Leveraging Contrastive Learning for Enhanced Node Representations in Tokenized Graph Transformers

Jinsong Chen[1,2,3], **Hanpeng Liu**[1,3], **John E. Hopcroft**[3,4], **Kun He**[1,3*]
[1]School of Computer Science and Technology, Huazhong University of Science and Technology
[2]Institute of Artificial Intelligence, Huazhong University of Science and Technology
[3]Hopcroft Center on Computing Science, Huazhong University of Science and Technology
[4]Department of Computer Science, Cornell University
{chenjinsong,hanpengliu}@hust.edu.cn,
jeh@cs.cornell.edu, brooklet60@hust.edu.cn

Abstract

While tokenized graph Transformers have demonstrated strong performance in node classification tasks, their reliance on a limited subset of nodes with high similarity scores for constructing token sequences overlooks valuable information from other nodes, hindering their ability to fully harness graph information for learning optimal node representations. To address this limitation, we propose a novel graph Transformer called GCFormer. Unlike previous approaches, GCFormer develops a hybrid token generator to create two types of token sequences, positive and negative, to capture diverse graph information. And a tailored Transformer-based backbone is adopted to learn meaningful node representations from these generated token sequences. Additionally, GCFormer introduces contrastive learning to extract valuable information from both positive and negative token sequences, enhancing the quality of learned node representations. Extensive experimental results across various datasets, including homophily and heterophily graphs, demonstrate the superiority of GCFormer in node classification, when compared to representative graph neural networks (GNNs) and graph Transformers.

1 Introduction

Node classification, a crucial machine learning task in graph data mining, has garnered significant attention recently due to its wide applicability in diverse areas such as social network analysis [24, 35]. Among numerous techniques developed for this task, graph neural networks (GNNs) stand out as the leading architecture due to their exceptional ability to model graph structural data.

Built on the message-passing mechanism [14], GNNs [19, 8, 9, 33, 34] efficiently integrate node and graph topology features to learn informative node representations, effectively preserving both attribute and structural information. However, as research on GNNs progresses, inherent limitations of the message-passing framework, such as over-smoothing [5] and over-squashing [1], have emerged. These limitations hinder GNNs' ability to capture long-range dependencies in graphs, ultimately constraining their potential for node classification.

Recently, the emerging graph Transformer has attracted great attention in the field of graph representation learning. The crux of this approach is to leverage the Transformer architecture to learn node representations from the input graph. Benefiting from the self-attention mechanism in Transformer, graph Transformers [45, 17, 6, 7, 50] can effectively capture the long-range dependencies in graphs. Serving as a new deep learning-based technique for graphs, graph Transformers have showcased

*Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

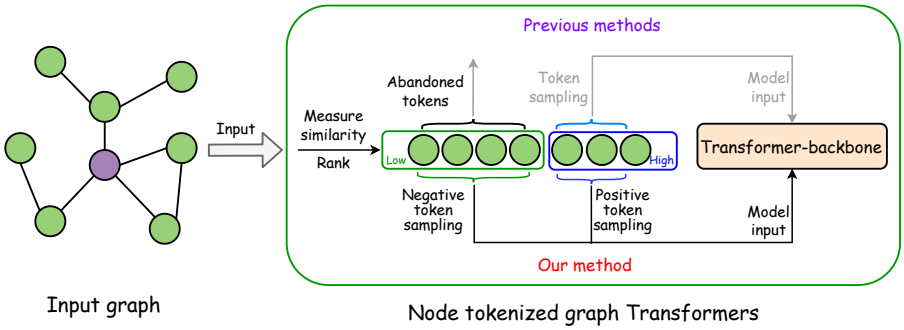


Figure 1: A toy example to illustrate the difference of the token generator between the token generator in our method and that used in the previous node tokenized graph Transformers. Previous methods only sample nodes with high similarity to construct token sequences. In contrast, our method introduces both positive and negative token sampling to preserve information carried by diverse nodes in the graph.

remarkable performance in the node classification task. In this study, We roughly divide the existing graph Transformers designed for node classification into two categories according to the model architecture: GNN-based graph Transformers and tokenized graph Transformers.

GNN-based graph Transformers [30, 42, 41, 23, 27] utilize a hybrid framework that merges Transformer layers with GNN-style modules to learn node representations. However, this approach may constrain the modeling capacity of the Transformer architecture due to the deeply coupled design of the Transformer and GNN layers. A recent study [44] also theoretically proves that directly applying Transformer to calculate the attention scores of all node pairs could cause the over-globalizing problem, which causes the model to overly rely on global information, negatively affecting the model's performance.

In contrast, tokenized graph Transformers [52, 50, 7, 13] initially generate token sequences for each node and only calculate attention scores between tokens within the token sequence, naturally avoiding the over-globalizing issue. These token sequences are then processed by a Transformer-based backbone to learn node representations. This mechanism allows the Transformer to flexibly extract informative node representations based on the input token sequences, demonstrating impressive performance in node classification. Note that, tokenized graph Transformers focus on building token sequences for each target node as model inputs, which is different from TokenGT [18] that transforms all elements in graphs as tokens.

Token generation is a crucial step in tokenized graph Transformers, where node [50] and neighborhood [7] elements form the core of the token sequences. While neighborhood tokens primarily preserve local topology features [13], node tokens can capture a broader range of graph information, including long-range dependencies and intrinsic graph properties (*e.g.*, homophily and heterophily). These advantages allow graph Transformers built on node-oriented token sequences [52, 50, 13] to learn more informative node representations, compared to those based on neighborhood-oriented token sequences.

In this study, we observe that the techniques employed by existing tokenized graph Transformers for generating node-orient token sequences could be summarized as a two-step method. First, they estimate the node similarity matrix according to node information across various feature spaces, such as topology features [52] and attribute features [50, 13]. They then sample a fixed number of nodes with high similarity scores from the generated similarity matrix to construct the input token sequence for a target node. As depicted in Figure 1, only a small subset of nodes is considered, while other nodes are excluded during the training stage.

Compared to sampled nodes which capture the commonality with the target node, these abandoned nodes preserve the disparity, which is also valuable for learning distinguishable node representations. A previous study [3] has proved that leveraging the information from dissimilar nodes aids the

learning of node representations. Nevertheless, existing tokenized graph Transformers can not comprehensively utilize both similar and dissimilar nodes to learn the representation of the target node, inevitably limiting the model performance for node classification. Hence, a natural question arises: *How should we design a new graph Transformer to comprehensively and effectively leverage diverse nodes in graphs to learn distinguishable node representations?*

To answer this question, we propose a new method called Graph Contrastive Transformer (GCFormer). Unlike previous graph Transformers, GCFormer first introduces a novel token sequence generator that produces both positive and negative token sequences for each node in different feature spaces. In this way, various graph information carried by node tokens can be carefully preserved in different types of token sequences. Then, GCFormer develops a new Transformer-based backbone tailored for effectively learning node representations from the generated positive and negative token sequences. Finally, GCFormer leverages the contrastive learning to comprehensively utilize the tokens in both positive and negative sequences to further enhance the quality of learned node representations.

