A Closer Look at Graph Transformers: Cross-Aggregation and Beyond

Jiaming Zhuo[1], **Ziyi Ma**[1], **Yintong Lu**[1], **Yuwei Liu**[1], **Kun Fu**[1], **Di Jin**[2], **Chuan Wang**[3], **Wenning Wu**[4], **Zhen Wang**[4], **Xiaochun Cao**[5], **Liang Yang**[1*]
[1]Hebei Province Key Laboratory of Big Data Calculation,
School of Artificial Intelligence, Hebei University of Technology, Tianjin, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin, China
[3]School of Computer Science and Technology, Beijing JiaoTong University, Beijing, China
[4]School of Artificial Intelligence, OPtics and ElectroNics (iOPEN),
School of Cybersecurity, Northwestern Polytechnical University, Xi'an, China
[5]School of Cyber Science and Technology,
Shenzhen Campus of Sun Yat-sen University, Shenzhen, China
jiaming.zhuo@outlook.com, zyma@hebut.edu.cn,
{202332803037, 202322802030}@stu.hebut.edu.cn, fukun@hebut.edu.cn,
jindi@tju.edu.cn, wangchuan@iie.ac.cn, wuwenning@nwpu.edu.cn,
w-zhen@nwpu.edu.cn, caoxiaochun@mail.sysu.edu.cn, yangliang@vip.qq.com

Abstract

Graph Transformers (GTs), which effectively capture long-range dependencies and structural biases simultaneously, have recently emerged as promising alternatives to traditional Graph Neural Networks (GNNs). Advanced approaches for GTs to leverage topology information involve integrating GNN modules or modulating node attributes using positional encodings. Unfortunately, the underlying mechanism driving their effectiveness remains insufficiently understood. In this paper, we revisit these strategies and uncover a shared underlying mechanism—Cross Aggregation—that effectively captures the interaction between graph topology and node attributes. Building on this insight, we propose the Universal Graph Cross-attention Transformer (UGCFormer), a universal GT framework with linear computational complexity. The idea is to interactively learn the representations of graph topology and node attributes through a linearized Dual Cross-attention (DCA) module. In theory, this module can adaptively capture interactions between these two types of graph information, thereby achieving effective aggregation. To alleviate overfitting arising from the dual-channel design, we introduce a consistency constraint that enforces representational alignment. Extensive evaluations on multiple benchmark datasets demonstrate the effectiveness and efficiency of UGCFormer.

1 Introduction

Node classification, aimed at accurately predicting node categories based on the graph topology and node attributes, is a fundamental task in identifying the properties of individual nodes [12, 19, 14, 30, 11, 10]. As a powerful class of models for fusing topology and attribute information in graphs, Graph Neural Networks (GNNs) have achieved initial successes in this task [27, 15, 58, 51, 5, 50]. In general, they follow the graph-bound Message Passing (MP) paradigm [17]. While this paradigm endows GNNs with the localizing property, it also restricts their ability to capture long-range dependencies [9], resulting in well-known challenges such as over-smoothing [6, 59] and over-squashing [18].

*Corresponding author

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Inspired by the remarkable success of Transformers in NLP [33], Graph Transformers (GTs) have emerged as powerful architectures for node classification tasks. The core component of Transformers is the Self-Attention (SA) module [44], which models full interactions among tokens within a sequence, thereby endowing the Transformers with globalizing properties. The initial success of GTs can be attributed to the strategic integration of discriminative graph topology into Transformer architectures, enabling the simultaneous capture of structural biases and long-range dependencies. To date, two primary strategies have achieved SOTA performance in existing GTs: (1) integrating GNN blocks [29, 48, 7, 61], and (2) modulating node attributes utilizing Positional Encodings (PEs) [2, 47, 42]. However, both strategies face inherent limitations. The first tends to inherit drawbacks from GNNs due to its reliance on them, whereas the second introduces additional computational complexity due to the use of PEs, thereby restricting the models' universality[1] and scalability.

This leads to a fundamental question:

What underlying mechanism drives the effectiveness of diverse Graph Transformers?

A thorough understanding of the underlying mechanisms can offer valuable insights for developing more advanced and efficient architectures. Following this line, this paper theoretically investigates the mechanism shared by the aforementioned types of GTs and, based on this insight, proposes a novel GT architecture. In particular, the unified cross-aggregation mechanism (as formally defined in Definition 1) is explored by analytically decoupling topology and attribute representations from node representations. Specifically, the GNN block in GTs can be interpreted as aggregating topology representations into attribute representations (in Theorem 1), indicating that this category of GTs inherently incorporates cross-aggregation. Furthermore, GTs employing PEs contain diverse forms of cross-aggregation between topology and attribute representations. Therefore, the shared underlying mechanism among these GTs is cross-aggregation between graph topology and node attributes.

This understanding naturally leads to a key question:

How can we design an effective and efficient GT architecture grounded in cross-aggregation?

To this end, this paper proposes the *Universal Graph Cross-attention Transformer (UGCFormer)*, which implements the cross-aggregation mechanism via cross-attention. To be specific, it separately encodes graph topology and node attributes to obtain their initial representations. At its core lies a linearized Dual Cross-Attention (DCA) module that updates the topology and attribute representations by computing cross-attention scores among nodes and utilizing them for weighted aggregation. In theory, the DCA module adaptively captures both correlation and exclusion relationships between graph topology and node attributes, making it *simple yet effective*. Finally, the two representations are integrated to yield a comprehensive node representation. To prevent representation distortion, a consistency constraint is introduced to enforce mutual alignment between them.

The main contributions of this work are summarized as follows:

- **Mechanism Revelation**: We theoretically reveal a unified mechanism across typical Graph Transformers, namely cross-aggregation between graph topology and node attributes.
- **Model Innovation**: We propose UGCFormer, a GT architecture equipping with a linearized Dual Cross-Attention (DCA) module that implements the cross-aggregation mechanism.
- **Comprehensive Evaluation**: Extensive evaluations conducted on sixteen homophilic, heterophilic, and large-scale graphs demonstrate the universality and scalability of UGCFormer.

2 Preliminaries

This section begins by presenting the notation used throughout this paper. Then, it introduces the concepts of Graph Neural Networks (GNNs) and Graph Transformers (GTs).

2.1 Notations

The subject of this paper is the widely-used undirected attribute graph, denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the node set and edge set. $\mathcal{V}$ consists of n node instances $\{(\mathbf{x}_v, \mathbf{y}_v)\}_{v \in \mathcal{V}}$, where

[1]The ability of models to handle both homophilic and heterophilic graphs.

$\mathbf{x}_v \in \mathbb{R}^f$ and $\mathbf{y}_v \in \mathbb{R}^c$ denote the node attribute and label of node v, respectively. f is the dimension of attributes and c is the dimension of labels. $\mathcal{E} = \{(v_i, v_j)\}$ terms the edge set. Typically, graph topology is described by the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ where $a_{i,j} = 1$ only if $(v_i, v_j) \in \mathcal{E}$, and $a_{i,j} = 0$ otherwise. In formal terms, the graph $\mathcal{G}$ can be redescribed as $\mathcal{G}(\mathbf{A}, \mathbf{X})$. In the context of semi-supervised learning, the node labels are segmented into two sets: $\mathbf{Y}_L \in \mathbb{R}^{n_l \times c}$ for the labeled nodes and $\mathbf{Y}_U \in \mathbb{R}^{n_u \times c}$ for the unlabeled nodes.

To verify the model's universality, this paper examines graphs with varying degrees of homophily. In *homophilic* graphs, edges are typically formed between nodes with similar labels. Conversely, in *heterophilic* graphs, edges tend to form between nodes with dissimilar labels [35, 6, 60, 62].

2.2 Graph Neural Networks

Message Passing (MP)-based Graph Neural Networks (GNNs) follow an aggregation-combination strategy. Specifically, the representation of each node is iteratively updated by aggregating the features from its local neighbors and combining the aggregated features with its features, which is given by

$$\mathbf{h}_v^l \triangleq COM^l\left(\mathbf{h}_v^{l-1}, AGG^l\left(\{\mathbf{h}_u^{l-1} | u \in \mathcal{N}(v)\}\right)\right), \tag{1}$$

where $\mathcal{N}(v)$ denotes the set of neighboring nodes of node v. For the functions $AGG(\cdot)$ and $COM(,)$, vanilla GNNs, *e.g.*, GCN [27], adopt the sum function to implement them, that is,

$$GCN(\mathbf{A}, \mathbf{H}):\ \mathbf{H}^{l+1} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^l\mathbf{W}),\ \mathbf{H}^0 = \mathbf{X}, \tag{2}$$

where $\sigma(\cdot)$ stands for the nonlinear activation functions, and $\tilde{\mathbf{A}} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}$ is the normalized adjacency matrix with $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. $\mathbf{W}$ denotes the trainable projection parameters.

2.3 Transformers

Inspired by the success of Transformers in NLP [44], numerous variant models have been designed for multiple fields, including CV [22] and Graph Learning. They typically consist of four functional components: *attention module, feed-forward network, residual connection, and normalization*.

Self-attention Module. This is a core component of the vanilla Transformer to model intra-sequence relationships among all tokens [44]. Given a sequence containing n tokens $\mathbf{H} = [\mathbf{h}_i]_{i=0}^{n-1} \in \mathbb{R}^{n \times d}$, the module first projects $\mathbf{H}$ into Query $q(\mathbf{H})$, Key $k(\mathbf{H})$, and Value $k(\mathbf{H})$. It then employs the attention scores calculated from all Query-Key pairs to perform a weighted sum of the Value vectors.

A general formulation of the Self-Attention (SA) module is given by

$$SA(\mathbf{H}):\ \hat{\mathbf{H}}_{SA} = Softmax\left(\frac{q(\mathbf{H})k(\mathbf{H})^\top}{\sqrt{d}}\right)v(\mathbf{H}), \tag{3}$$

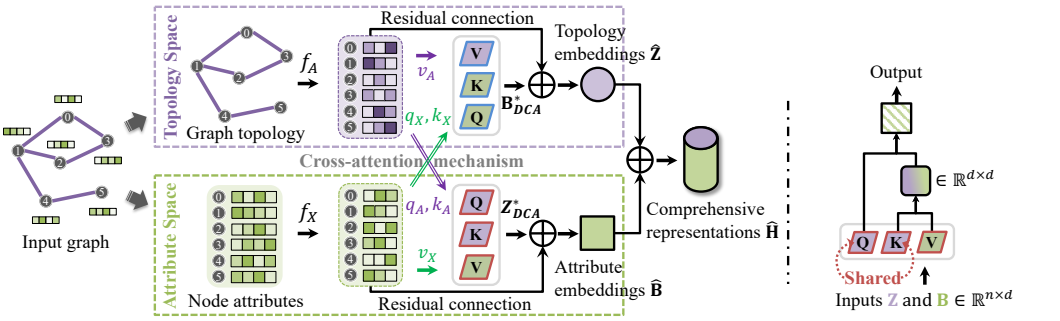
where $q(\cdot)$, $k(\cdot)$, and $v(\cdot)$ generate the Query, Key, and Value via MLPs [39] with learnable parameters $\mathbf{W}$. The attention score $Softmax(q(\mathbf{H})k(\mathbf{H})^\top/\sqrt{d}) \in \mathbb{R}^{n \times n}$ is computed via the scaled dot product of full-token pairs, resulting in a quadratic computational complexity.

