scientific reports

Check for updates

OPEN

A graph transformer with optimized attention scores for node classification

Yu Zhang[1,3], Xin Li[1,3], Yaoqun Xu[1]✉, Xitong Xu[2] & Zhe Wang[1]

The message-passing paradigm on graphs has significant advantages in modeling local structures, but still faces challenges in capturing global information and complex relationships. Although the Transformer architecture has become a mainstream approach for nonlocal modeling in many domains, Transformer-based architectures fail to demonstrate competitiveness in popular node-level prediction tasks when compared to mainstream graph neural network (GNN) variants. This can be attributed to the fact that existing research has largely focused on more efficient strategies to approximate the Vanilla Transformer, thereby overlooking its potential in node embedding representation learning. This paper introduces a novel graph Transformer model with optimized attention scores, named OGFormer, to address this gap. OGFormer employs a simplified single-head self-attention mechanism, incorporating several critical structural innovations in its self-attention layers to more effectively capture global dependencies within graphs. Specifically, we propose an end-to-end attention score optimization loss function, designed to suppress noisy connections and enhance the connection weights between similar nodes. Additionally, we introduce an effective structural encoding strategy integrated into the attention score computation, enabling the model to focus more on key local dependencies in the graph. Experimental results demonstrate that OGFormer achieves strong competitive performance in node classification tasks across multiple benchmark datasets, particularly in Homophilous and Heterophilous tests, surpassing current mainstream methods. Our implementation is available at https://github.com/LittleBlackBearLiXin/OGFormer.

Keywords Graph neural network, Graph transformers, Graph attention, Node classification

With the wide application of graph data, the graph neural network (GNN), as an important model for deep learning, has achieved remarkable success in modeling non-Euclidean data. The reason for this is its unique message-passing mechanism, which can effectively learn the local patterns of the nodes and edges in graph structures, thus providing effective solutions for tasks such as graph node classification[1–3], graph classification[4–6], link prediction[7], and knowledge graph embedding[8]. However, despite the excellent performance of graph neural networks in local structure modeling, there are still extremely difficult challenges in dealing with global information and complex relationships of graphs.

In the single-layer structure of a traditional message-passing neural network (MPNN)[5], each node passes its representation of a'message' to its neighboring nodes along fixed neighboring edges and aggregates all'messages' in the neighborhood through a substitution-invariant aggregation function to update the representation of the node. However, it is difficult to distinguish the directionality of the edges and the weight attributes conveyed by the class labels when aggregating neighbor messages. Therefore, although local graph attention networks[9–11] compute the attention weights between neighboring nodes based on node features, it is still difficult to effectively capture information about distant and potentially similar non-neighboring nodes. In addition, it is not easy to characterize the non-local attributes of nodes using deep network layers. Since neighborhood aggregation is essentially a Laplace smoothing process[12], simply put, if the graph is irreducible and nonperiodic, the representations of different nodes in a deep MPNN become very similar, and each node's representation actually carries the information of the entire graph. Although some approaches[12–14] effectively extend the scope of information transfer by increasing the depth of the network, these approaches are still inherently limited by the global nature of the spectral graph theory.

[1]School of Computer Science and Information Engineering, Harbin University of Commerce, Harbin 150028, China. [2]College of Geo-Exploration Science and Technology, Jilin University, Changchun 130026, China. [3]Yu Zhang and Xin Li contributed equally to this work. ✉email: xuyq@hrbcu.edu.cn

We provide a brief overview of the common paradigms of graph aggregators (as shown in Fig. 1). First, local aggregators exhibit significant limitations when modeling long-range dependencies in a graph, failing to effectively capture global interactions between nodes. Furthermore, the representational power of these methods is often constrained by the assumption of homogeneity in the input graph, which particularly hinders their performance in heterogeneous graphs. In contrast, the transformative applications of Transformer models in language and vision tasks[15,16] have provided insight into the design of graph aggregators. Transformers enable global interactions between nodes, overcoming the limitations of local aggregators. Consequently, graph Transformers (GT), which combine GNNs with Transformers, have received significant attention in recent years. However, existing GT models still face several key challenges, which can be summarized as follows:

Firstly, stacking additional layers onto the vanilla Transformer architecture[17–19], which operates with quadratic computational complexity, may lead to redundant computations and, in some cases, fail to meet the training requirements for those containing tens of thousands of graph datasets[20]. Secondly, the inherent disorder of graph structures complicates the direct application of encoding strategies developed in the language and vision domains. To capture local positional information of nodes, computationally expensive graph traversal processes are often necessary[19,21–26], and the resulting encoding information is typically used as a soft bias in node features, often compressed into low-dimensional vectors, resulting in the loss of the original local diversity present in the topological structure. Thirdly, while some studies have utilized sparsification strategies[27,28] or decomposable linear kernel operators[20,29–31] to achieve attention computation with linear complexity, the softmax attention mechanism, which uses non-linear weighting, is crucial in supporting feature diversity[32,33], allowing the model to focus on the most salient features. As a result, these linear Transformer models may suffer from feature representation distortion. Recent benchmark tests[33,34] show that while these linear GT models have demonstrated some success on large datasets, they still fall short of the performance of classical GNN models on several main datasets, raising doubts about their actual advantages.

In this study, our study focuses on investigating the potential of Transformers in node embedding representation learning, as opposed to previous approaches that compressed model training time through efficient relaxation strategies. To address this, we conducted the following principled research: First, we adopted the same non-linear mapping to generate queries and keys, allowing the definition of a symmetric positive-definite kernel between any pair of nodes, thus effectively capturing pairwise node similarities. Simultaneously, we discarded the multi-layer multi-head attention mechanism to minimize redundant computations and laid the groundwork for more flexible design techniques in the self-attention layer. Second, to address the challenge of graph node disorder, we introduced an innovative permutation-invariant positional encoding strategy that challenges the conventional approach of encoding the entire graph. Unlike traditional methods that treat positional encoding as a latent bias for node features, we integrated potential information from node class labels and local spatial relationships, using structural encoding as a latent bias for attention coefficients. This approach facilitates efficient transmission of messages across the graph. Moreover, since our attention coefficients are derived from a single non-linear mapping of a positive-definite kernel and utilize a simple single-head attention mechanism, we are able to more effectively ensure that connected nodes of the same class exhibit higher connection weights. However, the main Laplacian smoothing losses[10,35] primarily focus on the graph structure, without explicitly constraining the correlations of the node class. This is especially problematic in fully connected communication within heterophilous graphs, where such methods may inadvertently mix the label information

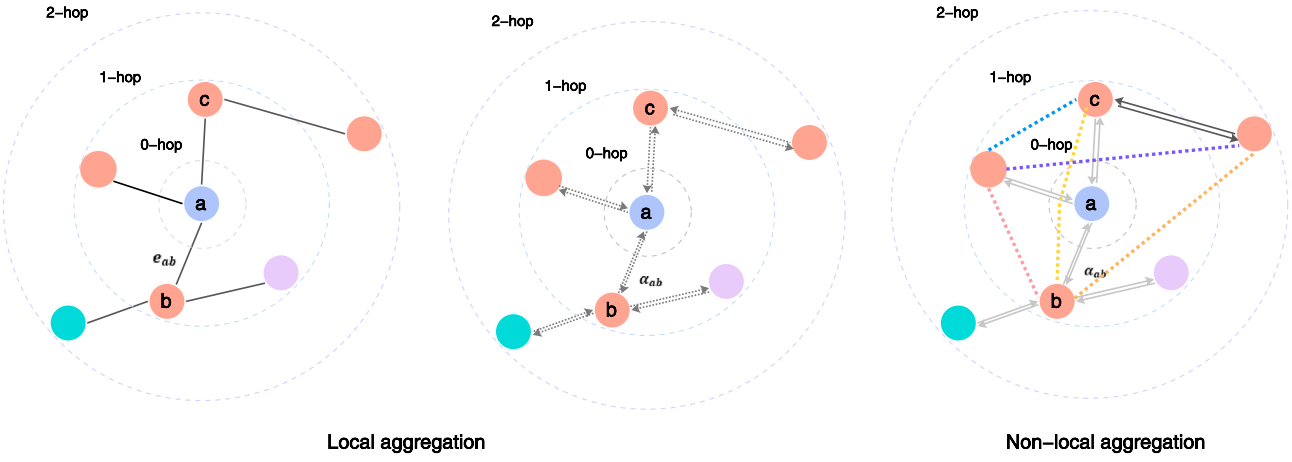

Fig. 1. We show a schematic of different graph aggregation methods. First, for the standard MPNN paradigm (left), we cannot accurately determine the contribution of each node to the final aggregated output due to the lack of explicit interpretability of the aggregation operations among nodes. Second, for a local aggregator with an attention mechanism (middle), node b, when aggregating information from node c, first needs to integrate information from neighboring nodes with different labels, an operation that may result in aggregated information that may not necessarily be beneficial to the task. Therefore, in order to more accurately capture the relationship between remote and similar nodes in the graph, we introduce a non-local aggregator (right) to directly obtain useful information from long-distance nodes.

of different classes. To overcome this, we approach the maximization of neighborhood homogeneity for both training and prediction nodes as a convex hull problem, treating node signals as probability distributions after appropriate processing. The Kullback–Leibler (KL) divergence between nodes then measures the similarity of their probability distributions. By balancing these divergence values with the attention scores between nodes, we can significantly enhance the representation of node relationships, promote neighborhood homogeneity, and improve the model's sensitivity to label class differences.

