice. In our hyperparameter search, we attempt to adjust the number of heads in GEANet, as well as hyperparameters related to positional encoding, message-passing GNN type and Transformer. Considering the large number of hyperparameters and datasets, we do not conduct an exhaustive search or grid search. For a fair comparison, we follow commonly used parameter budgets: for benchmarking datasets from Benchmarking GNNs (Dwivedi et al., 2020), a maximum of 500k parameters for PATTERN and approximately 100k parameters for MNIST and CIFAR10; for datasets from LRGB (Dwivedi et al., 2022b), we adhere to a parameter budget of 500k. See Table 6 for detailed information.

Optimization. We use the AdamW (Loshchilov & Hutter, 2019) optimizer in all our experiments, with the default settings of $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, and use a cosine scheduler (Loshchilov & Hutter, 2017). The choice of loss function, length of the warm-up period, base learning rate and total number of epochs are adjusted based on the dataset.

Table 6. Hyperparameters used for GEAET across datasets: CIFAR10, MNIST, PATTERN, Peptides-Struct, PascalVOC-SP, COCO-SP.

Hyperparameter	CIFAR10	MNIST	PATTERN	Peptides-Struct	PascalVOC-SP	COCO-SP
Layers	5	5	7	6	8	8
Hidden Dim d	40	40	64	224	68	68
MPNN	GatedGCN	GatedGCN	GatedGCN	GCN	GatedGCN	GatedGCN
Self Attention	Transformer	Transformer	Transformer	None	Transformer	Transformer
External Network	GEANet	GEANet	GEANet	GEANet	GEANet	GEANet
Self Heads	4	4	4	None	4	4
External Heads	4	4	4	8	4	4
Unit Size S	10	10	16	28	17	17
PE	ESLapPE-8	ESLapPE-8	RWPE-16	LapPE-10	None	None
PE Dim	8	8	7	16	None	None
Batch Size	16	16	32	200	50	50
Learning Rate	0.001	0.001	0.0005	0.001	0.001	0.001
Num Epochs	150	150	100	250	200	200
Warmup Epochs	5	5	5	5	10	10
Weight Decay	1e-5	1e-5	1e-5	0	0	0
Num Parameters	113,235	113,155	429,052	463,211	506,213	505,661

C. Additional Results

We provide additional results here, including detailed results from the attention heads and position encoding experiments, the results of GEAET on the link prediction task of the PCQM-Contact dataset and a substantial number of additional ablation studies.

Impact of Attention Heads. We conduct experiments with different numbers of attention heads on the Peptides-Struct and Peptides-Func datasets. We adopt a framework where the base GNN and the attention block (Transformer or GEANet) operate in parallel, with the output of the model at each layer being the sum of the outputs from the GNN and the attention block. To ensure fairness, we use the same number of layers, apply Laplacian positional encoding (LapPE), use GCN as the base GNN and keep the total number of parameters to about 500k. The detailed results are shown in Table 7, where all results are averaged over four different random seeds. We observe that having multiple attention heads does not significantly improve Transformer for both datasets. However, there is a notable improvement in GEANet, with the lowest MAE achieved with 8 heads on Peptides-Struct and the best AP achieved with 8 heads on Peptides-Func.

Table 7. The results on Transformer and GEANet with different number of attention heads.