Graph Transformers (GTs). Most existing models [49, 1, 37, 34, 4, 52, 61, 3] build upon the SA module. GTs differ from traditional Transformers in how they leverage topology information to capture structural biases. As discussed in the Introduction, two main strategies for incorporating topology information have achieved SOTA performance on node-level tasks: (1) integrating GNN blocks, and (2) modulating node attributes utilizing Positional Encodings (PEs).

Cross-Attention Module. Unlike self-attention, which models the intra-source relationships, cross-attention captures the interactions between two distinct sources. For the features from two different sources $\mathbf{H} \in \mathbb{R}^{n_1 \times d}$ and $\mathbf{Z} \in \mathbb{R}^{n_2 \times d}$, the Cross-Attention (CA) module can be expressed as

$$CA(\mathbf{Z}, \mathbf{H}):\ \hat{\mathbf{H}}_{CA} = Softmax\left(\frac{q(\mathbf{Z})k(\mathbf{H})^\top}{\sqrt{d}}\right)v(\mathbf{H}). \tag{4}$$

After the representation $\hat{\mathbf{H}}_{CA}$ is obtained, it is typically used as the cross-source representation to update $\mathbf{Z}$. Due to its exceptional capacity for modeling inter-source relationships, this module has been applied in diverse domains, *e.g.*, NLP [16] and CV [24]. However, it has received little attention in Graph Learning, largely due to *the lack of motivation and well-defined applied target*. Moreover, similar to the self-attention (Eq. 3), its computational complexity is quadratic, *i.e.*, $O(n_1 n_2)$.



(a) The pipeline of UGCFormer. (b) Linear Attention Module.

Figure 1: Overview of the proposed GT architecture UGCFormer and its linear attention module. (a) The pipeline of UGCFormer, which incorporates a dual cross-attention (DCA) module. First, two basic elements of graphs (*i.e.*, graph topology and node attributes) are independently processed in their respective spaces utilizing distinct projection layers $f_A(\cdot)$ and $f_X(\cdot)$. Next, the dual cross-attention (DCA) module with residual connections operates across the topology and attribute spaces, updating each representation by integrating correlated features from the other space. Finally, the two representations are combined to produce the final output representation. (b) Illustration of the proposed efficient cross-attention module, where parameters are shared between the query ($\mathbf{Q}$) and key ($\mathbf{K}$), and the representations are computed using linearized attention, given by $\mathbf{Q}(\mathbf{K}^\top\mathbf{V})$.

3 Methodology

This section starts by theoretically exploring the functional mechanism shared by Graph Transformers (GTs) that use Graph Neural Network (GNN) blocks and GTs that utilize Positional Encodings (PEs). Inspired by this mechanism, it introduces *UGCFormer*, a simple yet universal graph cross-attention Transformer with linear complexity. Finally, it gives a comprehensive analysis of UGCFormer.

3.1 Motivations

As previously discussed, the underlying mechanism behind the effectiveness of typical GTs remains insufficiently explored. To address this issue, this subsection proposes a cross-aggregation mechanism and theoretically examines how it is manifested in the two types of SOTA GTs.

The cross-aggregation mechanism is formally defined as follows.

Definition 1. (Cross-aggregation mechanism) Given two representations $\mathbf{B} \in \mathbb{R}^{n_1\times d_1}$ and $\mathbf{Z} \in \mathbb{R}^{n_2\times d_2}$ from different modalities (sources), which share at least one same dimension, *i.e.*, $n_1 = n_2$ or $d_1 = d_2$. A general formula for two types of cross-aggregations can be expressed as

$$\hat{\mathbf{Z}} \triangleq \begin{cases} Sim(\mathbf{Z},\mathbf{B})\mathbf{B}, & \text{if } d_1 = d_2, \\ \mathbf{B}\, Sim(\mathbf{B},\mathbf{Z}), & \text{if } n_1 = n_2, \end{cases} \tag{5}$$

where $Sim(\mathbf{Z},\mathbf{B})$ denotes a similarity function between $\mathbf{Z}$ and $\mathbf{B}$, such as cosine similarity.

The first case corresponds to *sample (node)-level* aggregation, *e.g.*, cross-attention (Eq. 4), while the second corresponds to *dimension (feature)-level* aggregation [55, 56, 61]. When $\mathbf{Z} = \mathbf{B}$, Eq. 5 reduces to a self-aggregation (e.g., self-attention). Accordingly, two theorems are presented.

Theorem 1. *In typical Graph Transformers, the diffusion matrix of GNN blocks can be expressed via eigendecomposition as* $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, *where* $\mathbf{U}$ *and* $\mathbf{\Lambda} = diag([\lambda_1,\ldots,\lambda_n])$ *represents eigenvectors and eigenvalues, respectively (in descending order). Accordingly, the GNN block can be viewed as a cross-aggregation between attribute representations* $\mathbf{XW}$ *and topology representations* $\mathbf{U}\sqrt{\mathbf{\Lambda}}$.

Theorem 2. *Given the modulated node attributes using any PE, i.e.,* $\hat{\mathbf{X}} = [\mathbf{X};\mathbf{P}]$, *where* $\mathbf{P} \in \mathbb{R}^{n\times k}$ *represents the PE and* $[;]$ *denotes concatenation operator. PE-based GTs (Eq. 3) inherently contain a cross-aggregation between attribute representations* $\mathbf{XW}$ *and topology representations* $\mathbf{PW}$.

The proofs for Theorems 1 and 2 are provided in Sections B and C, respectively. In short, the key mechanism of GTs using GNNs and PEs is *Cross-Aggregation between topology and attributes*.

3.2 UGCFormer

Motivated by the cross-aggregation mechanism explored in the previous subsection, this subsection introduces UGCFormer, a simple yet universal GT. At its core, UGCFormer employs a linearized cross-attention module that implements the cross-aggregation mechanism to capture interactions between graph topology and node attributes. UGCFormer consists of four modules, each of which is described below. The detailed implementation is provided in Algorithm 1.

Initial Representation Layer. Two different projection layers are utilized to independently generate initial representations for the two types of graph information. For simplicity, MLPs are used to process the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ and the attribute matrix $\mathbf{X} \in \mathbb{R}^{n\times f}$, producing the corresponding initial representations $\mathbf{Z}$ and $\mathbf{B}$, that is,

$$\mathbf{Z}^0 = MLP_A(\mathbf{A}), \quad \mathbf{B}^0 = MLP_X(\mathbf{X}) \in \mathbb{R}^{n\times d}, \tag{6}$$

where $MLP_A(\cdot)$ and $MLP_X(\cdot)$ term the MLPs for processing topology and attributes, respectively.

Dual Cross-attention Module. As an implementation of the cross-aggregation (in Definition 1), this module is designed to capture the interactions between these two types of graph information. However, directly employing the cross-attention (Eq. 4) may result in two issues: (1) unacceptable quadratic computational complexity due to the calculation of dot products for all node pairs, and (2) an increased number of parameters and overfitting risk due to the use of two separate channels.

To alleviate these drawbacks, the proposed Dual Cross-Attention (DCA) module adopts two strategies: (1) *linearized attention computation* [48] and (2) *parameter sharing*, as shown in Fig. 1(b). Firstly, through approximating or replacing the Softmax attention utilizing separate kernel functions, the computation order in the SA module can be reordered from the standard (Query×Key)×Value (Eq. 3) to the more efficient Query×(Key×Value) format [26]. However, this strategy is unsuitable for the cross-attention module. Specifically, the attention score $k(\mathbf{B})^\top v(\mathbf{B}) \in \mathbb{R}^{d\times d}$ computes the similarity between features within the same space, rather than across different spaces. To ensure cross-space interaction, DCA sets the Key to originate from the same space as the Query. Moreover, DCA shares parameters between the Query and Key to reduce the number of parameters.

To streamline the description of the update process for topology representations and attribute representations, two abstract representations $\mathbf{H}_1$ and $\mathbf{H}_2$ are introduced. For clarity, layer indices are omitted. The general formulation of the DCA module is given as follows:

$$DCA(\mathbf{H}_1, \mathbf{H}_2): \qquad \mathbf{Q} = q(\mathbf{H}_1), \mathbf{K} = k(\mathbf{H}_1), \mathbf{V} = v(\mathbf{H}_2), \tag{7}$$

$$\tilde{\mathbf{Q}} = \frac{\mathbf{Q}}{\|\mathbf{Q}\|_{\mathcal{F}}}, \quad \tilde{\mathbf{K}} = \frac{\mathbf{K}}{\|\mathbf{K}\|_{\mathcal{F}}}, \tag{8}$$

$$\mathbf{H}_1^* = \mathbf{D}^{-1}\left(\mathbf{V} + \frac{1}{n}\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \mathbf{V})\right), \tag{9}$$

where $q(\cdot)$ and $k(\cdot)$ stand for the Query and Key functions, respectively, with $q(\cdot) = k(\cdot)$. And $v_A(\cdot)$ represents the Value function. These functions are implemented as MLPs. $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm. $\mathbf{D} = Diag(\mathbf{1} + \frac{1}{n}\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \mathbf{1}))$ stands for a diagonal matrix and $\mathbf{1}$ is an all-one vector.

The topology-related attribute representations can be obtained as $\mathbf{B}^*_{DCA} = DCA(\mathbf{B}, \mathbf{Z})$. Then, the topology representations are updated via

$$\hat{\mathbf{Z}} = (1-\tilde{\lambda})\tilde{\mathbf{A}}\mathbf{V} \ + \ \tilde{\lambda}\mathbf{B}^*_{DCA}, \tag{10}$$

where $\tilde{\mathbf{A}}$ denotes the normalized adjacency matrix. The first term denotes the topology representation updated purely from the topology space, which can be viewed as being obtained via *spectral clustering* [46, 53] (see Theorem 3). $\tilde{\lambda} = \mathrm{Tanh}(\lambda)$ stands for a scalar to balance these two terms, where λ is a learnable parameter. Combining these two terms allows for the fusion of topological details alongside the topology-related attribute information into the final topology representations.