Ultimately, we introduce an optimized graph Transformer model (OGFormer), which seamlessly integrates the core principles of Transformer and GNNs for node embedding representation learning. To validate the effectiveness of our approach, we conducted node classification experiments in 10 challenging benchmark datasets, comprising 6 homophilous and 4 heterophilous datasets. The experimental results reveal that OGFormer consistently delivers superior performance compared to the powerful MPNN across all datasets, demonstrating the ability of our method to effectively learn richer node representations from multi-hop neighborhoods. Due to its innovative encoding strategy and neighborhood homogeneity maximization loss, OGFormer is highly competitive against existing GT techniques, and in some cases, it even outperforms current methods. The key contributions of this paper are summarized as follows:

- We introduce the OGFormer framework, which departs from the traditional multi-layer multi-head attention mechanism by adopting a single attention head and a symmetric positive-definite kernel. This approach opens up additional possibilities for designing attention layers that facilitate node embedding representation learning. Notably, this work is the first to optimize global attention scores between nodes through a variety of strategies, conceptualizing this as a method for learning fine-grained graph structures, thereby further uncovering the potential of Transformers in node classification tasks.
- We reconsider the prevailing approach in mainstream methods, where positional and structural encodings are treated only as soft biases for node features. We introduce a permutation-invariant, unbiased structural encoding strategy. This strategy is derived directly from the input graph through simple transformations, taking into account the latent patterns within node neighborhoods and the spatial relationships of node positions, without the need for additional pre-processing.
- To overcome the limitations of the Laplacian smoothing loss in heterophilous graphs, we introduce the loss of maximization of neighborhood homogeneity. This approach optimizes attention scores by integrating the differences in probability distribution between node features with the neighborhood homogeneity rate. Using this strategy, the global attention mechanism in OGFormer can more effectively capture information from distant but similar nodes.
- We have highlighted the importance of optimizing attention scores between nodes and incorporating node position information. However, the explicit computation involved in quadratic complexity poses a significant challenge. This finding motivates us to reevaluate the design of GT models for node-level prediction tasks.

Related work

In this section, we describe the work that is relevant to us.

Node classification

Given an undirected and unweighted graph $G = (V, \mathscr{E})$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes and E represents the edges that indicate the connections between the nodes. We represent the connection relationships between the nodes using the adjacency matrix $A \in \mathbb{R}^{n\times n}$. Next, we define the one-hop neighborhood of node v_i as $\mathscr{N}_i = \{v_j \mid v_j \in V, (v_i, v_j) \in \mathscr{E}\}$, which contains all the nodes directly connected to v_i. Taking into account a semi-supervised node classification task on the graph G, each node v_i corresponds to a node characteristic $x_i \in \mathbb{R}^d$ (a vector of features) and a class label y_i. Our objective is, given a set of nodes labeled on the graph G, to learn a mapping function $f : V \to Y$ that predicts the class labels for all nodes, minimizing the loss function that measures the difference between the predicted and the true values. This loss function is typically based on the cross-entropy loss.

Graph neural network

GNNs have become a powerful technique for modeling graph data. With the development of deep learning, many graph convolution methods have emerged, which can be divided into spatial methods[35] and spectral methods[1,2]. Later, Gilmer et al.[5] provided a unified framework for these methods, known as Message Passing Neural Networks (MPNNs), which describes the general form of the node update equation of GNNs in terms of two stages: the aggregation mapping stage AGG^l and the combination mapping stage COM^l. In the aggregation mapping stage, the central node collects information from its neighbors through a permutation-invariant function (e.g., MEAN, MAX, SUM), and then in the combination mapping stage, the information from the neighbors is fused into the node representation. This process can be simply expressed as the following equation:

$$x_i^l = COM^l\left(x_i^{l-1}, m_i^l\right), \quad where \quad m_i^l = AGG^l\left(\left\{x_j^{l-1} | j \in \mathscr{N}_i\right\}\right) \tag{1}$$

where N_i is the neighborhood of node v_i. Moreover, both AGG^l and COM^l can be learned networks, and their different implementations give rise to specific architectures. Examples include different designs for neighborhood construction and permutation-invariant functions[36,37], linear aggregation[2], and attention-based aggregation[9,11]. This flexibility allows MPNNs to adapt well to different tasks, especially in handling node similarities or edge-encoding problems.

Laplacian regularization
The Laplacian regularization of a graph uses the structure of the graph to smooth node representations. Penalizes squared differences between the representations of adjacent nodes, encouraging their representations to be more similar, thus enhancing the smoothness of the model graph structure. This regularization method is typically represented as

$$\mathscr{L}_{Lap} = \sum_{(v_i, v_j) \in \mathscr{E}} ||f(x_i) - f(x_j)||^2 \quad (2) \tag{2}$$

where $f(x_i)$ and $f(x_j)$ are the embedded representations of nodes v_i and v_j, respectively. These embeddings can be either the node feature vectors[35] directly or embeddings obtained through the MPNN network[10]. By minimizing this loss function, the goal is to make the representations of adjacent nodes as similar as possible. This process approximates an infinitely deep GCN[2] model[12].

Transformer

The Transformer architecture[15] was initially applied to natural language processing tasks and has achieved remarkable success in various fields. The Transformer encoder is composed of multiple stacked layers, each consisting of two main components: the Multi-Head Self-Attention (MSA) mechanism and the Feed-Forward Network (FFN). In the self-attention mechanism, each input position interacts with other positions through queries (Q), keys (K), and values (V) to gain information about the other positions. Specifically, given an input sequence of length n, the input matrix $H \in \mathbb{R}^{n \times d}$ (where d is the embedding dimension) generates queries, keys, and values through non-linear transformations:

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V. \tag{3}$$

The core idea of self-attention is to compute the similarity between the queries and keys. This similarity allows each input sequence to focus on other positions, thus capturing long-range dependencies between the sequences. Then, based on the generated similarity matrix S, the values are weighted and summed to obtain the final result:

$$\mathrm{Attn(Q, K, V)} = \mathrm{softmax(S)V}, \quad where \quad S = \frac{QK^T}{\sqrt{d_k}} \tag{4}$$

Note that for simplicity, we assume that Q and K have the same dimensionality, and we omit the bias and activation functions, considering only single-head attention, since multi-head attention is a natural extension of single-head attention.

Graph transformer
GT, as an innovative approach to integrating graph structural information into Transformer architectures, has made notable advancements in graph data neighborhood learning. GT models employ self-attention mechanisms that ensure that all node pairs in the graph are interconnected, enabling effective learning of global dependencies between nodes. The core methods of existing GT models can be broadly divided into two main categories:

(I) Design of encoding strategies: In a manner similar to the language and vision domains, encoding strategies aim to help the model capture contextual relationships within the graph, typically defined as local connectivity relationships within the prior graph. For example, node positional information can be derived by combining node importance[19,21], shortest path distances[22], Laplacian eigenvectors[23–25], and graph domain knowledge[26] (such as the diagonal elements of the random walk matrix, triangles, and cycles), and this positional information is then used as a soft bias for node features, improving graph representation. However, the unordered nature of the graph means that these encoding strategies, with their non-trivial computational overhead, may not always be directly effective in all cases[24,31].
(II) Design of the model architecture: Building upon the quadratic kernel Vanilla Transformer, the architecture stacks multi-head attention layers[17,18] and cross-attention layers[19] to form a unified model structure, leveraging encoding strategies to maximize the use of prior graph structural information. To mitigate the computational burden of these complex models, researchers have employed feature sampling[24] or sparse strategies[27,28] to reduce the computation, although at the cost of reducing the model's ability to express interactions between all nodes. An alternative approach employs decomposable kernel operators[20,29–31] to achieve linear complexity attention computation, then combines (e.g. parallel[29,30], global-to-local[20], and local-to-global[31]) MPNN models, avoiding reliance on preprocessed graph encoding information to guide information propagation on the graph.