The main contributions of this paper are summarized as follows:

- We develop a new token sequence generator that can generate different types of token sequences in terms of positive and negative node tokens for each target node to preserve various graph information.
- We propose a new graph Transformer GCFormer that formulates a Transformer-based backbone and leverages the contrastive learning to comprehensively learn node representations from positive and negative token sequences.
- We conduct extensive experiments on both homophily and heterophily graphs to validate the effectiveness of the proposed method. Experimental results demonstrate the superiority of GCFormer in node classification compared to representative GNNs and graph Transformers.

2 Related Work

In this section, we first introduce recent studies of graph Transformers for node classification. We then briefly review studies about contrastive learning on graphs.

2.1 Graph Transformer

We categorize existing graph Transformers for node classification into GNN-based methods and tokenized methods. The former [30, 41, 42, 23, 27] combines the Transformer layers with GNN-style modules to learn node representations. GraphGPS [41], one of the representative approaches, incorporates various linear Transformers, such as Reformer [20] and BigBird [47], and GNN layers [19] into a unified framework for graph representation learning. However, these approaches require performing the attention calculation on all node pairs, which can lead to what is known as the over-globalizing problem. A recent study [44] provides both empirical evidence and theoretical analysis to show that calculating attention scores for all nodes can cause the model to overly rely on global information, which can negatively affect the model's performance in node classification tasks.

In contrast, tokenized methods purely depend on the Transformer architecture. The key idea is to generate token sequences for each node from the input graph, which are then fed to Transformer to learn node representations. Node-based [52, 50, 13] and neighborhood-based [7, 11, 13] token generators have been developed to generate various token sequences for nodes. Node-based token generators first calculate the similarity of nodes according to node features such as attribute features [50], then sample nodes with high similarity scores as tokens of the input sequence. While neighborhood-based token generators [7] aggregate the features of multi-hop neighborhoods and further transform them into tokens to construct the token sequence. Compared to neighborhood-based tokens, node-based tokens can express more complex graph information, such as long-range dependencies, which are more suitable for learning informative node representations.

Different from previous node token-based graph Transformers that only consider nodes with high similarity, our proposed GCFormer generates both positive and negative token sequences from all nodes in the graph. Various graph information carried by diverse nodes in two types of token sequences enables GCFormer to learn more distinguishable node representations, leading to superior performance.

2.2 Contrastive Learning on Graphs

Graph contrastive learning (GCL) [36, 54, 46, 31, 49] aims to introduce the contrastive learning mechanism into GNNs to learn informative representations of graphs. Most of GCL approaches share a similar framework that first performs graph augmentation techniques to generate various features of different graph views and then applies the contrastive loss on these generated views to learn graph representations [31]. Recent studies [32, 53, 48, 51] attempt to introduce contrastive learning into graph Transformers. However, these methods require the entire graph as the model input [51, 48] or need to combine GNN-based modules with tailored graph augmentation strategies [32, 53], which are hard to directly apply on tokenized graph Transformers in the node classification task.

Our proposed GCFormer develops a new token generator to generate both positive and negative token sequences for each node without any data augmentations. With the dedicated Transformer-based backbone, GCFormer can effectively leverage the contrastive learning to comprehensively learn informative node representations from two types of token sequences.

3 Preliminaries

3.1 Node Classification

Consider an attributed graph $\mathcal{G} = (V, E)$, where V and E are the node and edge sets, respectively. We have the corresponding adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where n is the number of nodes. For arbitrary two nodes v_i and v_j, $\mathbf{A}_{ij} = 1$ only if $e_{ij} \in E$. The diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$ is represented as $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$. The normalized version of the adjacency matrix with self-loops is represented as $\hat{\mathbf{A}} = (\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}$, where $\mathbf{I}$ denotes the identity matrix. Nodes in $\mathcal{G}$ are associated with attribute feature vectors, assembled into an attribute feature matrix denoted as $\mathbf{X}^a \in \mathbb{R}^{n\times d}$ where d is the dimension of the feature vector. The node label matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$, where c is the label count, consists of rows that are one-hot vectors encoding the label of each node. Each row in $\mathbf{Y}$ is a one-hot vector representing the label information of the corresponding node. Given a subset of nodes with known labels V_l, the objective of node classification is to infer the labels for the remaining nodes in the set $V - V_l$.

3.2 Transformer

Transformer stands as a notable model in deep learning, built upon the Encoder-Decoder architecture. This brief overview focuses on the Transformer layer, a pivotal component of the model. Each Transformer layer is composed of two essential parts: Multi-Head Self-Attention (MSA) and Feed-Forward Networks (FFN).

MSA harnesses multiple attention heads, employing the self-attention mechanism to refine the representations of input entities. Given the input feature matrix $\mathbf{H} \in \mathbb{R}^{n\times d_i n}$, the calculation of the i-th attention head is as follows:

$$\mathrm{head}_i(\mathbf{H}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d_k}})\mathbf{V}, \tag{1}$$

where $\mathbf{Q} = \mathbf{H}\mathbf{W}_{\mathbf{Q}}$, $\mathbf{K} = \mathbf{H}\mathbf{W}_{\mathbf{K}}$ and $\mathbf{V} = \mathbf{H}\mathbf{W}_{\mathbf{V}}$. $\mathbf{W}_{\mathbf{Q}} \in \mathbb{R}^{d_i n\times d_k}$, $\mathbf{W}_{\mathbf{K}} \in \mathbb{R}^{d_i n\times d_k}$ and $\mathbf{W}_{\mathbf{V}} \in \mathbb{R}^{d_i n\times d_v}$ are learnable parameter matrices. The output of MSA with m attention heads is calculated as:

$$\mathbf{H}' = (\mathrm{head}_1||\mathrm{head}_2||\cdots||\mathrm{head}_m)\mathbf{W}_{\mathbf{O}}, \tag{2}$$

where $||$ denotes the vector concatenation operation and $\mathbf{W}_{\mathbf{O}}$ is the learnable matrix.

FFN, comprised of two linear layers enveloping a nonlinear activation function, is defined as:

$$\mathbf{H}' = \mathrm{Linear}(\sigma(\mathrm{Linear}(\mathbf{H}))), \tag{3}$$

where $\mathrm{Linear}(\cdot)$ indicates a linear layer, and $\sigma(\cdot)$ symbolizes the nonlinear activation function.