Similarly, the attribute representations are updated by incorporating relevant information from the topology space, that is, $\mathbf{Z}^*_{DCA} = DCA(\mathbf{Z}, \mathbf{B})$, with their representations. This can be expressed as

$$\hat{\mathbf{B}} = (1-\tilde{\gamma})\mathbf{B}^0 \ + \ \tilde{\gamma}\mathbf{Z}^*_{DCA}, \tag{11}$$

where $\tilde{\gamma} = \mathrm{Tanh}(\gamma)$ denotes a scalar to trade off the two terms with γ denotes a learnable parameter. Note that DCA requires two separate sets of network parameters to generate the attribute and topology

Algorithm 1: UGCFormer

Input: Graph $\mathcal{G}(\mathbf{A}, \mathbf{X})$ with labels $\mathbf{Y}$, hyperparameters α, β and τ.
Output: Trained network parameters Θ^*.
Initialization: Network parameters Θ,
while not converged **do**
1. Generate two initial node representations $\mathbf{Z}^0$ and $\mathbf{B}^0$ via Eq. 6;
2. Get two updated node representations $\hat{\mathbf{Z}}$ and $\hat{\mathbf{B}}$ via Eqs. 10 and 11;
3. Obtain the final predictions $\hat{\mathbf{Y}}$ via Eq. 12;
4. Calculate the overall loss $\mathcal{L}_{final}$ via Eqs. 13 and 14;
5. Optimize the parameters via $\Theta^* \leftarrow \text{Adam}(\mathcal{L}, \Theta)$;

end
return Parameters Θ^*

representations, *e.g.*, $q_A(\cdot)$ and $q_X(\cdot)$ represent the Query for graph topology and node attributes, respectively, as shown in Fig. 1.

Prediction Layer. After obtaining the topology representations $\hat{\mathbf{Z}}$ and attribute representations $\hat{\mathbf{B}}$ through l layers, the final node representations can be generated by weight combining them. Next, the predictions are generated via an MLP network and nonlinearities (*i.e.*, $Softmax(\cdot)$), that is,

$$\hat{\mathbf{Y}} = Softmax\left(MLP\left((1-\alpha)\hat{\mathbf{Z}} + \alpha\hat{\mathbf{B}}\right)\right), \tag{12}$$

where α denotes a scalar that adjusts attention to topology and attribute representations. $\hat{\mathbf{Y}} \in \mathbb{R}^{n\times c}$ represents the predictions, indicating the estimated outcomes for each of the n nodes across c classes.

Objective Function. Note that the proposed DCA module, with a large number of parameters across two distinct spaces, is susceptible to representation distortion caused by overfitting [8], especially when the number of training nodes is limited. Thus, a consistency constraint is introduced to align the two representations $\hat{\mathbf{Z}}$ and $\hat{\mathbf{B}}$. First, pseudo-labels are derived by averaging the two representations. For a node v, its pseudo-label can be computed as $\mathbf{y}_v = \frac{1}{2}(\hat{\mathbf{z}}_v + \hat{\mathbf{b}}_v)$. Next, low-entropy pseudo-labels are obtained through a sharpening technique that controls the sharpness of the distribution. This can be formulated as $\bar{y}_{i,j} = y_{i,j}^{\frac{1}{\tau}} / \sum_{k=0}^{c-1} y_{i,j}, (0 \le j \le c-1)$, where $\tau \in (0, 1]$ denotes a scaling factor that controls the sharpness of the distribution.

Once the pseudo-label is obtained, the next step is to calculate the squared Euclidean distance between it and the two representations, which is given by

$$\mathcal{L}_{con}(\hat{\mathbf{Z}}, \hat{\mathbf{B}}) = \frac{1}{2}\sum_{i}^{n-1}\left(\|\bar{\mathbf{y}}_i - \hat{\mathbf{z}}_i\|_2^2 + \|\bar{\mathbf{y}}_i - \hat{\mathbf{b}}_i\|_2^2\right). \tag{13}$$

The overall objective of UGCFormer is to minimize the weighted sum of the cross-entropy loss and the consistency loss, defined as follows:

$$\mathcal{L}_{overall} = \mathcal{L}_{ce} + \beta\mathcal{L}_{con}, \tag{14}$$

where $\mathcal{L}_{ce} = -\sum_{v\in\mathcal{V}_L} \mathbf{y}_v \log \hat{\mathbf{y}}_v$ and β stands for a balance hyperparameter.

3.3 Model Analysis

This subsection provides a comprehensive analysis of UGCFormer. First, the computational complexity of UGCFormer is analyzed. Then, the simplicity of UGCFormer is examined through architectural comparison with existing GTs. Finally, the effectiveness of UGCFormer is theoretically justified.

Complexity Analysis. UGCFormer operates with **linear time complexity**. The time complexity for generating initial representations through the projection layer is $O(md+nd^2)$ as the adjacency matrix is sparse, where m represents the number of edges. Secondly, owing to the linearized cross-attention module, the aggregation operator incurs a computational overhead of $O(nd^2)$. Finally, obtaining the predictions involves feature mapping and element-wise operations, resulting in a complexity of $O(nd)$. UGCFormer operates with **linear space complexity**. The space required to store the input topology

Table 1: Accuracy (ACC) or ROC-AUC in percentage (mean$_{\pm \text{std}}$) over 10 trials of the node classification task on homophilic graphs. Best and runner-up models are in bold and underlined, respectively.

Model	**Cora**	**CiteSeer**	**PubMed**	**Photo**	**CS**	**Physics**	**Questions**		
Metric	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ROC-AUC ↑	Avg ↑	Rank ↓
GCN	$81.60_{\pm 0.40}$	$71.60_{\pm 0.40}$	$78.80_{\pm 0.60}$	$92.70_{\pm 0.20}$	$92.92_{\pm 0.12}$	$96.18_{\pm 0.07}$	$76.28_{\pm 0.64}$	84.30	13.29
GAT	$83.00_{\pm 0.70}$	$72.10_{\pm 1.10}$	$79.00_{\pm 0.40}$	$93.87_{\pm 0.10}$	$93.61_{\pm 0.14}$	$96.17_{\pm 0.08}$	$74.94_{\pm 0.56}$	84.67	11.43
GraphSAGE	$82.68_{\pm 0.47}$	$71.93_{\pm 0.85}$	$79.41_{\pm 0.53}$	$94.59_{\pm 0.14}$	$93.91_{\pm 0.13}$	$96.49_{\pm 0.06}$	$76.44_{\pm 0.62}$	85.06	9.71
APPNP	$83.30_{\pm 0.50}$	$71.80_{\pm 0.50}$	$80.10_{\pm 0.20}$	$94.32_{\pm 0.14}$	$94.49_{\pm 0.07}$	$96.54_{\pm 0.07}$	$75.51_{\pm 0.23}$	85.15	8.14
GPR-GNN	$84.20_{\pm 0.50}$	$71.60_{\pm 0.80}$	$80.07_{\pm 0.92}$	$94.49_{\pm 0.16}$	$95.13_{\pm 0.09}$	$96.85_{\pm 0.08}$	$67.15_{\pm 1.92}$	84.21	8.71
LINKX	$77.95_{\pm 0.12}$	$68.25_{\pm 0.24}$	$77.36_{\pm 0.42}$	$91.97_{\pm 0.19}$	$94.77_{\pm 0.19}$	$96.29_{\pm 0.13}$	$75.71_{\pm 1.40}$	83.19	13.14
GloGNN	$82.17_{\pm 0.29}$	$71.74_{\pm 0.88}$	$\underline{80.37}_{\pm 0.95}$	$95.10_{\pm 0.20}$	$95.00_{\pm 0.10}$	$96.97_{\pm 0.15}$	$67.15_{\pm 1.92}$	84.07	8.43
GraphGPS	$82.84_{\pm 1.03}$	$72.73_{\pm 1.23}$	$79.94_{\pm 0.26}$	$95.06_{\pm 0.13}$	$93.93_{\pm 0.12}$	$97.12_{\pm 0.19}$	$71.73_{\pm 1.47}$	84.76	8.29
NodeFormer	$82.20_{\pm 0.90}$	$72.50_{\pm 1.10}$	$79.90_{\pm 1.00}$	$93.46_{\pm 0.35}$	$95.64_{\pm 0.22}$	$96.24_{\pm 0.24}$	$74.27_{\pm 1.46}$	84.89	9.57
NAGphormer	$82.12_{\pm 1.18}$	$71.47_{\pm 1.30}$	$79.73_{\pm 0.28}$	$95.49_{\pm 0.11}$	$\underline{95.75}_{\pm 0.09}$	$\underline{97.34}_{\pm 0.03}$	$74.98_{\pm 0.63}$	85.27	7.71
Exphormer	$82.77_{\pm 1.38}$	$71.63_{\pm 1.19}$	$79.46_{\pm 0.35}$	$95.35_{\pm 0.22}$	$94.93_{\pm 0.01}$	$96.89_{\pm 0.09}$	$74.67_{\pm 0.79}$	85.10	8.86
GOAT	$83.18_{\pm 1.27}$	$71.99_{\pm 1.26}$	$79.13_{\pm 0.38}$	$92.96_{\pm 1.48}$	$94.21_{\pm 0.38}$	$96.45_{\pm 0.28}$	$75.76_{\pm 1.66}$	84.81	10.00
SGFormer	$\underline{84.50}_{\pm 0.80}$	$72.60_{\pm 0.20}$	$80.30_{\pm 0.60}$	$95.10_{\pm 0.47}$	$94.78_{\pm 0.20}$	$96.60_{\pm 0.18}$	$72.15_{\pm 1.31}$	85.14	6.57
Polynormer	$83.25_{\pm 0.93}$	$72.31_{\pm 0.78}$	$79.24_{\pm 0.43}$	$\mathbf{96.46}_{\pm 0.26}$	$95.53_{\pm 0.16}$	$97.27_{\pm 0.08}$	$\underline{76.91}_{\pm 1.63}$	$\underline{85.85}$	$\underline{4.71}$
Gradformer	$82.95_{\pm 0.73}$	$\underline{72.80}_{\pm 0.59}$	$80.14_{\pm 0.48}$	$95.76_{\pm 0.28}$	$94.21_{\pm 0.29}$	$97.06_{\pm 0.16}$	$74.71_{\pm 1.07}$	85.38	6.14
UGCFormer	$\mathbf{84.94}_{\pm 0.43}$	$\mathbf{73.41}_{\pm 0.27}$	$\mathbf{81.79}_{\pm 0.81}$	$\underline{96.21}_{\pm 0.31}$	$\mathbf{95.91}_{\pm 0.23}$	$\mathbf{97.35}_{\pm 0.17}$	$\mathbf{77.02}_{\pm 0.76}$	**86.66**	**1.14**

and attributes is $O(m + nd)$, where m corresponds to the number of edges and nd accounts for the feature matrix. The aggregated and updated representations each require $O(nd)$ space, since their dimensions do not exceed those of the input feature matrix. In the linearized attention computation (Fig. 1(b)), the attention matrix contributes an additional $O(d^2)$ space overhead.