Despite this, the full potential of GT has yet to be realized. This is due to the inherent characteristics of graph data, which make it challenging for low-dimensional graph encoding information to adequately capture the local structural relationships between nodes. Furthermore, linear kernel GT models, by explicitly decomposing queries and keys, lose the ability to learn fine-grained graph structures, such as the focusing capability provided by the softmax mechanism. In this study, we have simplified the design of the Vanilla Transformer as much as possible and further explored the potential of Transformers for node-level prediction tasks. By employing a structure encoding approach that does not require preprocessing, the model can focus more effectively on key local dependencies between nodes while optimizing an attention score loss function to learn more refined graph

structures. Our experimental results demonstrate that, on most medium- to small-scale datasets, the proposed method significantly outperforms existing powerful GNN models and state-of-the-art linear GT models in terms of inference capabilities.

Methods

In this study, we introduce the OGFormer model, an innovative framework designed for optimized attention scores. As depicted in Fig. 2, the model consists of three main modules: the Single-Head Self-Attention (SSA) module, the Structural Encoding (SE) module, and the loss module for end-to-end optimization of attention scores. In the following sections, we first present the overall framework of OGFormer and then provide a detailed explanation of the implementation of each module.

OGFormer layer

In the context of the semi-supervised node classification task on graphs, we treat the set of nodes as tokens. Unlike language or vision data, which have explicit sequential structures, graph nodes are numerous and unordered. As a result, instead of adopting the multihead transformer encoder proposed by Vaswani et al.[15], we introduce the SSA mechanism. This simplified attention structure reduces unnecessary computation while effectively capturing global dependencies between nodes in the graph. Furthermore, as noted by Wu et al.[20], information on nodes can be propagated layer by layer in a densely connected attention graph. Thus, we eliminate the FFN module that follows MSA, avoiding redundant processing.

Let $Z^0 = X$ represent the initial embedding state of the node, and let the node information be updated layer by layer using the SSA function. To streamline the structure and mitigate the issue of forgetting features caused by deep network stacking, we incorporate residual connections. The node information update formula for the -th layer of OGFormer is given by:

$$Z^l = \mathrm{SSA}(Z^{l-1}, SE^l) + f(Z^{l-1}) \tag{5}$$

where Z^l denotes the node embedding in layer , f is a neural network feedforward layer, and the SSA function, guided by SE of each layer, updates the node information. The detailed implementation of the SSA function will be elaborated in the following sections.

Single-head self-attention mechanism

In the context of graph representation, for the embedding of a specific node, we first apply a non-linear transformation to map the query Q and key K into a shared latent space, as follows:

$$Q = K = \mathrm{Sigmoid}(ZW + b) \tag{6}$$

In conventional transformers, Q and K are projected into different subspaces. In our design, both the query and the key share the same transformation matrix and bias. This symmetry in the attention score matrix facilitates the description of global dependencies between nodes, a design that is especially advantageous for node classification tasks on undirected graphs[35]. Additionally, using shared transformation matrices reduces the number of parameters and the computational complexity of the model.

Next, similarly to the calculation of correlation coefficients, we compute the dot product of the transpose of Q and K (after removing dimensionless factors), resulting in an attention score matrix $R \in \mathbb{R}^{N\times N}$, which represents node dependencies. The dependency R_{ij} between the nodes v_i and v_j is given by:

$$R_{ij} = \mathrm{Norm}_{j\in V}\left(\left(\hat{Q}_i \cdot \hat{Q}_j\right)^2\right), \text{ where } \hat{Q}_i = \frac{Q_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}} \tag{7}$$

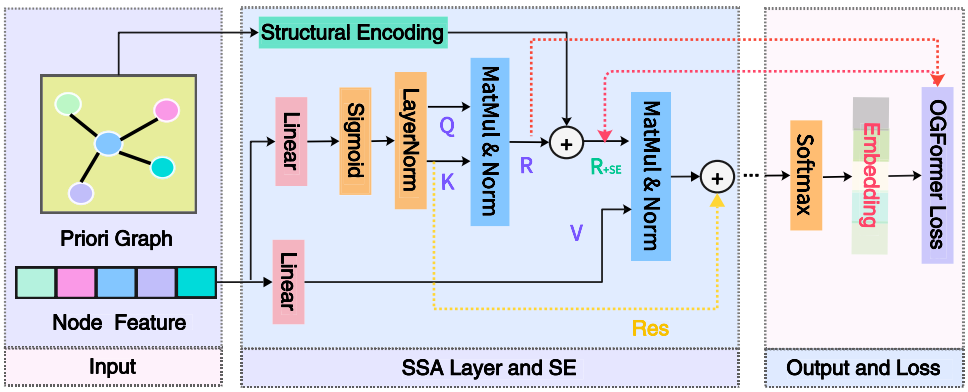

Fig. 2. Simplified architectural diagram of the proposed OGFormer model.

Here, $\mathrm{Norm}_{j\in V}$ refers to the normalization operation on the attention score matrix, which can be Softmax normalization, random walk normalization (similar to GAT[9]), or symmetric normalization (similar to GCN[2]). The small value ϵ prevents the division by zero. For clarity, the layer indices l for Q and R have been omitted. Since correlation coefficients typically range from -1 to 1, we square the result to treat attention scores as probabilities for message passing between nodes.

Finally, in the node embedding update process, node information is received from the embeddings of the previous layer through the SSA function. After computing the attention score matrix R, we perform a weighted sum of node embeddings, resulting in the updated node representation. The specific update formula for the -th layer is:

$$Z_i^l = \mathrm{SSA}(Z_i^{(l-1)}) = \mathrm{ReLU}\left(\sum_{j\in V} R_{ij} W^{(l-1)} Z_j^{(l-1)}\right) \tag{8}$$

It is important to note that the SSA function is permutation invariant. Like GAT, it models the influence of neighboring nodes through weighted averaging. However, unlike GAT, SSA treats all nodes in the graph as neighbors, allowing it to more comprehensively capture node dependencies.

Structural encoding

In the previous section, we defined the SSA function, which effectively captures the dependencies between the remote nodes in the graph. However, local structural information of the nodes is crucial for graph representation learning, especially in homophilous graphs, where nodes that are connected to each other often share the same label. This property enables MPNNs to perform particularly well on such types of graphs. Therefore, we believe that it is essential to effectively incorporate the local structural information of the nodes into the SSA module.

As mentioned in the Introduction, many existing GT models typically introduce local structural information from the node as a soft bias in the node features, for example, by adding or concatenating operations. These methods often compress node structural information into low-dimensional vectors and fuse it with node features to avoid significantly increasing feature dimensions. Typical approaches include using the first few non-trivial Laplacian eigenvectors[23–25], node degree centrality[19], or sampling the shortest path distances between nodes[22]. However, these encoding strategies tend to sacrifice the true local relationships between nodes and the improvement in the inference capability is often limited to small ranges.

To address this issue, we consider incorporating structural information from prior graph nodes SE_{ij} as an unbiased bias term for attention scores.

$$\hat{R}_{ij} = \mathrm{Norm}_{j\in V}\left(R_{ij} + SE_{ij}\right) \tag{9}$$

Ideally, we wish to learn a parameter θ that further explains the relative importance of the connection between each pair of nodes in the neighborhood, as follows:

$$SE_{ij} = \frac{\exp(\theta A_{ij})}{\sum_{j\in N_i} \exp(\theta A_{ij})} \tag{10}$$

However, the major challenge of this strategy lies in its computational overhead. Due to the sparsity of the graph, matrix sparsification and dense conversion operations are often needed, which undeniably reduce efficiency. Fortunately, the neighborhood average homophily rate[37] in the graph provides a new perspective. For the node classification task, the average homogeneity rate of a graph's neighborhood can effectively represent the importance of the connectivity relationship between nodes. An intuitive observation is that if all nodes in the neighborhood have different class labels from the center node, the influence of the connections in that neighborhood on the attention scores is minimal. On the other hand, if the labels are the same, the influence is stronger.