Model	#Layers	Positional Encoding	#Heads	#Parameters	Peptides-Struct MAE ↓	Peptides-Func AP ↑
GCN + Transformer	6	LapPE	2	490,571	0.2516 ± 0.0031	0.6644 ± 0.0052
GCN + Transformer	6	LapPE	3	490,571	0.2529 ± 0.0012	0.6688 ± 0.0072
GCN + Transformer	6	LapPE	4	490,571	0.2524 ± 0.0017	0.6634 ± 0.0033
GCN + Transformer	6	LapPE	6	490,571	0.2557 ± 0.0032	0.6630 ± 0.0085
GCN + Transformer	6	LapPE	8	490,571	0.2544 ± 0.0037	0.6593 ± 0.0060
GCN + GEANet	6	LapPE	2	626,219	0.2474 ± 0.0006	0.6828 ± 0.0059
GCN + GEANet	6	LapPE	3	539,123	0.2461 ± 0.0006	0.6890 ± 0.0060
GCN + GEANet	6	LapPE	4	508,571	0.2455 ± 0.0009	0.6892 ± 0.0042
GCN + GEANet	6	LapPE	6	486,683	0.2470 ± 0.0025	0.6880 ± 0.0025
GCN + GEANet	6	LapPE	8	463,211	**0.2445 ± 0.0013**	**0.6912 ± 0.0012**

Impact of Positional Encoding. Similar to the experiments on attention heads, we study the impact of different positional encodings on attention. Table 8 shows the results averaged over four different random seeds. We find that GEANet has a lower dependency on positional encoding compared to Transformer using self-attention.

Table 8. The results on Transformer and GEANet with different positional encodings.

Model	#Layers	Positional Encoding	#Heads	#Parameters	Peptides-Struct MAE ↓	Peptides-Func AP ↑
GCN + Transformer	6	None	4	492,731	0.3871 ± 0.0094	0.6404 ± 0.0095
GCN + Transformer	6	RWPE	4	490,143	0.2858 ± 0.0044	0.6564 ± 0.0122
GCN + Transformer	6	LapPE	4	490,571	0.2524 ± 0.0017	0.6589 ± 0.0069
GCN + GEANet	6	None	4	510,731	0.2512 ± 0.0003	0.6722 ± 0.0065
GCN + GEANet	6	RWPE	4	508,143	0.2546 ± 0.0018	0.6794 ± 0.0089
GCN + GEANet	6	LapPE	4	508,571	**0.2445 ± 0.0013**	**0.6892 ± 0.0042**

GEAET in Link Prediction Task. In the link prediction task, we evaluate common ranking metrics from the knowledge graph link prediction literature (Bordes et al., 2013) as shown in Table 9: Hits@1, Hits@3, Hits@10 and Mean Reciprocal Rank (MRR), where Hits@k indicates whether the actual answer is among the top-k predictions provided by the model.

Table 9. Performance of GEAET on the link prediction task of the PCQM-Contact dataset. We select baseline models that also report the Hits@1, Hits@3, Hits@10, and MRR metrics. Our results are averaged over 4 runs with 4 different seeds, while the results of the baseline models are either from (Dwivedi et al., 2022b) or the original papers.

Model	Test Hits@1 ↑	Test Hits@3 ↑	Test Hits@10 ↑	Test MRR ↑
SAN	0.1312 ± 0.0016	0.4030 ± 0.0008	0.8550 ± 0.0024	0.3341 ± 0.0006
GatedGCN	0.1288 ± 0.0013	0.3808 ± 0.0006	0.8517 ± 0.0005	0.3242 ± 0.0008
Transformer	0.1221 ± 0.0011	0.3679 ± 0.0033	0.8517 ± 0.0039	0.3174 ± 0.0020
GCN	0.1321 ± 0.0007	0.3791 ± 0.0004	0.8256 ± 0.0006	0.3234 ± 0.0006
GINE	0.1337 ± 0.0013	0.3642 ± 0.0043	0.8147 ± 0.0062	0.3180 ± 0.0027
Graph Diffuser	0.1369 ± 0.0012	0.4053 ± 0.0011	0.8592 ± 0.0007	0.3388 ± 0.0011
GEAET (ours)	**0.1566 ± 0.0014**	**0.4227 ± 0.0022**	**0.8626 ± 0.0032**	**0.3518 ± 0.0011**

Improving GNN with GEANet. To demonstrate the importance of GEANet, we conduct additional experiments on the Peptides-Struct, Peptides-Func, and PascalVOC-SP datasets. As shown in Table 10, compare with positional encoding, GEANet significantly improves the performance of all base message-passing GNNs.