Components. To leverage discriminative graph topology and capture structural biases, existing GTs often resort to auxiliary components that compromise their efficiency and effectiveness. Specifically, the positional or structural encodings (*e.g.*, Laplacian eigenvector encodings) used in GraphGPS [37], NAGphormer [2], Exphormer [42], and GOAT [29] as well as augmented training losses (*e.g.*, edge regularization loss) in NodeFormer, often necessitate cubic computational complexity and quadratic space consumption. Moreover, the GNN module tends to generate representations that are susceptible to issues caused by the limited message passing. In contrast, the proposed UGCFormer features a streamlined and efficient design that relies solely on a linear cross-attention module.

Theoretical Justification. Though designed to be simple and intuitive, the proposed UGCFormer is theoretically guaranteed to be effective from a graph optimization perspective [54, 57].

Theorem 3. *Let* $\mathbf{Z}$ *and* $\mathbf{B}$ *denote the topology representations and attribute representations, respectively. The representation update in the dual cross-attention module DCA (Eq. 10 and Eq. 11) is equivalent to solving an optimization problem with the objective function:*

$$\arg\min_{\mathbf{Z},\mathbf{B}} \lambda \operatorname{Tr}(\mathbf{Z}^\top \tilde{\mathbf{L}} \mathbf{Z}) + \|\mathbf{B} - MLP(\mathbf{X})\|_F^2 - \eta \|\mathbf{Z}^\top \mathbf{B}\|_F^2, \tag{15}$$

where $\tilde{\mathbf{L}}$ *terms the Laplacian matrix of* $\tilde{\mathbf{A}}$*,* λ *and* η *are the scalars used to balance these three terms.*

In Eq. 15, the first term stands for a relaxed optimization problem widely used in spectral clustering [46]. Thus, the DCA seeks to generate topology representations that capture mesoscopic community structures. The second term measures the distance between the attribute representation $\mathbf{B}$ and its initial representation $MLP(\mathbf{X})$. The third term denotes the statistical dependence measure, approximated by the Hilbert-Schmidt Independence Criterion (HSIC) [20], that is, $\text{HSIC}(\mathbf{Z}, \mathbf{B}) \approx \operatorname{Tr}(\mathbf{Z}\mathbf{Z}^\top \mathbf{B}\mathbf{B}^\top) = \|\mathbf{Z}^\top \mathbf{B}\|_F^2$, which reflects the dependence between topology and attribute representations. Therefore, the interaction, whether mutual correlation (positive weights) or exclusion (negative weights), can be modulated by the parameters. In summary, Theorem 3 indicates that UGCFormer focuses on learning representations by mining the interactions of two basic graph information.

4 Experiments

This section evaluates the effectiveness and universality of the proposed UGCFormer by comparing its performances against various diverse graph learning models on the node classification task. Moreover,

Table 2: Accuracy (ACC) in percentage (mean$_{\pm \text{std}}$) over 10 trials of the node classification task on heterophilic graphs. Best and runner-up models are in bold and underlined, respectively.

Model	**Cornell**	**Texas**	**Wisconsin**	**Actor**	**Chameleon**	**Squirrel**	**Ratings**		
Metric	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	ACC ↑	Avg ↑	Rank ↓
GCN	$58.41_{\pm 3.28}$	$65.61_{\pm 4.80}$	$61.28_{\pm 5.87}$	$30.63_{\pm 0.62}$	$43.43_{\pm 1.92}$	$41.30_{\pm 0.94}$	$47.77_{\pm 0.69}$	49.78	11.57
GAT	$58.29_{\pm 3.52}$	$60.73_{\pm 6.20}$	$63.64_{\pm 6.18}$	$30.36_{\pm 0.94}$	$40.14_{\pm 1.57}$	$35.09_{\pm 0.70}$	$47.95_{\pm 0.53}$	48.03	14.57
GraphSAGE	$75.95_{\pm 5.31}$	$82.43_{\pm 6.07}$	$81.18_{\pm 4.56}$	$34.23_{\pm 1.07}$	$39.11_{\pm 5.05}$	$36.46_{\pm 2.16}$	$53.11_{\pm 0.54}$	57.50	11.00
APPNP	$73.68_{\pm 3.97}$	$74.57_{\pm 2.48}$	$70.61_{\pm 3.47}$	$35.18_{\pm 1.21}$	$39.42_{\pm 3.87}$	$38.13_{\pm 2.67}$	$49.78_{\pm 0.72}$	54.49	12.71
GPR-GNN	$78.11_{\pm 6.55}$	$81.35_{\pm 5.32}$	$82.55_{\pm 6.23}$	$35.16_{\pm 0.85}$	$39.93_{\pm 3.30}$	$38.95_{\pm 1.99}$	$43.90_{\pm 0.48}$	57.14	11.86
LINKX	$77.84_{\pm 5.81}$	$74.60_{\pm 8.37}$	$75.49_{\pm 5.72}$	$36.10_{\pm 1.55}$	$40.02_{\pm 2.35}$	$39.88_{\pm 2.53}$	$51.36_{\pm 0.47}$	56.47	11.00
GloGNN	$\underline{83.51}_{\pm 4.26}$	$\underline{84.32}_{\pm 4.15}$	$\underline{87.06}_{\pm 3.53}$	$37.35_{\pm 1.30}$	$38.43_{\pm 3.74}$	$30.30_{\pm 1.92}$	$37.28_{\pm 0.66}$	56.89	8.14
GraphGPS	$82.06_{\pm 5.73}$	$82.21_{\pm 6.14}$	$85.36_{\pm 4.24}$	$36.18_{\pm 1.27}$	$40.79_{\pm 4.03}$	$39.67_{\pm 2.84}$	$53.10_{\pm 0.42}$	59.91	7.29
NodeFormer	$82.15_{\pm 6.72}$	$81.68_{\pm 4.65}$	$83.41_{\pm 5.51}$	$36.28_{\pm 1.25}$	$43.09_{\pm 2.81}$	$40.61_{\pm 1.25}$	$50.12_{\pm 0.64}$	59.62	7.43
NAGphormer	$79.97_{\pm 6.07}$	$80.18_{\pm 4.57}$	$82.97_{\pm 2.98}$	$34.36_{\pm 0.75}$	$\underline{44.61}_{\pm 3.10}$	$41.27_{\pm 1.09}$	$52.51_{\pm 0.83}$	59.41	7.86
Exphormer	$83.07_{\pm 4.31}$	$82.81_{\pm 3.52}$	$83.90_{\pm 4.31}$	$36.82_{\pm 1.95}$	$41.63_{\pm 3.12}$	$40.32_{\pm 1.59}$	$52.08_{\pm 0.81}$	60.06	6.00
GOAT	$83.18_{\pm 1.27}$	$71.99_{\pm 1.26}$	$79.13_{\pm 0.38}$	$36.55_{\pm 1.19}$	$42.56_{\pm 3.17}$	$40.81_{\pm 0.54}$	$49.68_{\pm 0.50}$	57.70	8.53
SGFormer	$81.64_{\pm 3.88}$	$84.29_{\pm 5.67}$	$83.59_{\pm 5.42}$	$\mathbf{37.79}_{\pm 1.89}$	$\mathbf{44.93}_{\pm 3.91}$	$\mathbf{41.80}_{\pm 2.27}$	$48.01_{\pm 0.49}$	$\underline{60.29}$	$\underline{4.86}$
Polynormer	$81.90_{\pm 4.17}$	$82.57_{\pm 5.11}$	$83.95_{\pm 2.98}$	$37.01_{\pm 1.10}$	$41.97_{\pm 3.18}$	$40.87_{\pm 1.96}$	$\underline{53.29}_{\pm 0.23}$	60.22	5.14
Gradformer	$83.06_{\pm 5.16}$	$82.19_{\pm 5.24}$	$84.26_{\pm 2.24}$	$36.58_{\pm 0.71}$	$40.73_{\pm 3.69}$	$40.29_{\pm 1.88}$	$53.11_{\pm 0.29}$	60.03	6.43
UGCFormer	$\mathbf{85.14}_{\pm 5.83}$	$\mathbf{84.59}_{\pm 4.69}$	$\mathbf{87.36}_{\pm 3.30}$	$\underline{37.41}_{\pm 0.79}$	$43.28_{\pm 2.17}$	$\underline{41.56}_{\pm 2.01}$	$\mathbf{53.48}_{\pm 0.14}$	**61.83**	**1.71**

it provides additional analysis experiments to enhance the understanding of UGCFormer. Refer to Section E for details on the datasets, baselines, and experimental setups.

4.1 Experimental Results

Homophilic Graphs. The experiment results for node classification on homophilic graphs are shown in Tab. 1, from which three key observations can be made. Firstly, the performance of the backbone GNNs (*e.g.*, GCN and GAT) lags behind that of GTs. To be specific, on six of the seven homophilic graphs, the models that rank in the top two positions are GTs. This is primarily because most GTs, such as NAGphormer, are built upon these backbone GNNs and specifically address the shortcomings of GNNs in capturing long-range dependencies. Secondly, the proposed UGCFormer outperforms all baseline GTs across six of the seven datasets and achieves the optimal rank, demonstrating its consistent superior performance. In particular, on PubMed, UGCFormer achieves a performance that is 2.55% higher than the baseline Polynormer, which has an average rank of second, and its average rank is significantly lower. Thirdly, compared with the baseline LINKX, which also processes graph topology and node attributes separately and does not leverage message passing, UGCFormer consistently achieves better results across all datasets. This can be attributed to its ability to capture the interactions between these two types of graph information and alleviate the representation distortion, which LINKX does not account for. This highlights the rationality of UGCFormer's design.