4 Method

In this section, we detail our proposed GCFormer. First, we introduce the hybrid token generator, which produces both positive and negative token sequences for each node. Then, we introduce the

tailored Transformer-based backbone for extracting node representations from the generated token sequences. Finally, we introduce how to integrate contrastive learning into GCFormer to enhanced node representations.

4.1 Hybrid Token Generator

The proposed hybrid token generator contains two steps: similarity estimating and node sampling. The critical operation of similarity estimating is to calculate the similarity score matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ of all node pairs. Obviously, different node features lead to different score matrices, describing node pairs' relations in different feature spaces. To preserve the complex relations of nodes in the graph, besides the attribute-aware feature matrix $\mathbf{X}^a$, we construct the topology-aware feature matrix $\mathbf{X}^t$:

$$\mathbf{X}^t = \hat{\mathbf{A}}^k \mathbf{X}^a, \tag{4}$$

where k is the propagation step. $\mathbf{X}^t$ preserves the local topology feature within the k-hop neighborhood for each node, which is the essential information to characterize the node property on the graph [7, 16].

Then, we utilize the cosine similarity to calculate the similarity score $\mathbf{S}^a \in \mathbb{R}^{n \times n}$ and $\mathbf{S}^t \in \mathbb{R}^{n \times n}$ based on the node feature matrices $\mathbf{X}^a$ and $\mathbf{X}^t$, respectively. Given a node pair (v_i, v_j), the similarity scores in the attribute feature space $\mathbf{S}^a_{ij}$ and topology feature space $\mathbf{S}^t_{ij}$ are calculated as follows:

$$\mathbf{S}^a_{ij} = \frac{\mathbf{X}^a_i \cdot \mathbf{X}^{a^{\mathrm{T}}}_j}{|\mathbf{X}^a_i||\mathbf{X}^a_j|}, \quad \mathbf{S}^t_{ij} = \frac{\mathbf{X}^t_i \cdot \mathbf{X}^{t^{\mathrm{T}}}_j}{|\mathbf{X}^t_i||\mathbf{X}^t_j|}. \tag{5}$$

After estimating the similarity scores of all node pairs, GCFormer then conducts a two-stage sampling process involving positive token sampling and negative node sampling to generate the token sequences. Here, we introduce the sampling process based on the attribute similarity matrix $\mathbf{S}^a$ for a simplified description. For a given target node v_i, in the positive token sampling stage, we adopt the top-k strategy to select nodes to construct the positive token sequence:

$$V^{a,p}_i = \{v_j | v_j \in \mathrm{Top}(\mathbf{S}^a_i)\}, \tag{6}$$

where $\mathrm{Top}(\cdot)$ denotes the top-k sampling function and $V^{a,p}_i$ denotes the positive token sequence with length p_k. As for the negative token sampling stage, we have the set of rest nodes for v_i after positive token sampling $V^{a,r}_i = V - V^{a,p}_i$. In this paper, we regard all nodes in $V^{a,r}_i$ as the negative samples since their similarity scores are below the threshold of top-k selection. Then, we apply the sampling function to sample nodes from $V^{a,r}_i$ to construct the negative token sequence for v_i:

$$V^{a,n}_i = \{v_j | v_j \in \mathrm{Sample}(V^{a,r}_i)\}, \tag{7}$$

where $\mathrm{Sample}(\cdot)$ denotes an arbitrary sampling function. Here, we use uniform sampling for computing efficiency. $V^{a,n}_i$ denotes the negative token sequence with length n_k.

Following the same sampling process, we can obtain positive and negative token sequences $V^{t,p}_i$ and $V^{t,n}_i$ based on the topology similarity matrix $\mathbf{S}^t$. The constructed positive and negative token sequences not only capture node relations in different feature spaces but also comprehensively extract valuable information from all nodes on the graph.

4.2 Transformer-based Backbone

GCFormer formulates a Transformer-based backbone to effectively learn node representations from positive and negative token sequences. For a node v_i, we first combine itself with generated positive and negative token sequences to construct the model input, $\mathbf{H}^{a,i^o} \in \mathbb{R}^{(1+p_k+n_k) \times d} = \{\mathbf{X}_i, \mathbf{X}_p, \mathbf{X}_n | v_p \in V^{a,p}_i, v_n \in V^{a,n}_i\}$ and $\mathbf{H}^{t,i^o} \in \mathbb{R}^{(1+p_k+n_k) \times d} = \{\mathbf{X}^t_i, \mathbf{X}^t_p, \mathbf{X}^t_n | v_p \in V^{t,p}_i, v_n \in V^{t,n}_i\}$. Note that we utilize the generated $\mathbf{X}^t$ to construct the model input of topology-aware token sequences. In this way, the topology features can be carefully preserved in the model input $\mathbf{H}^{t,i^o}$, exhibiting significant differences with previous methods [52, 50, 13] that utilize the attribute features to construct topology-aware token sequences. Following previous studies [8, 7, 13], we leverage projection layers to obtain the initial input:

$$\mathbf{H}^{a,i} = \mathbf{H}^{a,i^o}\mathbf{W}^a, \quad \mathbf{H}^{t,i} = \mathbf{H}^{t,i^o}\mathbf{W}^t, \tag{8}$$

where $\mathbf{W}^a \in \mathbb{R}^{d \times d_0}$ and $\mathbf{W}^t \in \mathbb{R}^{d \times d_0}$ denote the parameter matrices of the projection layers.

Given the model input $\mathbf{H}^{a,i}$ of the node v_i, GCFormer first separates the negative tokens from $\mathbf{H}^{a,i}$, resulting in two parts: $\mathbf{P}^{a,i^{(0)}} \in \mathbb{R}^{(1+p_k)\times d_0}$ and $\mathbf{N}^{a,i^{(0)}} \in \mathbb{R}^{n_k \times d_0}$. Next, GCFormer adds a virtual token with learnable features into $\mathbf{N}^{a,i^{(0)}}$ as the first token to facilitate extracting valuable information from negative tokens. Then, GCFormer adopts standard Transformers layers to learn node representations from $\mathbf{P}^{a,i^{(0)}}$ and $\mathbf{N}^{a,i^{(0)}}$:

$$\mathbf{P}^{a,i^{(l)'}} = \mathrm{MSA}(\mathbf{P}^{a,i^{(l-1)}}) + \mathbf{P}^{a,i^{(l-1)}}, \quad \mathbf{P}^{a,i^{(l)}} = \mathrm{FFN}(\mathbf{P}^{a,i^{(l)'}}) + \mathbf{P}^{a,i^{(l)'}}, \tag{9}$$

$$\mathbf{N}^{a,i^{(l)'}} = \mathrm{MSA}(\mathbf{N}^{a,i^{(l-1)}}) + \mathbf{N}^{a,i^{(l-1)}}, \quad \mathbf{N}^{a,i^{(l)}} = \mathrm{FFN}(\mathbf{N}^{a,i^{(l)'}}) + \mathbf{N}^{a,i^{(l)'}}, \tag{10}$$

where $\mathrm{MSA}(\cdot)$ and $\mathrm{FFN}(\cdot)$ denote the multi-head self-attention and feed-forward networks.