To prevent label leakage, we introduce a hyperparameter α to estimate the importance of known topology in the graph, combining it with the learned fine-grained graph structure, ie $SE_{ij}^l = \alpha R_{ij}^{(l-1)}, \quad SE_{ij}^0 = \alpha A_{ij}$. We recommend that for homophilous graphs, α range between [0.7, 1], and for heterophilous graphs, α should range between [0, 0.2]. Finally, we define the Structural Encoding (SE) combined with the SSA propagation layer as:

$$Z_i^l = \mathrm{ReLU}\left(\sum_{j\in V} \hat{R}_{ij} W^{(l-1)} Z_j^{(l-1)}\right), \quad \text{where } \hat{R}_{ij} = \mathrm{Norm}_{j\in V}\left(R_{ij}^l + SE_{ij}^l\right) \tag{11}$$

It should be noted that the SSA module, combined with structural encoding (SE), maintains permutation invariance in its design. This means that the module can effectively handle inputs from different graph structures and maintain stability in unordered graphs. Moreover, unlike compressed encoding strategies, the proposed SE scheme fully reflects the local connectivity relationships in the graph, allowing it to perform effectively in more complex graph representation learning tasks. In Section 4, we will demonstrate the superior performance of OGFormer in classification tasks, further validating the importance of the SE module in graph representation learning.

Neighborhood maximum homogeneity loss

In the previous section, we introduced the SSA module and designed it as a global extension of the local MPNN. However, given the complexity of graph structures, dependencies between nodes cannot always be accurately described by simple dot products. Inspired by the powerful capability of MPNNs on topologically similar graphs, to capture node dependencies more precisely, we treat the attention score matrix $\hat{R}_{ij}$ as the posterior probability under the prior condition ϕ, which can ideally be expressed as:

$$P(\hat{R}_{ij}|\phi) = \begin{cases} 1 & \text{if } Y_i = Y_j \\ 0 & \text{else} \end{cases} \tag{12}$$

In this expression, our aim, through the prior condition ϕ, is to ensure that connected nodes of the same class are linked, while nodes of different classes are not. However, given the finite label set, approximating this hypothesis is extremely challenging. As discussed in the previous section, we interpret ϕ as a learnable iterative attention score in Eq. (7), but it is insufficient to approximate this hypothesis alone.

In prior work, additional metrics have been defined in the latent space to better understand the finer dependencies of nodes. For example, Dornaika et al.[10] imposed explicit attention constraints through Euclidean distances between node features, achieving results superior to the GNN main models in node classification tasks. A natural approach is to introduce additional metrics into the attention score matrix to explore node dependencies as much as possible. However, attention is computed at every layer, which increases computation time. Furthermore, it is difficult to measure which metric in the space will always dominate. Therefore, introducing an additional relational operator between node embeddings as a loss function to enrich the representation of dependencies between nodes is undoubtedly a better choice because it allows us to adjust the influence of the loss via hyperparameters over a small range.

Unlike Laplacian smoothing-based loss, which is built on local graph structures, the loss in OGFormer should be based on global dependencies. Please note that we will continue to omit the layer index for Q and R.

Since attention scores between nodes are computed through queries Q and their semantic relevance, it is convenient to introduce the Kullback–Leibler (KL) divergence to measure the difference between the probability distributions of the node embeddings. Note that due to the shared mechanism between Q and K, we can easily extend this to symmetric JS divergence. KL divergence requires two conditions: first, all feature values must be in the range [0, 1], and second, the sum of all feature values must equal 1. Therefore, we normalize Q before computing the KL divergence to ensure that it meets the conditions for a valid probability distribution.

In the embedding space, the difference in probability distributions between nodes can be measured as follows:

$$D_{KL}(Q_i||Q_j) = \sum_d Q_i(d) \log\left(\frac{Q_i(d)}{Q_j(d)}\right) \tag{13}$$

Intuitively, when the difference $D_{KL}(Q_i||Q_j)$ between the probability distributions of the nodes in the embedding space is large, we would expect the attention score $\hat{R}_{ij}$ to be low and vice versa, the attention score $\hat{R}_{ij}$ should increase. Therefore, we define the loss function $\mathscr{L}_{KL}$ as:

$$\mathscr{L}_{KL} = \sum_{i,j \in V} \hat{R}_{ij} D_{KL}(Q_i||Q_j) \tag{14}$$

Minimizing $\mathscr{L}_{KL}$ promotes an increase in edge weights between similar nodes and effectively suppresses noisy connections. However, it does not guarantee high homogeneity for each neighborhood. To further strengthen the similarity between the neighborhoods, we introduce a weighted neighborhood homogeneity metric h, which measures the homogeneity within the neighborhood of each node. Taking into account the limited set of labels, we calculate the neighborhood homogeneity for each node using training labels Y_{train} and predicted labels Y_{pred} as inputs. We then use the loss function $\mathscr{L}_h$ to further enhance homogeneity within each neighborhood:

$$\mathscr{L}_h = \|1 - h(\hat{R}_{ij}, Y_{train}||Y_{pred})\|_2, \quad \text{where } h_i = \frac{\sum_{j \in V} \hat{R}_{ij} \text{ and } Y_i = Y_j}{\sum_{j \in V} \hat{R}_{ij}} \tag{15}$$

where $1, h \in \mathbb{R}^n$, and h_i denotes the neighborhood homogeneity rate of node v_i. Combining this with Eq. (9), the Neighborhood Maximum Homogeneity Loss is defined as:

$$\mathscr{L}_{NMH} = \sum_{l=1}^{L} \lambda_1^l L_{KL}^l + \lambda_2^l L_h^l \tag{16}$$

where λ_1^l and λ_2^l are hyperparameters used to adjust the influence of the losses $\mathscr{L}_{KL}^l$ and $\mathscr{L}_h^l$ in the corresponding network layers. For semi-supervised node classification tasks, we sum $\mathscr{L}_{NMH}$ with the cross-entropy loss to form the final loss function for OGFormer. At this point, we consider the attention score operator combined with the loss function as the final prior condition in Eq. (12).

Experiments

In the experimental section, we will conduct extensive evaluations to assess the performance of OGFormer on node classification tasks. Since many experimental results may be influenced by training settings and overfitting—particularly when using a single training/validation/test split, which can introduce superficial statistics and experimental bias[38,39]—we specifically perform multiple well-established benchmark split experiments and compare the results with the baseline methods for each respective split, thus demonstrating the advantages of OGFormer. Additionally, we plan to carry out ablation experiments to evaluate the importance and necessity of each module within OGFormer, ensuring that each component contributes to the final performance. Moreover, we will quantitatively assess the learned dependencies, with a particular focus on the model's ability to capture global information between nodes in the graph. Through these analyses, we aim to further confirm the effectiveness of OGFormer in graph representation learning and compare its performance with existing methods, ultimately verifying its competitiveness in handling complex graph-structured data.

Datasets

We summarize the details of each dataset in Table 1. where h is the average neighborhood homogeneity rate.

Homophilous graphs

Cora, CiteSeer, and PubMed are three widely utilized citation networks. We adopt the classic fixed splits for semi-supervised settings, consistent with the methodology in references[1,2,20]. Furthermore, following the approach suggested by Shchur et al.[39], we conducted experiments with 100 random initialization training/validation/test splits and computed the average precision, ensuring a balanced class distribution in the training split. For the Computer, Photo, and CS datasets[39], we applied the widely accepted 60%/20%/20% train/validation/test split and used accuracy as the evaluation metric. Specifically, the Computer and Photo datasets represent co-purchase networks where nodes represent products, and edges indicate frequent co-purchases of two products. The CS dataset represents a collaboration network, where nodes represent authors, and edges connect authors who co-authored a paper.

Heterophilous graphs

Squirrel and Chameleon are Wikipedia networks[40], where nodes represent Wikipedia pages, and edges represent the interconnections between these pages. Initially, Pei et al.[37] validated their proposed method using their experimental version of the SNAP dataset, which later became the standard version in the literature. Subsequently, Platonov et al.[38] conducted a series of heterophilous graph benchmark tests and discovered that the original splits of these datasets introduced overlapping nodes between the training and testing sets. They proposed a new data splitting method that filters out these overlapping nodes. Consequently, we evaluate the method using both the evaluation approach of Pei et al. and the most recent splits provided. The Wisconsin and Texas datasets are derived from the WebKB dataset, collected by Carnegie Mellon University, where the nodes represent the web pages and the edges represent the hyperlinks between them. We adopted the same splitting method used by Pei et al. and Zhu et al.[41].