Heterophilic Graphs. Tab. 2 shows the results of the node classification task on seven heterophilic graphs, highlighting three key observations. Firstly, the baseline GTs perform slightly better than the baseline GNNs, but the difference is not substantial. In specific, the baseline GNNs, particularly GloGNN on Cornell, Texas, and Wisconsin, and GraphSAGE on the Ratings, achieve top-two results on five of the seven datasets. This can be attributed to the high complexity and large number of parameters in GTs, which make them prone to overfitting. Therefore, the baseline SGFormer, which linearly combines the local representation from the GNN module and the global representation from the GT module, achieves superior performance. This is evidenced by its ranking in the top two for three datasets. Secondly, the proposed UGCFormer outperforms the GT baselines on the majority of heterophilic graphs, proving its effectiveness. For example, on Cornell, UGCFormer exceeds the second-ranked GT, *i.e.*, GOAT, by a significant margin of 1.96%. Thirdly, UGCFormer consistently outperforms the baseline LINKX on all heterophilic datasets, highlighting the significance of capturing the relevance between graph topology and node attributes. Overall, UGCFormer achieves performance improvements on both homophilic and heterophilic graphs, demonstrating its universality.

Scalability Study. To evaluate the scalability of the proposed UGCFormer, this experiment quantitatively changes the network size and records the running time and GPU memory usage. Specifically, it utilizes the ogbn-arxiv to randomly sample subsets of nodes, with the node numbers varying from 10K to 100K. As shown in Fig. 2, the running time and GPU memory usage of UGCFormer increase linearly with the size of the sampled graph. For example, the training time and memory usage with

100k nodes are approximately five times higher than with 20k nodes. This indicates that UGCFormer exhibits linear time and space complexity, consistent with the conclusion in Section 3.3.

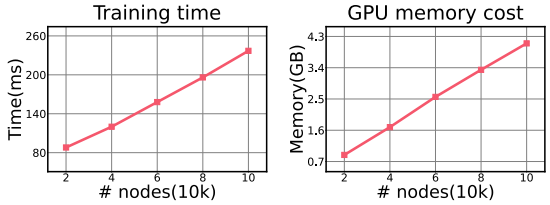


Figure 2: Training time and GPU memory usage of UGCFormer.

Node Property Prediction. This experiment seeks to evaluate the effectiveness and scalability of GTs by comparing them with GNNs on two large-scale benchmark datasets. Upon examining Tab. 3, which presents the results of the node property prediction task on these two datasets, two key conclusions can be drawn. Firstly, the backbone GTs generally outperform the backbone GNNs, which not only highlights the superiority of GTs but also underscores their scalability—a key challenge that GTs aim to address. This can be attributed to the integration of the GNN blocks in GTs, exemplified by SGFormer. These GTs generate the final prediction by combining the local representations from the GNN module with the global representations from the GT module. Secondly, the proposed UGCFormer achieves optimal performance on these two datasets, indicating its effectiveness and scalability on large graphs.

Table 3: Node property prediction performances on two large-scale graphs.

Model Metric	**ogbn-proteins** ROC-AUC ↑	**ogbn-arxiv** ACC ↑
GCN	$72.51_{\pm 0.35}$	$71.74_{\pm 0.29}$
GAT	$72.02_{\pm 0.44}$	$71.95_{\pm 0.36}$
GPRGNN	$71.10_{\pm 0.12}$	$71.10_{\pm 0.12}$
LINKX	$66.18_{\pm 0.33}$	$71.59_{\pm 0.71}$
GraphGPS	$76.83_{\pm 0.26}$	$70.97_{\pm 0.41}$
NodeFormer	$77.45_{\pm 1.15}$	$67.19_{\pm 0.83}$
NAGphormer	$73.61_{\pm 0.33}$	$70.13_{\pm 0.55}$
Exphormer	$74.58_{\pm 0.26}$	$72.44_{\pm 0.28}$
GOAT	$74.84_{\pm 1.16}$	$72.41_{\pm 0.40}$
SGFormer	$\underline{79.53}_{\pm 0.38}$	$72.63_{\pm 0.13}$
Polynormer	$78.97_{\pm 0.47}$	$\underline{73.46}_{\pm 0.16}$
Gradformer	$77.64_{\pm 0.51}$	$72.71_{\pm 0.20}$
UGCFormer	$\mathbf{79.95}_{\pm 0.75}$	$\mathbf{74.02}_{\pm 0.17}$

4.2 Additional Analysis

Ablation Study. This experiment evaluates the contributions of the proposed cross-attention module and the consistency constraint by comparing UGCFormer with two variants lacking these components. Fig. 3 shows that these variants consistently underperform UGCFormer across the four datasets. This illustrates that the efficacy of UGCFormer stems from the collective contribution of all components. Besides, even without the consistency loss, the variant model (w/o $\mathcal{L}_{con}$) still provides competitive performance compared to the baseline GTs, as seen in Table 1. This highlights the effectiveness of the cross-attention module and thereby reaffirms the rationality of the UGCFormer architecture.

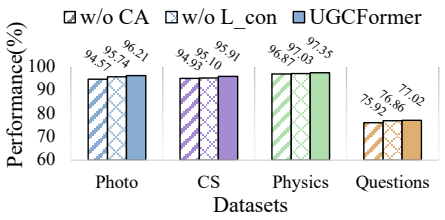


Figure 3: Impact of functional components (*i.e.*, the CA and consistency constraint).

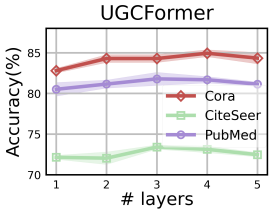


Figure 4: Performance variations for varying l.

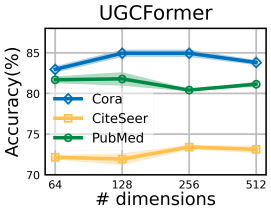


Figure 5: Performance variations for varying d.

Parameter Sensitivity Analysis. These experiments aim to provide an intuitive understanding for the selection of hyperparameters. Performance changes due to varying the number of layers (l) and layer dimensions (d) are shown in Figs. 4 and 5, respectively. **Number of Layers.** Fig. 4 shows that UGCFormer achieves stable performance across various layer numbers $\{1, 2, 3, 4, 5\}$. Specifically, performance fluctuations are minimal, within 2.2% on the Cora, 1.4% on the CiteSeer, and 1.3% on PubMed. This indicates that UGCFormer is relatively insensitive to the number of layers. Additionally, optimal performance is achieved with $\{3, 4\}$, likely due to the risk of over-smoothing in

deeper models. **Hidden Layer Dimension.** As shown in Fig. 5, UGCFormer maintains consistent performance across the hidden dimension range $\{64, 128, 256, 512\}$. For example, on the Cora, which shows the most significant performance variation, the difference is less than 2%. This indicates that UGCFormer is not sensitive to this parameter. Additionally, optimal performance on the three datasets corresponds to $d \in \{128, 256\}$, rather than the highest value of 512. This suggests that larger dimensions can lead to overfitting and distorted representations. Additional hyper-parameters (including α and β) are analyzed in Section E.4.

5 Conclusions

By revisiting two typical Graph Transformers (GTs), this study has uncovered a potential functional mechanism: cross-aggregation between graph topology and node attributes. To effectively implement this mechanism, this paper introduces UGCFormer, a linearized graph cross-attention Transformer. Extensive experiments on sixteen graph benchmarks demonstrate its effectiveness and efficiency.

6 Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (No. 92570118, U22B2036, 62376088, 62272020, 62025604, 92370111, 62272340, 62261136549), in part by the Hebei Natural Science Foundation (No. F2024202047), in part by the National Science Fund for Distinguished Young Scholarship (No. 62025602), in part by the Hebei Yanzhao Golden Platform Talent Gathering Programme Core Talent Project (Education Platform) (HJZD202509), in part by the Post-graduate's Innovation Fund Project of Hebei Province (CXZZBS2025036), in part by the Tencent Foundation, and in part by the XPLORER PRIZE.

References

[1] Dexiong Chen, Leslie O'Bray, and Karsten M. Borgwardt. Structure-aware transformer for graph representation learning. In ICML, volume 162, pages 3469–3489, 2022.

[2] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In ICLR, 2023.

[3] Jinsong Chen, Chenyang Li, Gaichao Li, John E. Hopcroft, and Kun He. Rethinking tokenized graph transformers for node classification. CoRR, abs/2502.08101, 2025.

[4] Jinsong Chen, Hanpeng Liu, John E. Hopcroft, and Kun He. Leveraging contrastive learning for enhanced node representations in tokenized graph transformers. In NeurIPS, 2024.

[5] Zhaoliang Chen, Zhihao Wu, Ylli Sadikaj, Claudia Plant, Hong-Ning Dai, Shiping Wang, Yiu-Ming Cheung, and Wenzhong Guo. Adedgedrop: Adversarial edge dropping for robust graph neural networks. IEEE Transactions on Knowledge and Data Engineering, 37(9):4948–4961, 2025.

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In ICLR, 2021.

[7] Chenhui Deng, Zichao Yue, and Zhiru Zhang. Polynormer: Polynomial-expressive graph transformer in linear time. In ICLR, 2024.

[8] Claudio Filipi Goncalves dos Santos and João Paulo Papa. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. ACM Comput. Surv., 54(10s):213:1–213:25, 2022.

[9] Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In NeurIPS, 2022.

[10] Ruiyi Fang, Bingheng Li, Zhao Kang, Qiuhao Zeng, Nima Hosseini Dashtbayaz, Ruizhi Pu, Charles Ling, and Boyu Wang. On the benefits of attribute-driven graph domain adaptation. In ICLR, 2025.

[11] Ruiyi Fang, Bingheng Li, Jingyu Zhao, Ruizhi Pu, Qiuhao Zeng, Gezheng Xu, Charles Ling, and Boyu Wang. Homophily enhanced graph domain adaptation. In ICML, 2025.

[12] Ruiyi Fang, Liangjian Wen, Zhao Kang, and Jianzhuang Liu. Structure-preserving graph representation learning. In ICDM, pages 927–932. IEEE, 2022.

[13] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. CoRR, abs/1903.02428, 2019.

[14] Chaofan Fu, Guanjie Zheng, Chao Huang, Yanwei Yu, and Junyu Dong. Multiplex heterogeneous graph neural network with behavior pattern modeling. In SIGKDD, pages 482–494, 2023.

[15] Lele Fu, Bowen Deng, Sheng Huang, Tianchi Liao, Shirui Pan, and Chuan Chen. Less is more: Federated graph learning with alleviating topology heterogeneity from a causal perspective. In ICML, 2025.

[16] Mozhdeh Gheini, Xiang Ren, and Jonathan May. Cross-attention is all you need: Adapting pretrained transformers for machine translation. In EMNLP, pages 1754–1765, 2021.

[17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In ICML, 2017.