Through several Transformer layers, the corresponding $\mathbf{P}^{a,i} \in \mathbb{R}^{(1+p_k)\times d_{out}}$ and $\mathbf{N}^{a,i} \in \mathbb{R}^{(1+n_k)\times d_{out}}$ contains information extracted from positive and negative token sequences, respectively. To effectively fuse information from different types of token sequences, inspired by signed attention mechanism in previous approaches [3, 10], we develop the following readout function:

$$\mathbf{H}^{a,i} = \mathbf{P}_0^{a,i} - \mathbf{N}_0^{a,i}, \tag{11}$$

where $\mathbf{H}^{a,i} \in \mathbb{R}^{1\times d_{out}}$ denote the node representation of v_i extracted from the attribute-aware token sequence.

The rationale of Equation 11 is that the representations $\mathbf{P}_0^{a,i}$ (the target node) and $\mathbf{N}_0^{a,i}$ (the virtual node) contain the learned information from positive and negative token sequences, respectively. The desired representation of v_i should be far away from the representations of negative tokens in the hidden feature space since there is a high probability that they belong to different labels. While the signed aggregation operation can enforce $\mathbf{H}^{a,i}$ to be dissimilar with the representations of negative tokens according to the previous study [3, 10].

We can also obtain the representation $\mathbf{H}^{t,i} \in \mathbb{R}^{1\times d_{out}}$ extracting from the topology-aware token sequence $\mathbf{H}^{t,i^o}$ via the same operation. Considering the contributions of attribute information and topology information vary on different graphs, we develop a weighted fusion strategy to obtain the final representation $\mathbf{Z}^i$:

$$\mathbf{Z}^i = \alpha \cdot \mathbf{H}^{a,i} + (1-\alpha) \cdot \mathbf{H}^{t,i}, \tag{12}$$

where $\alpha \in [0, 1]$ is a hyper-parameter to determine the contributions of attribute information and topology information to the final representation.

4.3 Integrating Contrastive Learning

Though Equation 11 leverages information of negative tokens to learn node representation, it fails to directly model relations between the target node and its negative tokens. To this end, we introduce the contrastive learning loss [15] to fully utilize negative tokens for enhanced node representations. For a node v_i, the contrastive learning loss is calculated as follows:

$$\mathcal{L}_{cl}(v_i) = -\log \frac{\exp(\mathbf{P}_0^{a,i} \cdot \hat{\mathbf{P}}^{a,i^{\mathrm{T}}}/\tau)}{\sum_{j=1}^{n_k} \exp(\mathbf{P}_0^{a,i} \cdot \mathbf{N}_j^{a,i^{\mathrm{T}}}/\tau)} - \log \frac{\exp(\mathbf{P}_0^{t,i} \cdot \hat{\mathbf{P}}^{t,i^{\mathrm{T}}}/\tau)}{\sum_{j=1}^{n_k} \exp(\mathbf{P}_0^{t,i} \cdot \mathbf{N}_j^{t,i^{\mathrm{T}}}/\tau)}, \tag{13}$$

where $\hat{\mathbf{P}}^{a,i} = \frac{1}{p_k}\sum_{j=1}^{p_k} \mathbf{P}_j^{a,i}$ and $\hat{\mathbf{P}}^{t,i} = \frac{1}{p_k}\sum_{j=1}^{p_k} \mathbf{P}_j^{t,i}$. τ is a temperature hyper-parameter. Equation 13 enforces the representation of the target node to be close to the central representation of all positive tokens and away from all negative samples, which promotes learning distinguishable node representations, beneficial for downstream classification tasks. We further adopt the Cross-entropy loss for node classification:

$$\mathcal{L}_{ce} = -\sum_{i \in V_l} \mathbf{Y}_i \ln \hat{\mathbf{Y}}_i, \hat{\mathbf{Y}}_i = \mathrm{MLP}(\mathbf{Z}^i), \tag{14}$$

where $\mathrm{MLP}(\cdot)$ denotes the Multilayer Perceptron-based classifier. Hence, the overall loss function of GCFormer is as follows:

$$\mathcal{L} = \mathcal{L}_{ce} + \beta \cdot \mathcal{L}_{cl}, \tag{15}$$

where β is the coefficient for the contrastive learning term.

Table 1: Comparison of all models in terms of mean accuracy ± stdev (%). The best results appear in **bold**. The second results appear in underline.

Dataset	Photo	ACM	Comuter	Corafull	BlogCatalog	UAI2010	Flickr	Romanempire
$H(\mathcal{G})$	0.83	0.82	0.78	0.57	0.40	0.36	0.24	0.05
APPNP	93.00±0.55	93.00±0.55	91.31±0.29	63.37±0.04	94.77±0.19	76.41±0.47	84.66±0.31	52.96±0.35
SGC	93.74±0.07	93.24±0.49	88.90±0.11	62.77±0.19	72.61±0.07	69.87±0.17	47.48±0.40	34.42±0.77
GPRGNN	94.57±0.44	93.42±0.20	90.15±0.34	69.08±0.11	94.36±0.29	76.94±0.64	85.91±0.51	67.06±0.27
FAGCN	94.06±0.03	93.37±0.24	83.17±1.81	56.61±2.94	79.92±4.39	72.17±1.57	82.03±0.40	48.21±3.15
ACM-GCN	94.56±0.21	93.04±1.28	85.19±2.26	65.11±1.98	94.53±0.53	76.87±1.42	83.85±0.73	63.35±1.80
SGFormer	92.93±0.12	93.79±0.34	81.86±3.82	64.62±1.20	94.33±0.19	57.98±3.95	61.05±0.68	41.31±0.51
ANS-GT	94.88±0.23	93.92±0.21	89.58±0.28	67.94±0.21	91.93±0.31	74.16±0.71	85.94±0.25	73.95±0.32
Specformer	95.22±0.13	93.63±1.94	85.47±1.44	69.18±0.24	94.21±0.23	73.06±0.77	86.55±0.40	63.69±0.61
VCR-Graphormer	95.13±0.24	93.24±0.31	90.14±0.43	68.96±0.28	93.92±0.37	75.78±0.69	86.23±0.74	74.76±0.83
GraphGPS	93.79±0.32	93.31±0.26	89.21±0.28	62.08±0.35	94.35±0.52	75.44±0.48	83.61±0.57	68.29±0.92
NAGphormer	95.47±0.29	93.32±0.30	90.79±0.45	69.34±0.52	94.42±0.63	76.36±1.12	86.85±0.85	74.94±0.52
GCFormer	**95.65±0.41**	**94.32±0.47**	**92.09±0.21**	**69.53±0.35**	**96.03±0.44**	**77.57±0.86**	**87.90±0.45**	**75.38±0.68**

5 Experiments

5.1 Experimental Setup

We briefly introduce the experimental setup including datasets, baselines and parameter settings. Detailed information is provided in Appendix A due to the space limitation.