Model settings and baselines

Model settings

All experiments were carried out on a system configured with 15 vCPUs, an Intel (R) Xeon (R) Platinum 8474C CPU and a single NVIDIA GeForce RTX 4090D (24GB) GPU. We implemented our approach and related methods using PyTorch[42] and PyTorch Geometric[43]. Except for the experiments on the latest Squirrel and Chameleon datasets, which used a 3-layer network, all other experiments used a 2-layer network with hidden layer sizes of 32 or 64. All trainable parameters were initialized using Glorot initialization and we employed the Adam optimizer. The hyperparameter search space for OGFormer includes: learning rates 0, 0.005, 0.01, 0.02, 0.05, weight decay 5e−4, 8e−4, 1e−3, and dropout rates 0, 0.2, 0.5. We also implemented a learning rate decay strategy to ensure convergence within the predefined number of epochs. Furthermore, the hyperparameter in Eq. (11) was set to 0.8 for homophilous graphs and a smaller value for heterophilous graphs. We recommend using the search space for the hyperparameters in Eq. (16) as 0, 1e−3, 1e−4, 1e−5, 1e−7, with decay applied as the

Datasets	Nodes	Edges	Features	Classes	h	Type
Cora	2708	5429	1433	7	0.83	Homophily
Citeseer	3327	4732	3703	6	0.71	Homophily
Pubmed	19,717	44,324	500	3	0.79	Homophily
Computer	13,752	245,861	767	10	0.80	Homophily
Photo	7650	119,081	745	8	0.84	Homophily
CS	18,333	81,894	6805	15	0.85	Homophily
Chameleon	2277	36,101	2325	5	0.25	Heterophily
Squirrel	5201	217,073	2089	5	0.22	Heterophily
Texas	183	309	1703	5	0.06	Heterophily
Wisconsin	251	499	1703	5	0.16	Heterophily

Table 1. Details of the 10 datasets used in this work.

number of network layers increases. Further details of the experimental setup can be found in our open-source code.

Baseline

We conducted extensive comparative experiments between OGFormer and three categories of state-of-the-art baseline methods. These baselines include four advanced local GNN models: GCN[2], GAT[9], APPNP[44] and GraphSAGE[36]; Four linear kernel Transformer models integrated with local GNN modules: GraphGPS[23], NodeFormer[29], SGFormer[20], and Polynormer[31]; and three powerful GT models: Graphormer, which has quadratic computational complexity, Exphormer[27], which uses sparse sampling strategies, and NAGphormer[24], which represents neighborhood features as tokens. Additionally, we examined two non-local GNN models for node classification on heterophilous graphs: GEOM-GCN[37] and H2GCN[41]. It is important to note that for all of the aforementioned baseline methods and their variants, we utilized the best configurations reported in the literature. For the remaining results, we sourced them as closely as possible from their original papers or publicly available rankings, ensuring that the comparisons were based on models that had been thoroughly tuned.

Results and analysis

Comparison with baseline methods

Tables 2 and 3 present a comparison of OGFormer performance in node classification tasks on both homophilous and heterophilous graphs, compared to several leading baseline methods. Overall, OGFormer exhibits superior performance across the vast majority of test scenarios: its classification accuracy not only surpasses two non-local GNN models specifically designed for heterophilous graphs, but also outperforms six GT models with subquadratic computational complexity and one GT model with quadratic complexity. In particular, OGFormer consistently ranks first in average performance in all experimental configurations. This result provides strong evidence of the significant advantage of OGFormer in handling complex neighborhood relationships and confirms the potential of its self-attention mechanism to effectively capture complex dependencies between nodes.

In terms of specifics, we first use the fixed splitting scheme proposed by Yang et al.[1] for the citation network node classification experiment as a reference. The results in Table 2 reveal that OGFormer improves accuracy by 2.1%, 2.1% and 1.2% over the best-performing GT model. In particular, Graphormer, which shares the same computational complexity as OGFormer, performs the worst. This can be attributed to the fact that in homophilous graphs, neighboring nodes typically share the same label, and the Graphormer graph structure encoding strategy does not accurately capture the true local topological structure. Furthermore, Graphormer's multi-head attention mechanism increases computational overhead, limiting its training capacity on GPUs with 24 GB of memory, while OGFormer's single-head attention mechanism ensures efficient training on medium- to small-scale datasets. Although GT models with linear computational complexity have a clear advantage in terms of time and resource efficiency, their classification performance is comparable to local GNN models and often significantly worse than OGFormer, suggesting that OGFormer offers a meaningful improvement in inference ability compared to the quadratic computational complexity of the Vanilla Transformer.

Moreover, OGFormer excels in capturing long-range dependencies, and this is not merely an incidental result from different data splits or citation network datasets. For example, in the random splitting scheme proposed by Shchur et al.[39] for the citation network dataset (see Table 3, left), despite the general decrease in performance in all local GNN methods, including OGFormer, it still maintains the best classification performance. In the Computer, Photo, and CS datasets (see Table 2), which also have homophilous properties, OGFormer outperforms four robust GNN models and ranks second among the GT models, trailing only Polynormer. The superior performance of Polynormer is probably due to its unique training strategy, which involves local warm-up training followed by global training, thus improving the model's understanding of node dependencies. In experiments across four heterophilous datasets (see Tables 2 and 3, right), OGFormer consistently demonstrates superior non-local feature extraction compared to GNN models and baseline GT models, regardless of whether the experimental setup follows the work of Platonov et al.[38] or Pei et al.[37].

Method	Cora	Cite	PubMed	Computer	Photo	CS	Chameleon	Squirrel
GCN	81.6±0.4	71.6±0.4	78.8±0.6	89.6±0.5	92.7±0.2	92.9±0.1	41.3±3.0	38.6±1.8
GAT	83.0±0.7	72.1±1.1	79.0±0.4	90.7±0.1	93.8±0.1	93.6±0.1	39.2±3.0	35.6±2.0
GraphSAGE	82.6±0.4	71.9±0.8	79.4±0.5	91.2±0.2	94.5±0.1	93.9±0.1	37.7±4.1	36.0±1.9
APPNP	83.3±0.5	71.8±0.5	80.1±0.2	90.1±0.1	94.3±0.1	94.4±0.0	38.4±3.5	35.3±1.9
GraphGPS	82.8±1.0	*72.7±1.2*	79.9±0.2	91.1±0.5	95.0±0.1	93.9±0.1	40.7±4.0	39.6±2.8
Graphormer	75.8±1.1	65.6±0.6	OOM	OOM	92.7±0.1	OOM	41.9±2.8	40.9±2.5
Exphormer	82.7±1.3	71.6±1.1	79.4±0.3	91.4±0.1	95.3±0.2	94.9±0.0	–	–
NodeFormer	82.2±0.9	72.5±1.1	79.9±1.0	86.9±0.6	93.4±0.3	*95.6±0.2*	34.7±4.1	38.5±1.5
SGFormer	*84.5±0.8*	72.6±0.2	*80.3±0.6*	91.9±0.7	95.1±0.4	94.7±0.2	***44.9±3.9***	*41.8±2.2*
NAGphormer	82.1±1.1	71.4±1.3	79.7±0.2	91.2±0.1	95.4±0.1	***95.7±0.0***	–	–
Polynormer	83.2±0.9	72.3±0.7	79.2±0.4	***93.6±0.2***	***96.4±0.2***	95.5±0.1	41.8±3.4	40.8±1.9
OGFormer	***86.4±0.3***	***74.7±0.5***	***81.5±0.5***	*92.9±0.3*	*95.5±0.0*	95.2±0.1	*43.1±1.5*	***42.2±2.0***

Table 2. Comparison of node classification accuracy on different datasets.

Method	Cora	Cite	PubMed
GCN	79.4 ± 1.9	68.1 ± 1.7	77.4 ± 2.4
SGC	80.2 ± 1.6	68.7 ± 1.6	76.5 ± 2.4
GAT	81.0 ± 1.4	69.2 ± 1.9	78.3 ± 2.3
APPNP	82.2 ± 1.5	70.0 ± 1.4	79.4 ± 2.2
OGFormer	83.1 ± 1.5	71.0 ± 1.6	79.8 ± 2.6

Method	Chameleon	Squirrel	Texas	Wisconsin
GCN	59.8 ± 2.5	36.8 ± 1.3	59.4 ± 5.2	59.8 ± 6.9
GAT	54.6 ± 1.9	30.6 ± 2.1	58.3 ± 4.4	55.2 ± 8.7
GEOM-GCN	60.9	38.1	67.5	64.1
H2GCN	59.3 ± 1.9	37.9 ± 2.0	84.8 ± 6.7	86.6 ± 4.6
OGFormer	67.0 ± 1.4	57.6 ± 1.8	82.1 ± 4.0	86.8 ± 4.3

Table 3. Comparison of node classification accuracy on different datasets.