[18] Francesco Di Giovanni, T. Konstantin Rusch, Michael M. Bronstein, Andreea Deac, Marc Lackenby, Siddhartha Mishra, and Petar Velickovic. How does over-squashing affect the power of gnns? Trans. Mach. Learn. Res., 2024.

[19] Maoguo Gong, Hui Zhou, A. K. Qin, Wenfeng Liu, and Zhongying Zhao. Self-paced co-training of graph neural networks for semi-supervised node classification. IEEE Transactions on Neural Networks and Learning Systems, 34(11):9234–9247, 2023.

[20] Arthur Gretton, Olivier Bousquet, Alexander J. Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In ALT, volume 3734, pages 63–77, 2005.

[21] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In NeurIPS, pages 1024–1034, 2017.

[22] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence, 45(1):87–110, 2022.

[23] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In NeurIPS, 2020.

[24] Zilong Huang, Xinggang Wang, Yunchao Wei, Lichao Huang, Humphrey Shi, Wenyu Liu, and Thomas S. Huang. Ccnet: Criss-cross attention for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 45(6):6896–6908, 2023.

[25] Md. Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In KDD, pages 655–665. ACM, 2022.

[26] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In ICML, volume 119, pages 5156–5165, 2020.

[27] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR, 2017.

[28] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR, 2019.

[29] Kezhi Kong, Jiuhai Chen, John Kirchenbauer, Renkun Ni, C. Bayan Bruss, and Tom Goldstein. GOAT: A global transformer on large-scale graphs. In ICML, volume 202, pages 17375–17390, 2023.

[30] Xiang Li, Chaofan Fu, Zhongying Zhao, Guangjie Zheng, Chao Huang, Yanwei Yu, and Junyu Dong. Dual-channel multiplex graph neural networks for recommendation. IEEE Transactions on Knowledge and Data Engineering, 37(6):3327–3341, 2025.

[31] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In ICML, volume 162, pages 13242–13256, 2022.

[32] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In NeurIPS, pages 20887–20902, 2021.

[33] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. AI Open, 3:111–132, 2022.

[34] Chuang Liu, Zelin Yao, Yibing Zhan, Xueqi Ma, Shirui Pan, and Wenbin Hu. Gradformer: Graph transformer with exponential decay. In IJCAI, pages 2171–2179, 2024.

[35] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In ICLR, 2020.

[36] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In ICLR, 2023.

[37] Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In NeurIPS, 2022.

[38] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. J. Complex Networks, 9(2), 2021.

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. nature, 323(6088):533–536, 1986.

[40] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. AI Mag., 29(3):93–106, 2008.

[41] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. CoRR, abs/1811.05868, 2018.

[42] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J. Sutherland, and Ali Kemal Sinop. Exphormer: Sparse transformers for graphs. In ICML, volume 202, pages 31613–31632. PMLR, 2023.

[43] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In SIGKDD, pages 807–816, 2009.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In NIPS, pages 5998–6008, 2017.

[45] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. CoRR, abs/1710.10903, 2017.

[46] Ulrike von Luxburg. A tutorial on spectral clustering. Stat. Comput., 17(4):395–416, 2007.

[47] Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In NeurIPS, 2022.

[48] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In NeurIPS, 2023.

[49] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In NeurIPS, pages 13266–13279, 2021.

[50] Zhihao Wu, Zhaoliang Chen, Shide Du, Sujia Huang, and Shiping Wang. Graph convolutional network with elastic topology. Pattern Recognition, 151:110364, 2024.

[51] Zhihao Wu, Xincan Lin, Zhenghong Lin, Zhaoliang Chen, Yang Bai, and Shiping Wang. Interpretable graph convolutional network for multi-view semi-supervised learning. IEEE Transactions on Multimedia, 25:8593–8606, 2023.

[52] Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: on the over-globalizing problem in graph transformers. In ICML, 2024.

[53] Liang Yang, Zhenna Li, Jiaming Zhuo, Jing Liu, Ziyi Ma, Chuan Wang, Zhen Wang, and Xiaochun Cao. Graph contrastive learning with joint spectral augmentation of attribute and topology. In AAAI, pages 21983–21991, 2025.

[54] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. Why do attributes propagate in graph convolutional neural networks? In AAAI, pages 4590–4598, 2021.

[55] Syed Waqas Zamir, Aditya Arora, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Restormer: Efficient transformer for high-resolution image restoration. In CVPR, pages 5718–5729, 2022.

[56] Zhongying Zhao, Zhan Yang, Chao Li, Qingtian Zeng, Weili Guan, and Mengchu Zhou. Dual feature interaction-based graph convolutional network. IEEE Transactions on Knowledge and Data Engineering, 35(9):9019–9030, 2023.

[57] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In WWW, pages 1215–1226, 2021.

[58] Shuman Zhuang, Zhihao Wu, Zhaoliang Chen, Hong-Ning Dai, and Ximeng Liu. Refine then classify: Robust graph neural networks with reliable neighborhood contrastive refinement. In AAAI, pages 13473–13482, 2025.

[59] Jiaming Zhuo, Can Cui, Kun Fu, Bingxin Niu, Dongxiao He, Yuanfang Guo, Zhen Wang, Chuan Wang, Xiaochun Cao, and Liang Yang. Propagation is all you need: A new framework for representation learning and classifier training on graphs. In MM, pages 481–489, 2023.

[60] Jiaming Zhuo, Can Cui, Kun Fu, Bingxin Niu, Dongxiao He, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Graph contrastive learning reimagined: Exploring universality. In WWW, pages 641–651, 2024.

[61] Jiaming Zhuo, Yuwei Liu, Yintong Lu, Ziyi Ma, Kun Fu, Chuan Wang, Yuanfang Guo, Zhen Wang, Xiaochun Cao, and Liang Yang. Dualformer: Dual graph transformer. In ICLR, 2025.

[62] Jiaming Zhuo, Feiyang Qin, Can Cui, Kun Fu, Bingxin Niu, Mengzhu Wang, Yuanfang Guo, Chuan Wang, Zhen Wang, Xiaochun Cao, and Liang Yang. Improving graph contrastive learning via adaptive positive sampling. In CVPR, pages 23179–23187, 2024.

A Algorithm Description

A layer of the proposed dual cross-attention module DCA is depicted in Algorithm 2.

Algorithm 2: PyTorch-style Code for DCA layer

```
# N: instance number
# D: hidden dimension
# z: data embeddings sized [N, D]
# b: data embeddings sized [N, D]
# H: head number
# Wq, Wk, Wv: parameter matrices for feature transformation
q = Wq(z) # [N, H, D]
k = Wk(z) # [N, H, D]
v = Wv(b) # [N, H, D]
# numerator
kv = torch.einsum("lhm, lhd → hmd", k, v)
num = torch.einsum("nhm, hmd → nhd", q, kv)
num += N * v # [N, H, D]
# denominator
all_ones = torch.ones(N)
k_sum = torch.einsum("lhm, l → hm", k, all_ones)
den = torch.einsum("nhm, hm → nh", q, k_sum) # [N, H]
# aggregated results
den += torch.ones_like(den) * N
output = num / den.unsqueeze(2) # [N, H, D]
# head average
output = output.mean(dim=1) # [N, D]
```

B Proof for Theorem 1

Proof. The proof unfolds in three stages: firstly, a concise description of the model to be validated is provided; subsequently, the model is decomposed into its topology and attribute components; and finally, it is verified that this structure aligns with the cross-aggregation form.

For a single-layer GNN block, such as GCN [27] widely used in GTs [37, 2, 48], the feature update can be expressed as

$$\mathbf{H} = \sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}). \tag{16}$$

where the diffusion matrix $\tilde{\mathbf{A}}$ denotes the normalized adjacency matrix.

By omitting the nonlinear activation function and performing eigendecomposition on the adjacency matrix, the above equation can be reformulated as

$$\begin{aligned} \mathbf{H} &= \tilde{\mathbf{A}}\mathbf{X}\mathbf{W} \\ &= \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top\mathbf{X}\mathbf{W} \\ &= \mathbf{U}\sqrt{\mathbf{\Lambda}}\sqrt{\mathbf{\Lambda}}\mathbf{U}^\top\mathbf{X}\mathbf{W}. \end{aligned} \tag{17}$$

Next, let $\mathbf{B} = \mathbf{U}\sqrt{\mathbf{\Lambda}}$ and $\mathbf{Z} = \mathbf{X}\mathbf{W}$. The similarity function $\mathrm{Sim}(\cdot,\cdot)$ is implemented as matrix multiplication. With the definitions in Eq. 5, the GCN block can be interpreted as a cross-aggregation from the attribute space to the topology space.

Extension to Multi-layer GNN Blocks. Following the discussion on single-layer GNNs, the solution for multi-layer GNNs is presented. Under the above assumptions, the node representations in the l-th layer can be formulated as

$$\begin{aligned} \mathbf{H}^l &= (\tilde{\mathbf{A}}^1\tilde{\mathbf{A}}^2\dots\tilde{\mathbf{A}}^l)\mathbf{X}(\mathbf{W}^1\mathbf{W}^2\dots\mathbf{W}^l) \\ &= \mathcal{S}\ \mathbf{X}\ \mathcal{W}, \end{aligned} \tag{18}$$

where $\tilde{\mathbf{A}}^i$ and $\mathbf{W}^i$ denote the diffusion matrix and the parameter matrix, respectively, in the i-th layer. $\mathcal{S} = \prod_{i=0}^{l} \mathbf{S}^l$ and $\mathcal{W} = \prod_{i=0}^{l} \mathbf{W}^l$ stand for the product of the diffusion matrices and the projection matrices. Given the properties of the diffusion matrix, the product diffusion matrix can be eigen-decomposed as $\mathcal{S} = \mathbf{U}\mathbf{\Lambda}^{(l)}\mathbf{U}^\top$, where $\mathbf{\Lambda}^{(l)}$ represents the l power of $\mathbf{\Lambda}$. Therefore, the above conclusion still holds in the context of multi-layer GNNs. □

Remark. This interpretation holds under the assumption that the diffusion operators across layers share a common eigenspace (*i.e.*, identical or mutually commutative $\tilde{\mathbf{A}}$ across layers). Otherwise, the equivalence serves as a first-order approximation of the aggregation process.

C Proof for Theorem 2

Proof. This proof first expands the model based on the feature and parameter matrices. Then, it identifies the representation updates of the topology and attributes within it. Finally, it establishes the relationship between these updated expressions and the cross-aggregation.