Datasets. We adopt eight widely used datasets, including four homophily and four heterophily graphs: Photo [7], ACM [37], Computer [7], Corafull [4], BlogCatalog [28], UAI2010 [38], Flickr [28] and Romnnempire [29]. The edge homophily ratio [22] $H(\mathcal{G}) \in [0, 1]$ is adopted to evaluate the graph's homophily level. $H(\mathcal{G}) \to 1$ means strong homophily, while $H(\mathcal{G}) \to 0$ means strong heterophily. Statistics of datasets are summarized in Appendix A. Following the settings of previous studies [41, 42], we randomly choose 50% of each label as the training set, 25% as the validation set, and the rest as the test set.

Baselines. We adopt eleven powerful approaches on node classification as baselines, including GNNs and graph Transformers: APPNP [21], SGC [40], GPRGNN [12], FAGCN [3], ACM-GCN [26], SGFormer [42], ANS-GT [50], Specformer [2], VCR-Graphormer [13], GraphGPS [30] [1] and NAGphormer [7]. The first five are representative GNNs and others are recent graph Transformers.

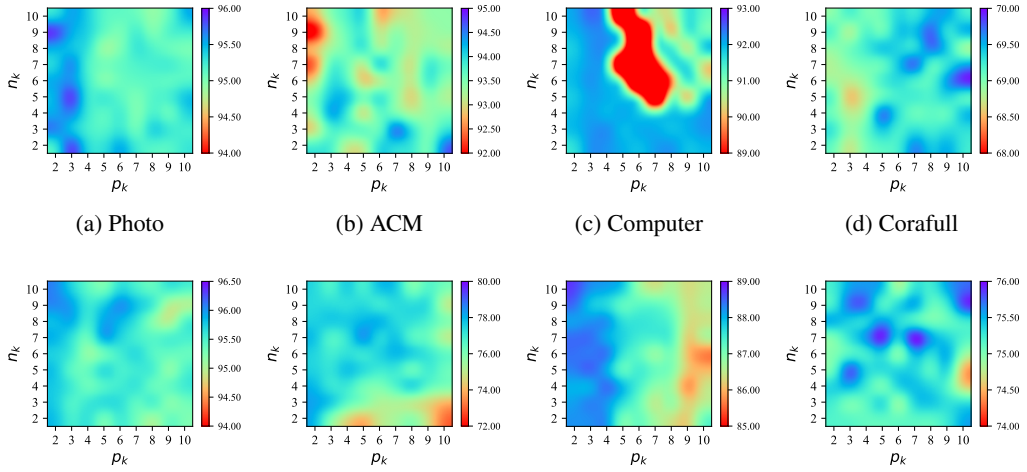
Parameter settings. For baselines, referring to recommended settings in their official implementations, we perform hyper-parameter tuning for all models. For GCFormer, we try the dimension of hidden representations in $\{128, 256, 512\}$, number of layers in $\{1, 2, 3\}$, learning rate in $\{0.01, 0.005, 0.001\}$, dropout rate in $\{0.1, 0.3, 0.5\}$. The training process is early stopped within 50 epochs and parameters are optimized using AdamW [25].

5.2 Performance Comparison

To evaluate the model performance in node classification, we run each model with different random seeds on datasets and report the average value of accuracy and the corresponding standard deviation.

Table 1 reports the results. We can observe that GCFormer achieves the best performance on all datasets, indicating the superiority of GCFormer on the node classification task. Specifically, GCFormer beats recent tokenized graph Transformers on all datasets, especially ANS-GT which is the representative method of node token sequence-based graph Transformers. This is because that GCFormer generates both positive and negative token sequences for each node, which preserve both commonality and disparity between node features. In addition, the tailored Transformer-based backbone and contrastive learning enable GCFormer to comprehensively learn distinguishable node representations from different types of token sequences, further enhancing the performance in the node classification task. Moreover, we also find graph Transformer-based baselines achieve higher accuracy values than GNN-based baselines on over half of datasets. This is because graph Transformers can

[1] Due to the various implementations of GraphGPS, here we only report the best combination. Detailed results of all combinations can refer to Appendix C.



(a) Photo (b) ACM (c) Computer (d) Corafull

(e) BlogCatalog (f) UAI2010 (g) Flickr (h) Romanempire

Figure 2: Performance of GCFormer with different sampling sizes on all datasets.

effectively preserve various graph information, such as local topology features [7] and long-range dependencies [50], revealing the potential of graph Transformers in graph mining tasks.

5.3 Parameter Sensitivity Analysis

The token sampling size and the aggregation weight α in Equation 12 are key parameters in GCFormer. The former determines the model input and the latter controls the learning of final node representations from different feature spaces. Here, we conduct experiments to analyze the influence of these parameters on model performance.

Analysis of token sampling sizes. To analyze the influence of different sampling sizes on model performance, we vary p_k and n_k in $\{2, 3, \ldots, 10\}$ where p_k and n_k are the lengths of positive token sequences and negative token sequences. Figure 2 shows the changes in model performance across all datasets. Generally speaking, a large sampling size of negative tokens can lead to competitive model performance. For instance, n_k over six can enable GCFormer to achieve high accuracy on almost all datasets except ACM. This is because a large value of n_k is more conducive to preserving the disparity between target nodes and negative node tokens, leading to more distinguishable node representations. This phenomenon also indicates that introducing negative tokens can effectively enhance the performance of tokenized graph Transformers in node classification. In addition, GCFormer is relatively sensitive to n_p. Half of the datasets, such as Photo and BlogCatalog, require a small value of n_p to achieve competitive performance. While other datasets prefer large n_p. This is because different graphs can exhibit diverse features, including node attribute features and graph topology features, which affect the sampling of positive tokens. And a large n_p could introduce irrelevant nodes into positive token sequences when the features of graphs are too complex to sample relevant nodes, further hurting the performance of GCFormer.

Analysis of α. To explore the influence of α on model performance, we vary α in $\{0, 0.1, \ldots, 1\}$ and observe the changes of model performance. $\alpha = 0$ or $\alpha = 1$ mean that we abandon the information from attribute-aware token sequences or topology-aware token sequences when generating the final node representations. Results across all datasets are shown in Figure 3. We can find that the optimal α falls in $(0, 1)$ for all datasets. This observation indicates that comprehensively considering the features of attribute and topology information is essential to learn distinguishable node representations. Another observation is that the model performances on graphs extracted from the same domain exhibit similar changing trends. For instance, GCFormer achieves the best performance when $\alpha = 0.5$ on BlogCatalog and Flickr, which are extracted from the social platforms. This may be because graphs extracted from the same domains exhibit similar graph topology features and node attribute features.