Table 10. Improving GNN performance with GEANet. We run the experiments with 4 different seeds and average the results.

Model	Positional Encoding	PascalVOC-SP F1 score ↑	Peptides-Struct MAE ↓	Peptides-Func AP ↑
GCN	None	0.1268 ± 0.0060	0.3496 ± 0.0013	0.5930 ± 0.0023
GCN + GEANet	None	0.2250 ± 0.0103	0.2512 ± 0.0003	0.6722 ± 0.0065
GCN + GEANet	LapPE	**0.2353 ± 0.0070**	**0.2445 ± 0.0013**	**0.6892 ± 0.0042**
GCN + GEANet	RWPE	0.2325 ± 0.0165	0.2546 ± 0.0018	0.6794 ± 0.0089
GINE	None	0.1265 ± 0.0076	0.3547 ± 0.0045	0.5498 ± 0.0079
GINE + GEANet	None	0.2742 ± 0.0032	0.2544 ± 0.0012	0.6509 ± 0.0021
GINE + GEANet	LapPE	0.2746 ± 0.0071	**0.2480 ± 0.0023**	**0.6654 ± 0.0055**
GINE + GEANet	RWPE	**0.2762 ± 0.0022**	0.2546 ± 0.0011	0.6618 ± 0.0059
GatedGCN	None	0.2873 ± 0.0219	0.3420 ± 0.0013	0.5864 ± 0.0077
GatedGCN + GEANet	None	0.3933 ± 0.0027	0.2547 ± 0.0009	0.6485 ± 0.0035
GatedGCN + GEANet	LapPE	**0.3944 ± 0.0044**	**0.2468 ± 0.0014**	0.6715 ± 0.0034
GatedGCN + GEANet	RWPE	0.3899 ± 0.0017	0.2577 ± 0.0006	**0.6734 ± 0.0028**

D. Complexity and Expressivity Analysis

Complexity Analysis. We first analyze the complexity of GEANet. As the model dimensions d and the size of external units S are hyper-parameters, GEANet scales linearly with the number of nodes and edges, resulting in a complexity of $O(|\mathcal{V}| + |\mathcal{E}|)$. For GEAET, the complexity is primarily determined by GEANet, Transformer and message-passing GNN. The GEANet, as described above, has linear complexity. The message-passing GNN has a complexity of $O(|\mathcal{E}|)$. In typical cases, the Transformer uses the self-attention mechanism with a complexity of $O(|\mathcal{V}|^2)$, resulting in complexity of $O(|\mathcal{V}|^2)$. In practice, we observe that on certain datasets such as Peptides-Struct and Peptides-Func, not using Transformer yields better results, achieving linear complexity in such cases. Additionally, we can use linear Transformers to reduce the complexity of GEAET to linearity.

Expressivity Analysis. Kreuzer et al. (2021) demonstrate the universality of Graph Transformer. Based on their findings, since our model can add the complete set of Laplacian eigenvectors on the Graph Embedding layer, as long as there are enough parameters, GEAET will become a universal function approximator on graphs. It can provide an approximate solution to the isomorphic problem, making it more powerful than any Weisfeiler-Leman (WL) isomorphism test.

E. Model Interpretation

In addition to the examples presented in the main paper, we provide additional visualization results in Figure 8. GEANet and Transformer use the same positional encoding, and other hyperparameter settings are generally consistent. The first column shows the original molecules from ZINC, the middle and right columns show the visualization results of GEANet and Transformer, respectively. We observe that GEANet focuses more on the important nodes or connected nodes of specific structures, which improves the ability to distinguish different graphs or motifs. This indicates that GEANet excels in handling structural information and concentrates on discriminative nodes.