Firstly, the function $Softmax(\cdot)$ can be approximated using Random Features mappings, that is, $\mathbf{H} = Softmax(\mathbf{Q}\mathbf{K}^\top)\mathbf{V} \approx \phi(\mathbf{Q})\phi(\mathbf{K})^\top\mathbf{V}$. Here, $\phi(\cdot)$ denotes a kernel-based feature mapping that linearizes the attention computation. In practice, the learnable projection matrix $\mathbf{W}$ applied to $\mathbf{X}$ can be viewed as a parametric approximation to this mapping.

By expanding node attributes $\mathbf{X} = [\mathbf{X}; \mathbf{P}]$, where $\mathbf{X} \in \mathbb{R}^{n\times f}$ and $\mathbf{P} \in \mathbb{R}^{n\times k}$, it can be obtained as

$$\mathbf{H} = \left([\mathbf{X}; \mathbf{P}]\mathbf{W}^q(\mathbf{W}^k)^\top \begin{bmatrix} \mathbf{X}^\top \\ \mathbf{P}^\top \end{bmatrix}\right)[\mathbf{X}; \mathbf{P}]\mathbf{W}^v \tag{19}$$

Then, by expanding the Query (like $\mathbf{W}^q = \begin{bmatrix} \mathbf{W}_1^q \\ \mathbf{W}_2^q \end{bmatrix}$), and the Key and Value, it can be derived as

$$\begin{aligned}
\mathbf{H} &= \left((\mathbf{X}\mathbf{W}_1^q + \mathbf{P}\mathbf{W}_2^q)\left((\mathbf{W}_1^k)^\top\mathbf{X}^\top + (\mathbf{W}_2^k)^\top\mathbf{P}^\top\right)\right)(\mathbf{X}\mathbf{W}_1^v + \mathbf{P}\mathbf{W}_2^v) \\
&= \left((\mathbf{X}\mathbf{W}_1^q + \mathbf{P}\mathbf{W}_2^q)\left((\mathbf{W}_1^k)^\top\mathbf{X}^\top + (\mathbf{W}_2^k)^\top\mathbf{P}^\top\right)\right)\mathbf{X}\mathbf{W}_1^v \\
&+ \left((\mathbf{X}\mathbf{W}_1^q + \mathbf{P}\mathbf{W}_2^q)\left((\mathbf{W}_1^k)^\top\mathbf{X}^\top + (\mathbf{W}_2^k)^\top\mathbf{P}^\top\right)\right)\mathbf{P}\mathbf{W}_2^v.
\end{aligned} \tag{20}$$

It is evident that the equation includes several terms that describe the self-aggregation of topology and attributes. For instance, $\mathbf{P}\mathbf{W}_2^q(\mathbf{P}\mathbf{W}_2^k)^\top\mathbf{P}\mathbf{W}_2^v$ and $\mathbf{X}\mathbf{W}_1^q(\mathbf{X}\mathbf{W}_1^k)^\top\mathbf{X}\mathbf{W}_1^v$ stand for the self-aggregation of topology and attributes, respectively. Furthermore, this equation contains several terms that include cross-aggregation across topology and attributes. Exampled by the term $\mathbf{X}\mathbf{W}_1^q(\mathbf{P}\mathbf{W}_2^k)^\top\mathbf{P}\mathbf{W}_2^v$, by setting $\mathbf{Z} = \mathbf{X}\mathbf{W}_1^q$ and $\mathbf{B} = \mathbf{P}\mathbf{W}_2$, the cross-aggregation from the topology space to the attribute space can be obtained. Similarly, terms (*e.g.*, $\mathbf{P}\mathbf{W}_2^q(\mathbf{P}\mathbf{W}_2^k)^\top\mathbf{X}\mathbf{W}_1^v$) describing cross-aggregation from the attribute space to the topology space can be found. □

D Proof for Theorem 3

Proof. This proof includes two main steps. First, we plan to derive the closed-form solutions for $\mathbf{Z}$ and $\mathbf{B}$ from the convex optimization objective (Eq. 15). These solutions are denoted as $\mathbf{Z}^*$ and $\mathbf{B}^*$, respectively. Second, we aim to establish the equivalence between these derived solutions $\mathbf{Z}^*$ and $\mathbf{B}^*$ and the feature updates in Eq. 10 and Eq. 11, respectively.

Let us denote the objective function (Eq. 15) as $\mathcal{O}(\mathbf{Z}, \mathbf{B})$, that is

$$\begin{aligned}
\mathcal{O}(\mathbf{Z}, \mathbf{B}) &= \lambda\,\mathrm{Tr}(\mathbf{Z}^\top\tilde{\mathbf{L}}\mathbf{Z}) + \|\mathbf{B} - MLP(\mathbf{X})\|_F^2 - \eta\|\mathbf{Z}^\top\mathbf{B}\|_F^2 \\
&= \lambda\,\mathrm{Tr}(\mathbf{Z}^\top(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{Z}) \\
&+ \mathrm{Tr}\left((\mathbf{B} - MLP(\mathbf{X}))^\top(\mathbf{B} - MLP(\mathbf{X}))\right) \\
&- \eta\,\mathrm{Tr}((\mathbf{Z}^\top\mathbf{B})^\top(\mathbf{Z}^\top\mathbf{B}))
\end{aligned} \tag{21}$$

Firstly, the partial derivatives of $\mathcal{O}(\mathbf{Z}, \mathbf{B})$ with respect to $\mathbf{Z}$ can be calculated as

$$\frac{\partial\mathcal{O}(\mathbf{Z}, \mathbf{B})}{\partial\mathbf{Z}} = 2\lambda(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{Z} - 2\eta(\mathbf{B}\mathbf{B}^\top\mathbf{Z}) \tag{22}$$

The closed-form solution $\mathbf{Z}^*$ for the objective function $\mathcal{O}$ can be derived by setting $\frac{\partial \mathcal{O}(\mathbf{Z},\mathbf{B})}{\partial \mathbf{Z}} = 0$, that is

$$2\lambda(\mathbf{I} - \tilde{\mathbf{A}})\mathbf{Z} - 2\eta(\mathbf{B}\mathbf{B}^\top\mathbf{Z}) = 0 \tag{23}$$

$$\Rightarrow \mathbf{Z}^* = \tilde{\mathbf{A}}\mathbf{Z} + \frac{\eta}{\lambda}\mathbf{B}\mathbf{B}^\top\mathbf{Z} \tag{24}$$

Then, by setting $\omega_1 = \frac{\eta}{\lambda+\eta}$, we obtain

$$\mathbf{Z}^* = (1-\omega_1)\tilde{\mathbf{A}}\mathbf{Z} + \omega_1\mathbf{B}\mathbf{B}^\top\mathbf{Z} \tag{25}$$

For the update of topology representations in the proposed DCA module, *i.e.*, Eq. 10, the process can be rewritten as

$$\hat{\mathbf{Z}} = (1-\tilde{\lambda})\tilde{\mathbf{A}}\mathbf{V} + \tilde{\lambda}\mathbf{S}\mathbf{V}, \tag{26}$$

where $\mathbf{S} \in \mathbb{R}^{n\times n}$ stands for the cross-attention score matrix, with the scores $s_{i,i} = \frac{n+\mathbf{q}_{i,:}\mathbf{k}_{i,:}^\top}{n+\sum_t \mathbf{q}_{i,:}\mathbf{k}_{t,:}^\top}$ and $s_{i,j} = \frac{\mathbf{q}_{i,:}\mathbf{k}_{j,:}^\top}{n+\sum_t \mathbf{q}_{i,:}\mathbf{k}_{t,:}^\top}$ for $i \neq j$.

To demonstrate the equivalence between Eq. 25 and Eq. 26, the first step is to set $\omega_1 = \tilde{\lambda}$. With this condition met, the required proof is to derive a matrix $\mathbf{B}$ that satisfies the equation $\mathbf{B}\mathbf{B}^\top = \mathbf{S}$.

Let us denote the diagonal matrix as $\mathbf{D}$, where $d_{i,i} = \sqrt{\frac{n+1}{n+\sum_t \mathbf{q}_{i,:}\mathbf{q}_{t,:}^\top}}$, the construction of $\mathbf{B} = \mathbf{D}\mathbf{Q}$ ensures that the above equation holds.

For the diagonal elements of $\mathbf{B}\mathbf{B}^\top$, there is

$$(\mathbf{B}\mathbf{B}^\top)_{i,i} = \sum_{j=0}^{d-1} b_{i,j}^2 = \sum_{j=0}^{d-1} (d_{i,i}\cdot q_{i,j})^2 = d_{i,i}^2 \sum_{j=0}^{d-1} q_{i,j}^2 \tag{27}$$

Given that the rows of $\mathbf{Q}$ are L2-normalized, we have $\sum_{j=0}^{d-1} q_{i,j}^2 = 1$. Due to the same source and parameter sharing, it follows that $\mathbf{Q} = \mathbf{K}$. Thus, we obtain $(\mathbf{B}\mathbf{B}^\top)_{i,i} = d_{i,i}^2 = \frac{n+1}{n+\sum_t \mathbf{q}_{i,:}\mathbf{q}_{t,:}^\top}$, which matches the definition of the score $s_{v,v}$ on the main diagonal.

For the off-diagonal elements of $\mathbf{B}\mathbf{B}^\top$ where $v \neq u$, there is

$$\begin{aligned}(\mathbf{B}\mathbf{B}^\top)_{i,j} &= \sum_{t=0}^{d-1} b_{i,t}\cdot b_{j,t} = \sum_{t=0}^{d-1} (d_{i,i}\cdot q_{i,t})\,(d_{j,j}\cdot q_{j,t}) \\ &= d_{i,i}\cdot d_{j,j}\sum_{t=0}^{d-1} q_{i,t}\cdot q_{j,t} = c\frac{\mathbf{q}_{i,:}\mathbf{q}_{j,:}^\top}{n+\sum_k \mathbf{q}_{i,:}\mathbf{q}_{k,:}^\top}\end{aligned} \tag{28}$$

Since $d_{i,i}$ and $d_{j,j}$ are the square roots of the denominators in the formula for $s_{i,i}$ and $s_{j,j}$, respectively, and $\mathbf{q}_{i,:}\mathbf{q}_{j,:}^\top$ is the dot product of the i-th and j-th rows of $\mathbf{Q}$, this matches the definition of $s_{i,j}$. Therefore, the correct construction of $\mathbf{B}$ should be $\mathbf{B} = \mathbf{D}\mathbf{Q}$.