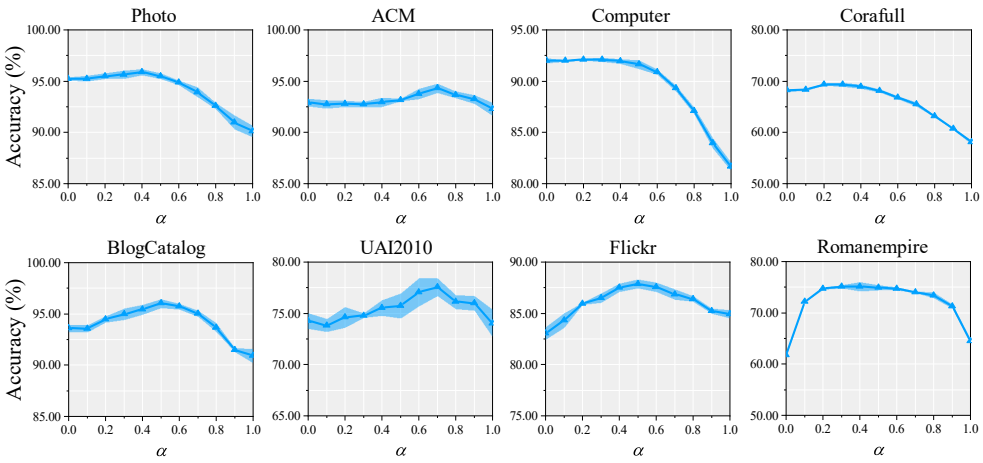


Figure 3: Performance of GCFormer with different α on all datasets.

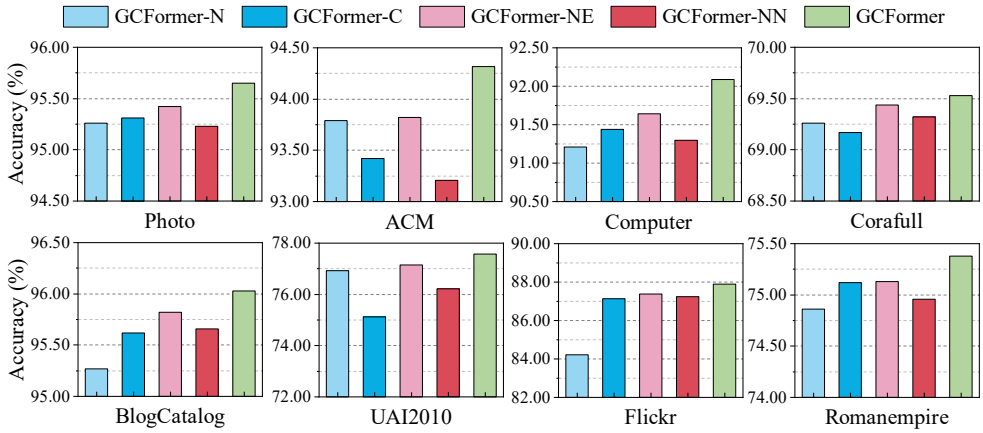


Figure 4: Performances of GCFormer and its variants.

5.4 Ablation Study

Generating negative token sequences and integrating contrastive learning loss are two key designs of GCFormer. To comprehensively validate the effectiveness of these designs, we propose four variants of GCFormer termed GCFormer-N, GCFormer-C, GCFormer-NE and GCFormer-NN. GCFormer-N removes the negative token sequences and the contrastive learning loss. GCFormer-C only removes the contrastive learning loss. GCFormer-NE retains the use of Transformer layers for learning negative token representations but only employs these representations in the contrastive learning loss (ignoring them in Equation 11). GCFormer-NN, conversely, directly uses the representations of negative tokens for contrastive learning without passing them through Transformer layers. We then run four variants on all datasets and the results are shown in Figure 4. We can observe that GCFormer beats four variants on all datasets, indicating the effectiveness of our key designs in enhancing the model performance. In addition, we can also find that GCFormer-C beats GCFormer-N on over half datasets. This phenomenon demonstrates that introducing negative token sequences can effectively improve the model performance. Nevertheless, the performances of GCFormer-C behind GCFormer-N on three citation networks. This situation reveals that different types of graphs can affect the gains of introducing negative tokens. In addition, The results demonstrate that GCFormer-NE outperforms GCFormer-NN on all datasets, indicating that leveraging the Transformer to learn representations of negative tokens can effectively enhance the benefits of introducing contrastive learning. Furthermore, GCFormer surpasses GCFormer-NE, suggesting that comprehensively utilizing the representations

of negative tokens through the signed aggregation operation and contrastive learning can further augment the model's ability to learn more discriminative node representations.

6 Conclusion

In this paper, we propose GCFormer, a novel graph Transformer for node classification. GCFormer establishes a new framework of tokenized graph Transformers to effectively learn node representations. Specifically, GCFormer develops a new hybrid token generator that generates both positive and negative token sequences. Compared to previous methods that only sample nodes with high similarity as tokens, GCFormer considers diverse nodes with high and low similarity. This merit enables GCFormer to preserve both commonality and disparity between node representations. By formulating a Transformer-based backbone and integrating contrastive learning, GCFormer can comprehensively learn distinguishable node representations from different types of token sequences. Extensive experimental results on diverse graphs extracted from different domains showcase the superiority of GCFormer in node classification compared to representative GNNs and graph Transformers.

The main limitation of GCFormer is the unified sampling strategy for different types of graphs. Experimental results show that the performance of GCFormer is sensitive to the sampling size on different graphs. The phenomenon implies that an adaptive sampling strategy is required to improve the performance and stability of GCFormer on diverse graphs.

Acknowledgments

This work is supported by National Natural Science Foundation (62076105,U22B2017).

References

[1] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *Proceedings of the International Conference on Learning Representations*.

[2] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. 2023. Specformer: Spectral Graph Neural Networks Meet Transformers. In *Proceedings of the International Conference on Learning Representations*.

[3] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3950–3957.

[4] Aleksandar Bojchevski and Stephan Günnemann. 2018. Deep Gaussian Embedding of Graphs: Unsupervised Inductive Learning via Ranking. In *Proceedings of the International Conference on Learning Representations*.

[5] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3438–3445.

[6] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. 2022. Structure-aware Transformer for Graph Representation Learning. In *Proceedings of the International Conference on Machine Learning*, Vol. 162. 3469–3489.

[7] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *Proceedings of the International Conference on Learning Representations*.