Additionally, since heterophilous graphs often involve neighboring nodes with different labels, non-local aggregation methods generally outperform local MPNN methods. Even Graphormer, which performs poorly on homophilous graphs because of its inability to effectively reflect local topological structure, outperforms local GNN models. This supports our hypothesis presented in Section 6, which evaluates the importance of local topological structure based on homophilous and heterophilous properties. However, on the Squirrel and Chameleon datasets, the performance improvement of non-local aggregation methods is not as pronounced. In contrast, on the Texa and Wisconsin datasets, global aggregation methods significantly outperform local aggregation methods. This may be due to the stronger correlation between node features and class labels in these datasets, allowing global attention to more effectively capture these higher-level features. However, the smoother feature correlations in the Squirrel and Chameleon datasets make it harder for the attention mechanism to distinguish dependencies between nodes. Therefore, whether this limitation can be alleviated through data augmentation strategies is an interesting avenue for future research.

Ablation study

To fully validate the effectiveness of the proposed method, we decomposed the OGFormer model into its individual components, including SE in Eq. (11), Laplacian smoothing $\mathscr{L}_{Lap}$ in Eq. (2), and the Neighborhood Maximum Homophily Constraint $\mathscr{L}_{NMH}$ in Eq. (16). We then performed node classification experiments on four homophilous and three heterophilous graph datasets, with results shown in Table 4. The experimental results demonstrate that the inclusion of structural encoding significantly improves classification performance on homophilous graphs, especially on the Cora dataset, where the introduction of SE increases the classification accuracy by 15.5%, verifying the effectiveness of structural encoding in strengthening node dependencies. However, on heterophilous graphs, the effect of SE is not always beneficial. For example, on the Texas and Wisconsin datasets, even when the hyperparameter α in Eq. (11) is set to a small value, the classification performance decreases by 0.13% and 1.57%, respectively. Additionally, Laplacian smoothing may exacerbate this issue by improving node connections in low-homophily neighborhoods. Fortunately, the Neighborhood Maximum Homophily Constraint effectively mitigates these noisy connections. As shown in Fig. 3, the overall average neighborhood homophily in the graph increases with the training iterations and eventually stabilizes. Overall, the introduction of structural encoding and the Neighborhood Maximum Homophily Loss significantly improves the optimization of global attention scores, while Laplacian smoothing has certain potential on homophilous graphs but may have limitations on heterophilous graphs.

On the one hand, in order to clearly visualize the global attention of OGFormer, we take 251 nodes on Photo and Wisconsin, and plot the top and bottom two sets of heatmaps in Fig. 4, and the coordinates in each heatmap correspond to the node indexes in the graph. First, we take 2-hop and 7-hop local neighborhoods from Photo and Wisconsin, respectively, and highlight the locally important neighbors of a node by assigning each heatmap value to 1 if the corresponding two nodes have the same label and 0 otherwise. Similarly, we further visualize globally significant nodes as shown in the second graph of each set in Fig. 4. Finally, we train the OGFormer and make it converge, and then visualize the attention matrix of the global attention layer force after linearly adjusting it, as shown in the third graph of each group in Fig. 4. By comparing the globally significant nodes and the attention matrices predicted by OGFormer, it can be observed that many globally significant nodes can be used for accurate prediction of the target nodes. This clearly shows that OGFormer's attention scores effectively distinguish those globally important nodes from unimportant ones.

On the other hand, we projected the learned node embeddings into the 2D space from the node embedding perspective, and then performed T-SNE visualization in the Cora and CiteSeer datasets, respectively, using the fixed segmentation of Yang et al.[1] as the experimental base setup, as shown in Fig. 5 (top and bottom). In order, starting from the highly disorganized initial node features, we compared with GCN and GAT in turn. The results show that it is difficult to effectively distinguish different node types in both datasets, even for the same matching graph. However, after training iterations of the three models, the categories of all nodes become more aggregated and distinguishable in space. Specifically, GCN and GAT are able to capture advanced embeddings between nodes to some extent and distinguish them; however, OGFormer significantly improves the expressiveness of node embeddings by optimizing the global attention scores, further enhancing node differentiation and demonstrating more obvious clustering effects.

Discussion

In this study, our goal was to design an MPNN model that is capable of effectively capturing global dependencies in graphs. Previous local MPNNs effectively aggregate remote node information by stacking network layers, but due to the global nature of graph theory, this approach has limitations in information grouping. Although existing GTs are capable of capturing long-range dependencies, most studies focus on scalability and computational efficiency for large graphs, with less attention on the potential of Transformers for node classification tasks, such

SE	$\mathscr{L}_{Lap}$	$\mathscr{L}_{NMH}$	Cora	Cite	Computer	Photo	Texas	Wisconsin	Chameleon
			68.69	67.50	86.11	91.91	82.02	84.12	40.31
✓			84.19	73.61	92.69	94.50	81.89	82.55	41.73
✓	✓		84.23	73.64	92.88	95.28	81.62	81.39	42.22
✓		✓	86.41	74.77	92.96	95.50	82.16	86.86	43.12

Table 4. Performance comparison of OGFormer with different components on various datasets.

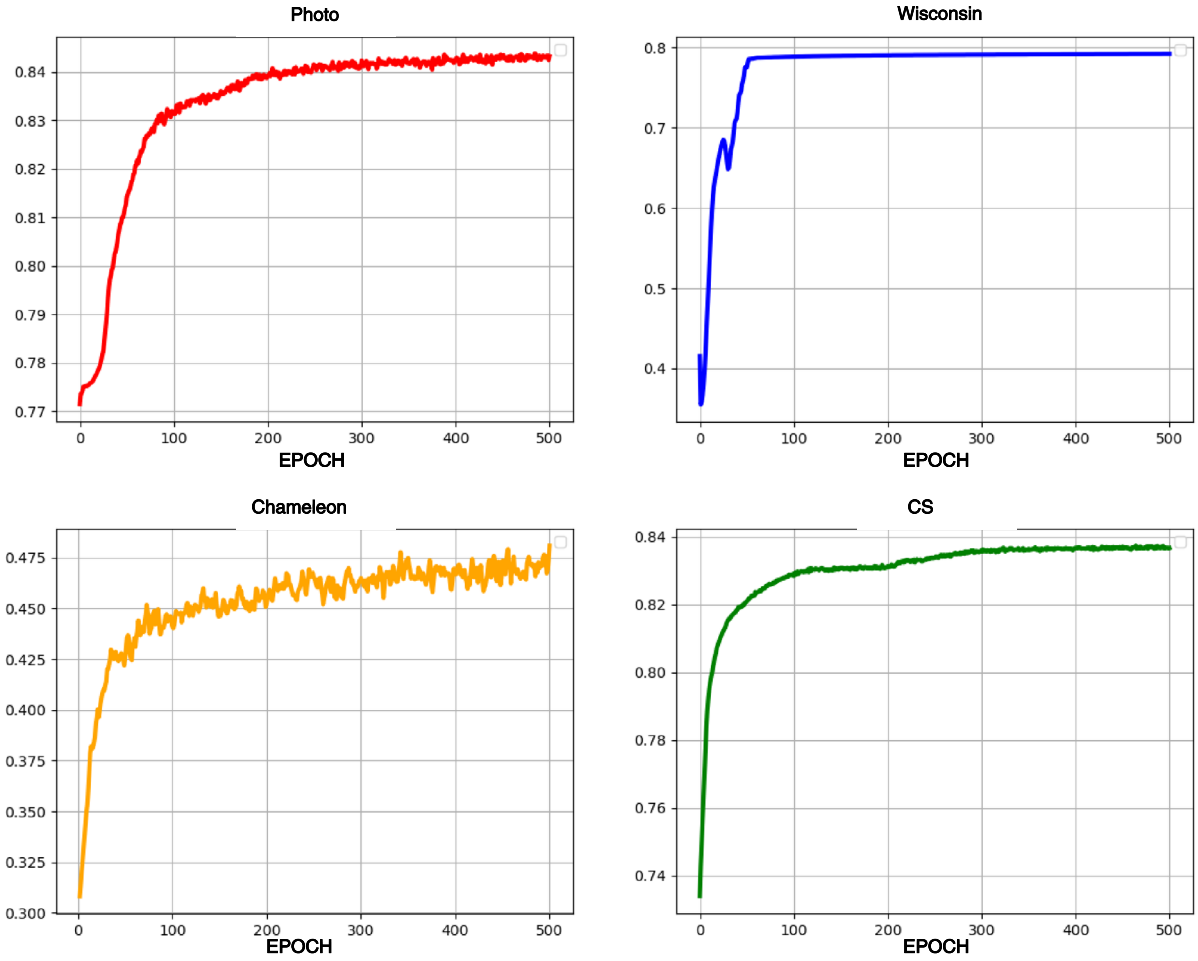

Fig. 3. Images of neighborhood-averaged homogeneity rate changes during training iterations.

as attention score optimization and unbiased structural encoding strategies. To address these issues, we propose the OGFormer model, with the aim of simplifying the model complexity and improving the inference capability. OGFormer captures global information in the graph by sharing attention layers for queries and keys and enhances local information perception through unbiased structural encoding without pre-processing. Given the tight integration of attention score calculation and structural encoding, OGFormer can be viewed as learning a non-local graph structure and propagating information over this structure. Therefore, OGFormer further learns fine-grained graph structures by maximizing the loss function of neighborhood homogeneity, ensuring the effective transmission of information between nodes. Experimental results demonstrate that OGFormer excels in multiple-node classification tasks and confirm the significance of the proposed graph structure encoding strategy and end-to-end optimization of attention scores between nodes.