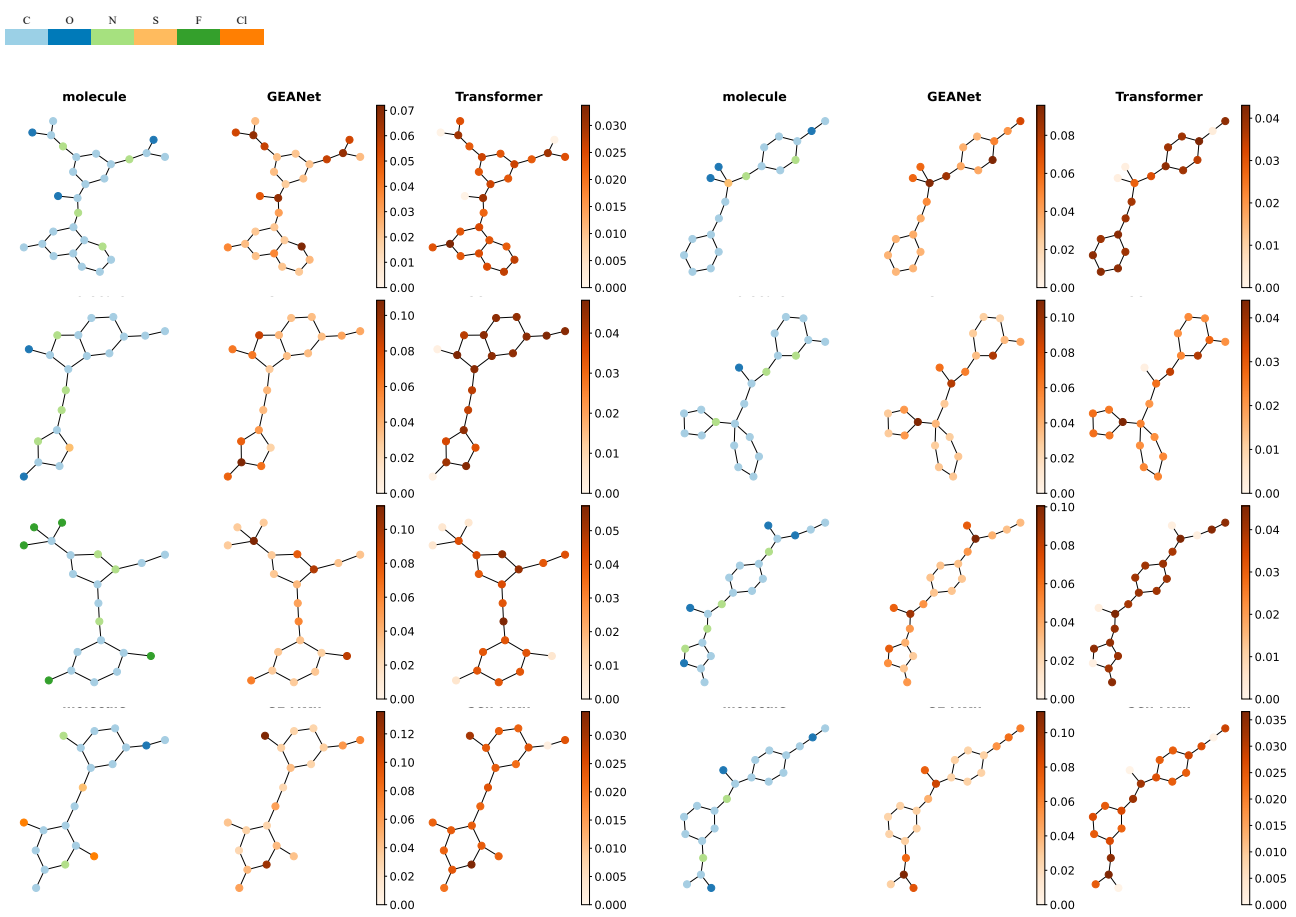


Figure 8. Attention visualization of GEANet and Transformer on ZINC molecular graphs. The center column shows the attention weights of GEANet and the right column shows the attention weights learned by the classic Transformer.