[8] Jinsong Chen, Boyu Li, and Kun He. 2024. Neighborhood Convolutional Graph Neural Network. *Knowledge-Based Systems* (2024), 111861.

[9] Jinsong Chen, Boyu Li, Qiuting He, and Kun He. 2024. PAMT: A Novel Propagation-Based Approach via Adaptive Similarity Mask for Node Classification. *IEEE Transactions on Computational Social Systems* (2024).

[10] Jinsong Chen, Gaichao Li, John E. Hopcroft, and Kun He. 2023. SignGT: Signed Attention-based Graph Transformer for Graph Representation Learning. *CoRR* abs/2310.11025 (2023).

[11] Jinsong Chen, Chang Liu, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. Tokenized Graph Transformer with Neighborhood Augmentation for Node Classification in Large Graphs. *CoRR* abs/2305.12677 (2023).

[12] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *Proceedings of the International Conference on Learning Representations*.

[13] Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. 2024. VCR-Graphormer: A Mini-batch Graph Transformer via Virtual Connections. *CoRR* abs/2403.16030 (2024).

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *Proceedings of the International Conference on Machine Learning*. 1263–1272.

[15] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum Contrast for Unsupervised Visual Representation Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9726–9735.

[16] Qiuting He, Jinsong Chen, Hao Xu, and Kun He. 2022. Structural Robust Label Propagation on Homogeneous Graphs. In *Proceedings of the IEEE International Conference on Data Mining*. 181–190.

[17] Paras Jain, Zhanghao Wu, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. 2021. Representing Long-Range Context for Graph Neural Networks with Global Attention. In *Proceedings of the Annual Conference on Neural Information Processing Systems*. 13266–13279.

[18] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. 2022. Pure Transformers are Powerful Graph Learners. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[19] Thomas N Kipf and Max Welling. 2017. Semi-supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*.

[20] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *Proceedings of the International Conference on Learning Representations*.

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks Meet Personalized PageRank. In *Proceedings of the International Conference on Learning Representations*.

[22] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. 2022. Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In *Proceedings of the International Conference on Machine Learning*, Vol. 162. 13242–13256.

[23] Chuang Liu, Yibing Zhan, Xueqi Ma, Liang Ding, Dapeng Tao, Jia Wu, and Wenbin Hu. 2023. Gapformer: Graph transformer with graph pooling for node classification. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*. 2196–2205.

[24] Zewen Liu, Guancheng Wan, B Aditya Prakash, Max SY Lau, and Wei Jin. 2024. A review of graph neural networks in epidemic modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6577–6587.

[25] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *Proceedings of the International Conference on Learning Representations*.

[26] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting Heterophily For Graph Neural Networks. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[27] Xiaojun Ma, Qin Chen, Yi Wu, Guojie Song, Liang Wang, and Bo Zheng. 2023. Rethinking Structural Encodings: Adaptive Graph Transformer for Node Classification Task. In *Proceedings of the ACM Web Conference*. 533–544.

[28] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Co-Embedding Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 393–401.

[29] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: Are we really making progress?. In *Proceedings of the Eleventh International Conference on Learning Representations*.

[30] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[31] Xiao Shen, Dewang Sun, Shirui Pan, Xi Zhou, and Laurence T. Yang. 2023. Neighbor Contrastive Learning on Learnable Graph Augmentation. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*. 9782–9791.

[32] Yundong Sun, Dongjie Zhu, Yansong Wang, and Zhaoshuo Tian. 2024. GTC: GNN-Transformer Co-contrastive Learning for Self-supervised Heterogeneous Graph Representation. *CoRR* abs/2403.15520 (2024).

[33] Zihan Tan, Guancheng Wan, Wenke Huang, and Mang Ye. 2024. FedSSP: Federated Graph Learning with Spectral Knowledge and Personalized Preference. In *Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[34] Guancheng Wan, Wenke Huang, and Mang Ye. 2024. Federated Graph Learning under Domain Shift with Generalizable Prototypes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 15429–15437.

[35] Guancheng Wan, Zewen Liu, Max S.Y. Lau, B. Aditya Prakash, and Wei Jin. 2024. Epidemiology-Aware Neural ODE with Continuous Disease Transmission Graph. *arXiv preprint arXiv:2410.00049* (2024).

[36] Guancheng Wan, Yijun Tian, Wenke Huang, Nitesh V Chawla, and Mang Ye. 2024. S3GCL: Spectral, Swift, Spatial Graph Contrastive Learning. In *Forty-first International Conference on Machine Learning*.

[37] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *Proceedings of the World Wide Web Conference*. 2022–2032.

[38] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1243–1253.

[39] Yanling Wang, Jing Zhang, Haoyang Li, Yuxiao Dong, Hongzhi Yin, Cuiping Li, and Hong Chen. 2022. ClusterSCL: Cluster-Aware Supervised Contrastive Learning on Graphs. In *Proceedings of the ACM Web Conference 2022*. 1611–1621.

[40] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the International Conference on Machine Learning*. 6861–6871.

[41] Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. 2022. NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, Vol. 35. 27387–27401.

[42] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. 2023. Simplifying and empowering transformers for large-graph representations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[43] Siyue Xie, Da Sun Handason Tam, and Wing Cheong Lau. 2022. CoCoS: Enhancing Semi-supervised Learning on Graphs with Unlabeled Data via Contrastive Context Sharing. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence*. 4272–4280.

[44] Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. 2024. Less is More: on the Over-Globalizing Problem in Graph Transformers. In *Proceedings of the International Conference on Machine Learning*.

[45] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, Vol. 34. 28877–28888.

[46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[47] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

[48] Chun-Yang Zhang, Wu-Peng Fang, Hai-Chun Cai, C. L. Philip Chen, and Yue-Na Lin. 2024. Sparse Graph Transformer With Contrastive Learning. *IEEE Trans. Comput. Soc. Syst.* 11, 1 (2024), 892–904.

[49] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. 2022. COSTA: Covariance-Preserving Feature Augmentation for Graph Contrastive Learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2524–2534.

[50] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. 2022. Hierarchical Graph Transformer with Adaptive Node Sampling. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, Vol. 35. 21171–21183.

[51] Han Zhao, Xu Yang, Kun Wei, Cheng Deng, and Dacheng Tao. 2024. Unsupervised Graph Transformer With Augmentation-Free Contrastive Learning. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[52] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. 2021. Gophormer: Ego-Graph Transformer for Node Classification. *arXiv preprint arXiv:2110.13094* (2021).

[53] Yangfu Zhu, Linmei Hu, Xinkai Ge, Wanrong Peng, and Bin Wu. 2022. Contrastive Graph Transformer Network for Personality Detection. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. 4559–4565.

[54] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. *CoRR* abs/2006.04131 (2020).