Limitations and future challenges

Although OGFormer utilizes a simplified attention layer without the need for complex preprocessing, the computation and optimization of its pairwise attention scores still incur quadratic complexity, posing scalability challenges for large-scale graph applications. Compared to GT models of quadratic complexity based on multi-head attention[17,19], OGFormer reduces both computational overhead and model complexity. However, compared to GT methods of approximate linear complexity of state-of-the-art[20,24,27,29–31], its computational cost remains significantly higher. Notably, these efficient methods generally underperform OGFormer in medium- to small-scale node representation learning tasks, a disparity arising from an inherent contradiction between Transformer models and graph structure data:

(I) Trade-off between model expressiveness and computational efficiency: Quadratic complexity GT models can explicitly construct attention matrices and model precise node dependencies using techniques such as structural encoding, loss constraints, and normalization strategies. In contrast, most linear complexity GT models rely on implicit attention approximations[45] (e.g., kernel operations), which results in the loss of key structural operators. For example, Wu et al.[29] attempted to approximate the Softmax normalization operator using positive random feature kernels; theoretically, when the temperature coefficient is sufficient-

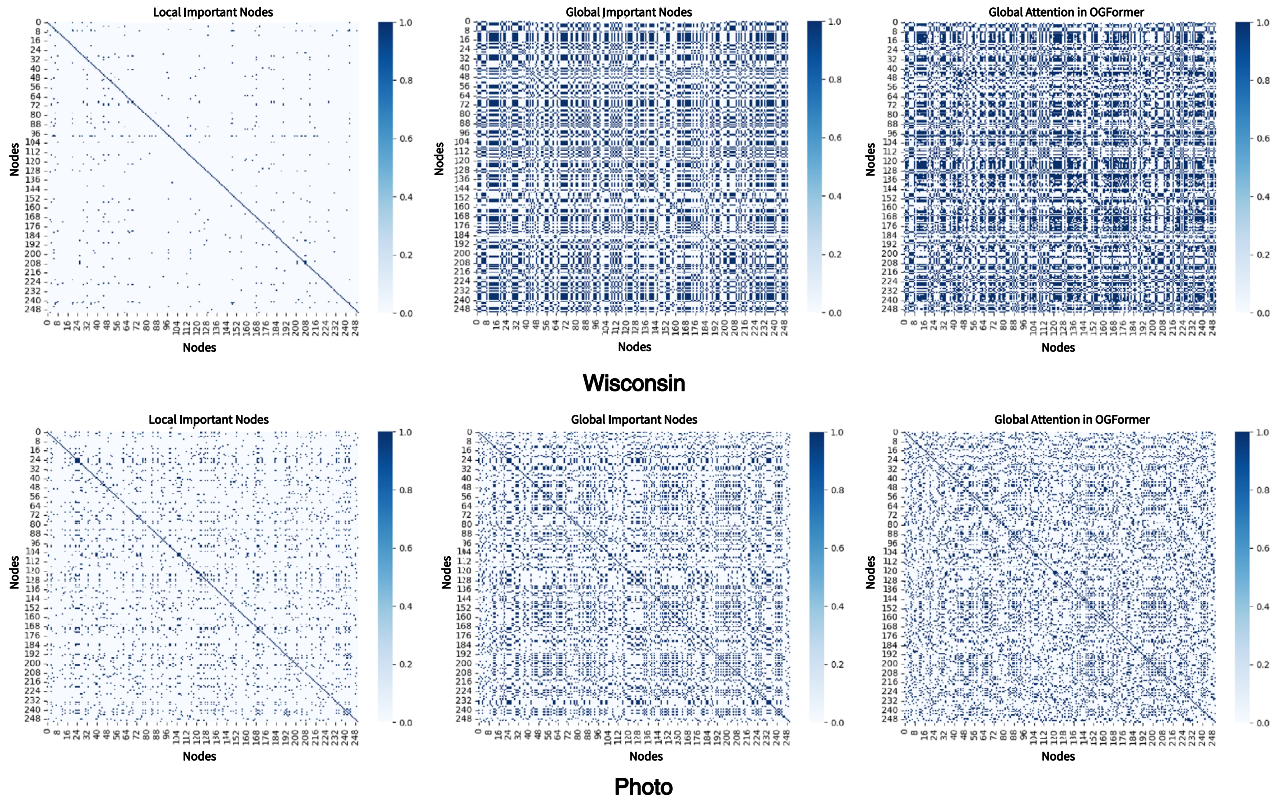

Fig. 4. Node connectivity relationship importance visualization . Where higher heatmap values indicate greater importance. The locally and globally important nodes in each row have the same labels and the homogeneous neighborhoods of these nodes are predicted based on the corresponding global attention in OGFormer.

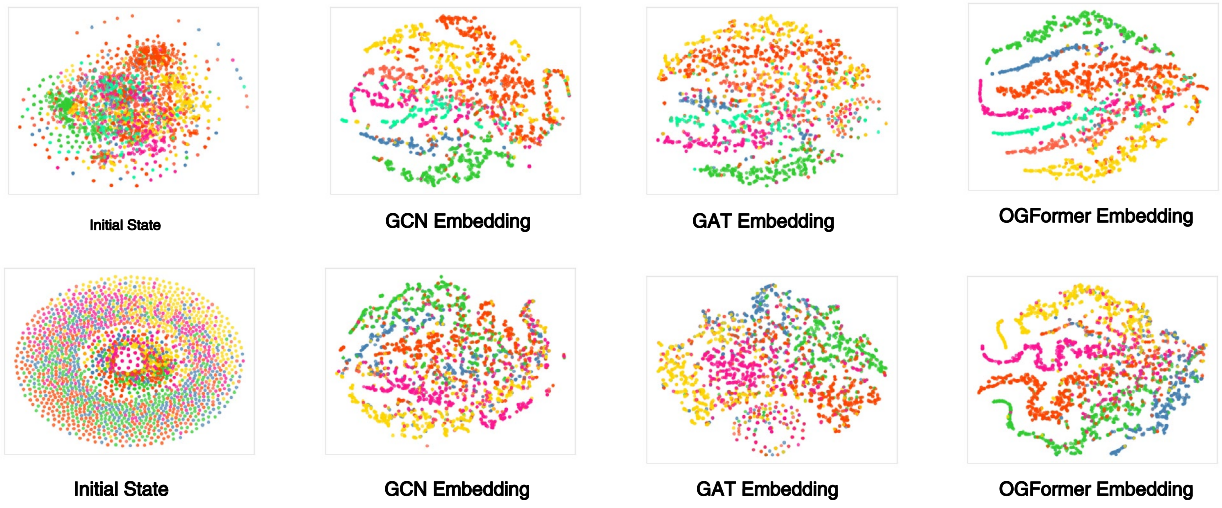

Fig. 5. Comparison of T-SNE visualizations of node embeddings learned by GCN, GAT and OGFormer ..

ly small, unbiased estimation can be achieved, but this significantly increases the computational burden. Moreover, a larger temperature coefficient can cause global attention to degrade into a mean-pooling operation. Additional efficient strategies, such as graph coarsening[24] or sampling[27], further sparsify global dependencies, thereby reducing the model's expressive power.

(II) The irreconcilability of graph structure fidelity and quadratic complexity: In semantics and vision, the ordered nature of token positions enables Transformers to efficiently integrate low-dimensional position encodings (e.g., local window strategies[46]), reducing computational complexity to near-linear levels. However, the permutation invariance of graph data means that its structural information is inherently an unbiased encoding. If compressed into low-dimensional feature biases[23,25], topological fidelity is lost; if used as a bias for attention scores, the complexity reverts to quadratic. As a result, existing linear kernel Transformers often couple local GNN modules to model graph structure diversity but are constrained by the inherent limitations of linear kernel operators and neighborhood aggregation.