A Detailed Experimental Settings

Here, we introduce the detailed information about experimental settings.

A.1 Datasets

In this paper, we adopt eight datasets from diverse domains, including homophily and heterophily graphs. The statistics of datasets are summarized in Table 2.

- **Citation networks.** ACM, Corafull and UAI2010 are constructed from citation networks where nodes represent research papers and edges represent the relations between papers (*e.g.*, having common authors or citation relation).
- **Co-purchase networks.** Photo and Computer are extracted from the Amazon purchase network where nodes represent goods and edges represent that two goods appear in a same shopping list.
- **Social networks.** BlogCatalog and Flickr are generated from social platforms BlogCatalog and Flickr, respectively. Nodes represent users and edges represent social relationships between users.
- **Wikipedia.** Romanempire is extracted from English Wikipedia where nodes represent words in the text and edges represent that two words connected in the dependency tree of the sentence.

ACM, UAI2010, BlogCatalog and Flickr can be downloaded from [1]. Corafull, Photo and Computer can be downloaded from [2]. Romanempire can be downloaded from [3]. In practice, we first apply the principal components analysis (PCA) to reduce the raw features into 256-dimension vectors on Corafull, BlogCatalog, UAI2010 and Flickr since the raw features of these datasets are too sparse which waste computing resources.

Table 2: Statistics on datasets, ranked by the homophily level from high to low.

Dataset	# nodes	# edges	# features	# labels	$H \downarrow$
Photo	7,650	238,163	745	8	0.83
ACM	3,025	1,3128	1,870	3	0.82
Computer	13,752	491,722	767	10	0.78
Corafull	19,793	126,842	8,710	70	0.57
BlogCatalog	5,196	171,743	8,189	6	0.40
UAI2010	3,067	28,311	4,973	19	0.36
Flickr	7,575	239,738	12,047	9	0.24
Romanempire	22,662	32,927	300	18	0.05

A.2 Parameter Configuration

Referring to the official implementations, we perform hyper-parameter tuning of baselines on each dataset. We adopt the grid search strategy to determine the optimal parameters. Specifically, We try learning rate in $\{0.001, 0.005, 0.01\}$, dropout in $\{0.3, 0.5, 0.7\}$, dimension of hidden representations in $\{128, 512\}$. For GCFormer, we try p_k and n_k in $\{3, 5, 7\}$, α in $\{0.1, \ldots, 0.9\}$, β in $\{0.05, 0.1, 0.5, 1\}$. We implement all codes based on Python 3.8, Pytorch 1.11, and CUDA 11.0. All experiments are conducted on a Linux server with one Intel Xeon(R) Sliver 4210, 256G RAM and one RTX TITAN.

[1] https://github.com/zhumeiqiBUPT/AM-GCN

[2] https://github.com/JHL-HUST/NAGphormer

[3] https: //github.com/yandex-research/heterophilous-graphs

Table 3: Comparison of all models in terms of mean accuracy ± stdev (%).

Dataset	Photo	ACM	Comuter	Corafull	BlogCatalog	UAI2010	Flickr	Romanempire
$H(\mathcal{G})$	0.83	0.82	0.78	0.57	0.40	0.36	0.24	0.05
ClusterSCL	93.98±0.43	93.27±0.29	88.74±0.64	62.32±0.29	84.62±0.1.24	74.37±0.58	83.84±0.42	67.37±0.81
CoCoS	93.73±0.12	93.24±0.66	89.66±0.48	64.25±0.38	87.56±0.26	75.89±0.33	83.43±0.59	66.28±0.47
NCLA	94.21±0.36	93.46±0.39	89.52±0.45	62.79±0.34	86.69±0.68	76.28±0.82	84.06±0.54	71.89±0.49
GCFormer	95.65±0.41	94.32±0.47	92.09±0.21	69.53±0.35	96.03±0.44	77.57±0.86	87.90±0.45	75.38±0.68

Table 4: Performance of different GraphGPS's implementations."T" and "P" indicate the original Transformer and Performer. "L", "R" and "D" indicate the Laplacian positional encoding, RWSE structural encoding and degree-based encoding. "OOM" indicates the out-of-memory issue.

Dataset	Photo	ACM	Comuter	Corafull	BlogCatalog	UAI2010	Flickr	Romanempire
$H(\mathcal{G})$	0.83	0.82	0.78	0.57	0.40	0.36	0.24	0.05
GCN+T+L	93.79	93.12	OOM	OOM	84.62	74.37	83.84	OOM
GCN+T+R	93.81	93.26	OOM	OOM	84.62	74.37	83.84	OOM
GCN+T+D	92.95	92.84	OOM	OOM	84.62	74.37	83.84	OOM
GCN+P+L	93.74	93.23	89.21	61.27	94.21	75.44	83.54	68.29
GCN+P+R	93.62	93.31	89.18	62.08	94.35	75.14	82.72	67.52
GCN+P+D	92.38	92.43	88.06	59.86	92.75	70.16	80.88	64.56
GCFormer	95.65	94.32	92.09	69.53	96.03	77.57	87.90	75.38

B Performance Comparison with GSL-based Approaches

Here, we conduct additional experiments to validate the effectiveness of GCFormer on node classification, compared with representative graph contrastive learning-based methods. Specifically, we select three approaches, CluterSCL [39], CoCoS [43] and NCLA [31] for performance comparison. We adopt their official implementations and turn hyper-parameters accordingly on each dataset. The results are shown in Table 3. We can observe that GCFormer outperforms representative GCL-based approaches on all datasets, demonstrating its superiority in node classification.

C Detailed results of GraphGPS

Here, we provide the detailed results of different implementations of GraphGPS. We adopt this resource code[1] for experiments. The results are shown in Table 4. The results demonstrate that GCFormer outperforms GraphGPS on all datasets, highlighting the effectiveness of GCFormer in comparison to representative graph Transformers in the task of node classification.

[1]https://github.com/luis-mueller/probing-graph-transformers

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We have claimed that this paper focuses on developing a new graph Transformer for the node classification task in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed in the conclusion that the limitation of the proposed method is the unified sampling strategy for diverse graphs, making the performance sensitive to the sampling size of node tokens.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: There are no theoretical results in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In the experiment section and appendix, we have provided the detailed experimental settings for results reproducing.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have provided open access to the datasets and detailed experimental settings for reproducing results in the appendix. The code will be released after the paper has been accepted.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the detailed experimental settings in the experiment section and appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Following previous studies, we use the average accuracy and standard deviation to evaluate the model performance in the node classification task.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In the appendix, we have provided detailed information on hardware and software environments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our paper is conducted with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper focuses on foundational research of graph representation learning, which does not involve societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We use the public datasets for experiments in this paper and have cited the original paper that produced the dataset.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper focuses on graph representation learning, which does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.