Conclusion

We investigated the potential of Transformers for node classification and proposed the OGFormer model, featuring an innovative single-head self-attention layer, an unbiased structural encoding strategy, and a neighborhood maximum homogeneity loss function. The model achieved outstanding results across multiple well-known benchmark datasets, validating its effectiveness in node embedding representation. However, optimizing attention scores remains a substantial challenge. In particular, the quadratic complexity design limits its scalability to large graphs. As a result, future research could focus on enhancing computational efficiency, enabling the model to scale to larger graph datasets or to be applied to tasks with shorter sequence lengths, thus capitalizing on its strengths in smaller-scale graphs or tasks.

Data availibility

All datasets used in this article are available in the GitHub repository: https://github.com/kimiyoung/planetoid/raw/master/data; https://github.com/shchur/gnn-benchmark/raw/master/data; https://github.com/pyg-team/pytorch_geometric/tree/master; https://graphmining.ai/datasets/ptg/wiki. In addition, details of the implementation of OGFormer are available from: https://github.com/LittleBlackBearLiXin/OGFormer.

Received: 21 March 2025; Accepted: 8 August 2025
Published online: 16 August 2025

References

1. Yang, Z., Cohen, W. & Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, 40–48 (PMLR, 2016).
2. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)* (2017).
3. Zhao, T., Zhang, X. & Wang, S. Disambiguated node classification with graph neural networks. *Proc. ACM Web Conf.* **2024**, 914–923 (2024).
4. Klepl, D., Wu, M. & He, F. Graph neural network-based eeg classification: A survey. *IEEE Trans. Neural Syst. Rehabil. Eng.* **32**, 493–503 (2024).
5. Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O. & Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, 1263–1272 (PMLR, 2017).
6. Huang, Y., Zhang, Z., Zhao, W. & Hu, H. Spatial-temporal modeling for multi-scale wind speed predictions in rail transit networks: A graph neural network-based methodology. *Eng. Appl. Artif. Intell.* **158**, 111477 (2025).
7. Li, D. et al. Sdformer: A shallow-to-deep feature interaction for knowledge graph embedding. *Knowl.-Based Syst.* **284**, 111253 (2024).
8. Shi, F., Li, D., Wang, X., Li, B. & Wu, X. Tgformer: A graph transformer framework for knowledge graph embedding. *IEEE Trans. Knowl. Data Eng.* (2024).
9. Veličković, P. *et al.* Graph attention networks. In *International Conference on Learning Representations (ICLR)* (2018).
10. Dornaika, F., Bi, J. & Zhang, C. A unified deep semi-supervised graph learning scheme based on nodes re-weighting and manifold regularization. *Neural Netw.* **158**, 188–196 (2023).
11. Zhang, K., Zhu, Y., Wang, J. & Zhang, J. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations* (2020).
12. Xu, K. *et al.* Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, 5453–5462 (PMLR, 2018).
13. Li, G., Muller, M., Thabet, A. & Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9267–9276 (2019).
14. Liu, M., Gao, H. & Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 338–348 (2020).
15. Vaswani, A. *et al.* Attention is all you need. *Adv. Neural Inf. Process. Syst. (NeurIPS)***30** (2017).
16. Dosovitskiy, A. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)* (2021).
17. Chen, D., O'Bray, L. & Borgwardt, K. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, 3469–3489 (PMLR, 2022).
18. Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V. & Tossou, P. Rethinking graph transformers with spectral attention. *Adv. Neural. Inf. Process. Syst.* **34**, 21618–21629 (2021).
19. Ying, C. et al. Do transformers really perform badly for graph representation?. *Adv. Neural. Inf. Process. Syst.* **34**, 28877–28888 (2021).
20. Wu, Q. et al. Sgformer: Simplifying and empowering transformers for large-graph representations. *Adv. Neural Inf. Process. Syst. (NeurIPS)* **36**, 64753–64773 (2023).
21. Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y. & Bresson, X. Graph neural networks with learnable structural and positional representations. *arXiv preprint* arXiv:2110.07875 (2021).
22. Li, P., Wang, Y., Wang, H. & Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Adv. Neural. Inf. Process. Syst.* **33**, 4465–4478 (2020).
23. Rampášek, L. et al. Recipe for a general, powerful, scalable graph transformer. *Adv. Neural. Inf. Process. Syst.* **35**, 14501–14515 (2022).
24. Chen, J., Gao, K., Li, G. & He, K. Nagphormer: A tokenized graph transformer for node classification in large graphs. *arXiv preprint* arXiv:2206.04910 (2022).

25. Bo, D., Shi, C., Wang, L. & Liao, R. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint* arXiv:2303.01028 (2023).
26. Bouritsas, G., Frasca, F., Zafeiriou, S. & Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 657–668 (2022).
27. Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J. & Sinop, A. K. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 31613–31632 (PMLR, 2023).
28. Kong, K. *et al.* Goat: A global transformer on large-scale graphs. In *International Conference on Machine Learning*, 17375–17390 (PMLR, 2023).
29. Wu, Q., Zhao, W., Li, Z., Wipf, D. P. & Yan, J. Nodeformer: A scalable graph structure learning transformer for node classification. *Adv. Neural. Inf. Process. Syst.* **35**, 27387–27401 (2022).
30. Wu, Q. *et al.* Difformer: Scalable (graph) transformers induced by energy constrained diffusion. *arXiv preprint* arXiv:2301.09474 (2023).
31. Deng, C., Yue, Z. & Zhang, Z. Polynormer: Polynomial-expressive graph transformer in linear time. *arXiv preprint* arXiv:2403.01232 (2024).
32. Han, D., Pan, X., Han, Y., Song, S. & Huang, G. Flatten transformer: Vision transformer using focused linear attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5961–5971 (2023).
33. Yang, M. *et al.* Hypformer: Exploring efficient transformer fully in hyperbolic space. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3770–3781 (2024).
34. Luo, Y., Shi, L. & Wu, X.-M. Unlocking the potential of classic gnns for graph-level tasks: Simple architectures meet excellence. *arXiv preprint* arXiv:2502.09263 (2025).
35. Gao, X., Hu, W. & Guo, Z. Exploring structure-adaptive graph learning for robust semi-supervised classification. In *2020 IEEE International Conference on Multimedia and Expo (icme)*, 1–6 (IEEE, 2020).
36. Hamilton, W. L., Ying, Z. & Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1024–1034 (2017).
37. Pei, H., Wei, B., Chang, K. C.-C., Lei, Y. & Yang, B. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint* arXiv:2002.05287 (2020).
38. Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A. & Prokhorenkova, L. A critical look at the evaluation of gnns under heterophily: Are we really making progress? *arXiv preprint* arXiv:2302.11640 (2023).
39. Shchur, O., Mumme, M., Bojchevski, A. & Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint* arXiv:1811.05868 (2018).
40. Rozemberczki, B., Allen, C. & Sarkar, R. Multi-scale attributed node embedding. *J. Complex Netw.* **9**, cnab014 (2021).
41. Zhu, J. et al. Beyond homophily in graph neural networks: Current limitations and effective designs. *Adv. Neural. Inf. Process. Syst.* **33**, 7793–7804 (2020).
42. Paszke, A. *et al.* Automatic differentiation in PyTorch. In *31st Conference on Neural Information Processing Systems (NeurIPS)-Autodiff Workshop* (2017).
43. Fey, M. & Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019). ArXiv preprint arXiv:1903.02428.
44. Gasteiger, J., Bojchevski, A. & Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)* (2019).
45. Yuan, C. *et al.* A survey of graph transformers: Architectures, theories and applications. *arXiv preprint* arXiv:2502.16533 (2025).
46. Pinasthika, K., Laksono, B. S. P., Irsal, R. B. P., Shabiyya, S. & Yudistira, N. Sparseswin: Swin transformer with sparse transformer block. *Neurocomputing* **580**, 127433 (2024).

Acknowledgements

This work was supported by the China University Student Innovation Training Program (202410240141X) and the Philosophy and Social Sciences Research Planning Program (23GLD059).

Author contributions

Y.Z.: Investigation, software, methodology, funding acquisition; X.L.: Investigation, formal analysis, software, methodology; Y.Q.X.: Investigation, methodology, funding acquisition; X.T.X.: Investigation, supervision; Z.W.: Investigation, supervision.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Y.X.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.