Similarly, the closed-form solution $\mathbf{B}^*$ of the objective in Eq. 15 can be obtained by setting its derivative to 0 as

$$\frac{\partial \mathcal{O}(\mathbf{Z},\mathbf{B})}{\partial \mathbf{B}} = 2(\mathbf{B} - MLP(\mathbf{X})) + 2\eta(\mathbf{Z}\mathbf{Z}^\top\mathbf{B}) = 0 \tag{29}$$

$$\Rightarrow \mathbf{B}^* = MLP(\mathbf{X}) + \eta\mathbf{Z}\mathbf{Z}^\top\mathbf{B} \tag{30}$$

Then, by defining $\omega_2 = \frac{1}{1+\eta}$, there is

$$\mathbf{B}^* = (1-\omega_2)MLP(\mathbf{X}) + \omega_2\mathbf{Z}\mathbf{Z}^\top\mathbf{B} \tag{31}$$

The proposed dual cross-attention module for updating the attribute representations, *i.e.*, Eq. 11, can be rephrased as

$$\hat{\mathbf{B}} = (1-\tilde{\gamma})MLP(\mathbf{X}) + \tilde{\gamma}\mathbf{S}\mathbf{V} \tag{32}$$

Similarly, considering that matrix $\mathbf{S}$ maintains the same structure as described in Eq. 26, the crucial step to ensure $\mathbf{B}^* = \hat{\mathbf{B}}$ is to establish the parameter $\omega_2 = \tilde{\gamma}$ and to set $\mathbf{Z} = \mathbf{D}\mathbf{Q}$. This approach guarantees that the necessary conditions for the equivalence are met. □

E Experimental Details

E.1 Datasets and Splitting

Datasets. In the node classification experiments, sixteen publicly available benchmark datasets are utilized. These graphs can be classified into two categories based on whether their Edge Homophily [35] exceeds 0.5: seven graphs are tagged as *homophilic graphs*, including Cora [40], CiteSeer [40], PubMed [40], Photo [41], CS [41], Physics [41], and Questions [36]. The remaining seven graphs are marked as *heterophilic graphs*, containing Cornell [35], Texas [35], Wisconsin [35], Actor [43], Chameleon [38], Squirrel [38], and Ratings [36]. It is worth noting that the original Chameleon and Squirrel exhibit neighborhood overlap, and are thus filtered according to the study [36]. Additionally, two large-scale graph datasets, *i.e.*, ogbn-arxiv [23] and ogbn-proteins [23], are employed for node property prediction experiment. Statistics are shown in Tab. 4.

Table 4: Statistics of sixteen graph datasets. $\#h$ denotes the edge homophily shown in [35].

Dataset	Nodes	Edges	Features	Classes	$\#h$
Cora	2,708	5,278	1,433	7	0.81
CiteSeer	3,327	4,552	3,703	6	0.74
PubMed	19,717	44,324	500	3	0.80
Photo	7,650	238,163	745	8	0.83
CS	18,333	81,894	6,805	15	0.81
Physics	34,493	247,962	8,415	5	0.93
Questions	48,921	153,540	301	2	0.84
Cornell	183	280	1,703	5	0.30
Texas	183	295	1,703	5	0.11
Wisconsin	251	466	1,703	5	0.21
Actor	7,600	33,544	931	5	0.22
Chameleon	890	8,854	2,325	5	0.24
Squirrel	2,223	46,998	2,089	5	0.21
Ratings	24,492	93,050	300	5	0.38
ogbn-proteins	132,534	39,561,252	8	2	0.38
ogbn-arxiv	169,343	1,157,799	300	128	0.65

Dataset Splitting. To ensure that the experimental results are credible and reproducible, this paper follows well-established dataset splitting strategies. For the Cora, CiteSeer, and PubMed, the public standard splitting described in [27] is adopted, with 20 nodes per class for training, 500 for validation, and 1000 for testing. The Photo, CS, and Physics are randomly divided into training, validation, and testing sets in a 60%, 20%, and 20% ratio, respectively. For the heterophilic datasets Cornell, Texas, Wisconsin, Actor, and Chameleon, this paper employ 10 standard train/validation/test splits with a division ratio of 48%, 32%, and 20%, respectively. Note that the Chameleon and Squirrel used here are duplicates-removed filtered versions as referenced in [36]. The Ratings, and Questions follow a $50\%/25\%/25\%$ train/validation/test random split pattern. For the two datasets from the OGB [23], *i.e.*, ogbn-arxiv and ogbn-proteins, the provided standard splits are utilized.

E.2 Introduction of Baselines

The comparative analysis in the experiments involves seven Graph Neural Networks (GNNs) as well as seven Graph Transformers (GTs) as the baseline models. To be specific, the GNNs include four *standard GNNs*, *i.e.*, GCN [27], GAT [45], GraphSAGE [21], and APPNP [28], and two *universal GNNs* for graphs with diverse homophily, *i.e.*, GPR-GNN [6] and GloGNN [31], and a *non-message-passing GNN* with a separate topology and attribute design, *i.e.*, LINKX [32]. Besides, the baseline GTs encompass eight state-of-the art models, namely, GraphGPS [37], NAGphormer [2], Exphormer [42], GOAT [29], NodeFormer [47], SGFormer [48], Polynormer [7], and Gradformer [34]. These models are implemented following the released code of the original paper.

E.2.1 Graph Neural Networks (GNNs)

The following specifies the GNN baselines employed in our comparative analysis.

- GCN [27]: A seminal GNN that integrates graph topology and node attributes via graph convolution.
- GAT [45]: A classic graph attention network that weights propagation using an attention mechanism.
- GraphSAGE [21]: A scalable variant of GCN that employs neighbor sampling and diverse aggregation strategies.
- APPNP [28]: A variant of GCN that weights propagation based on personalized PageRank.
- GPR-GNN [6]: A universal variant of GCN that weights propagation using learnable layer coefficients.
- LINKX [32]: A non-message-passing GNN that directly combines representations of graph topology and node attributes.
- GloGNN [31]: A universal variant of GCN that obtains the propagation matrix from an optimization objective describing node relationships.

E.2.2 Graph Transformers (GTs)

The following specifies the GT baselines utilized in our comparative analysis.

- GraphGPS [37]: A general GT architecture incorporating positional encodings and local/global modules.
- NodeFormer [47]: A scalable GT architecture that learns layer-specific graph structures via a kernelized Gumbel-Softmax operator.
- NAGphormer [2]: A creative GT architecture constructing token vectors using neighborhood aggregation.
- Exphormer [42]: A general GT architecture combining local, extended, and virtual node-based global attention.
- GOAT [29]: A comprehensive GT architecture linearizing computational complexity based on the k-means algorithm.
- SGFormer [48]: A lightweight GT architecture featuring a single-layer self-attention module.
- Polynormer [7]: A polynomial-expressive GT architecture learning high-degree polynomials on input features.
- Gradformer [34]: An effective GT architecture dynamically modeling node relationships by exponentially diminishing values in the decay mask matrix.

For the four GNN baselines, including GCN, GAT, GraphSAGE, and APPNP, we utilize the public library, PyTorch Geometric (PyG) [13], for their implementation. For the other three GNN baselines, we utilize their original code. The sources are outlined as

- GPR-GNN: https://github.com/jianhao2016/GPRGNN
- LINKX: https://github.com/CUAI/Non-Homophily-Large-Scale
- GloGNN: https://github.com/RecklessRonan/GloGNN

For the GT baselines, including GraphGPS, NodeFormer, NAGphormer, Exphormer, GOAT, SGFormer, Polynormer, and Gradformer, we utilize their source code. The sources are detailed as

- GraphGPS: https://github.com/rampasek/GraphGPS
- NodeFormer: https://github.com/qitianwu/NodeFormer
- NAGphormer: https://github.com/JHL-HUST/NAGphormer
- Exphormer: https://github.com/hamed1375/Exphormer
- GOAT: https://github.com/devnkong/GOAT
- SGFormer: https://github.com/qitianwu/SGFormer
- Polynormer: https://github.com/cornell-zhang/Polynormer
- Gradformer: https://github.com/LiuChuang0059/Gradformer

E.3 Experimental Setups

Configurations. The experiment is performed on two Linux machines using a single GeForce RTX4090 24 GB GPU and a single NVIDIA A800 80GB GPU, respectively. The reported results are averaged over ten random trials. All models operate under a semi-supervised learning paradigm, where the results on validation sets are referenced to fine-tune hyperparameters.

Table 5: Hyperparameters of UGCFormer per dataset.

Dataset	# layers l	# dimensions d	lr	α	β	wd
Cora	4	256	0.001	0.5	1	5e-3
CiteSeer	3	256	0.001	0.6	0.1	5e-3
PubMed	3	128	0.001	0.5	0.1	1e-2
Photo	4	512	0.001	0.3	0.1	5e-5
CS	3	512	0.001	0.6	0.1	5e-4
Physics	3	512	0.001	0.5	1	5e-4
Questions	2	256	0.005	0.3	0.1	5e-4
Cornell	5	256	0.001	0.9	0.001	1e-2
Texas	4	128	0.001	0.7	0.001	1e-2
Wisconsin	4	128	0.001	0.9	0.001	1e-2
Actor	5	512	0.001	0.9	0.01	1e-2
Chameleon	2	512	0.001	0.2	0.01	5e-3
Squirrel	4	512	0.001	0.2	0.01	5e-5
Ratings	2	64	0.001	0.3	0.001	0
ogbn-proteins	1	128	0.001	0.7	0.01	5e-4
ogbn-arxiv	2	512	0.01	0.3	0.01	5e-4

Hyper-parameters. The hyperparameters are selected via a grid search strategy. In the node classification task, models are trained employing an Adam optimizer with the learning rate among $\{0.001, 0.005, 0.01\}$ and the weight decay among $\{0, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2\}$. The number of layers is selected from $\{1, 2, 3, 4, 5\}$, and the dimension of hidden layers is chosen from $\{64, 128, 256, 512\}$, and their impacts on model performance are analyzed in Section 4.2. For the node property prediction task, the hyperparameter selection follows the baseline [48]. For the unique hyperparameters in UGCFormer, α is chosen from a range starting at 0.1 and increasing by increments of 0.1, up to 0.9, β is selected from $\{0.001, 0.01, 0.1, 1\}$, and τ is fixed to 0.5. Refer to Tab. 5 for the chosen parameters that correspond to the reported results.

Table 6: Training time and GPU memory usage on three graphs.

	CiteSeer		PubMed		ogbn-arxiv	
Method	Train/Epoch (ms)	Mem. (MB)	Train/Epoch (ms)	Mem. (MB)	Train/Epoch (ms)	Mem. (MB)
GraphGPS	16.82	140	46.99	470	166.10	8,102
NodeFormer	10.20	110	11.14	218	84.00	2,066
NAGphormer	10.81	166	17.65	352	760.50	1,962
Exphormer	14.20	159	25.00	696	145.70	6,758
SGFormer	5.80	84	6.07	142	36.90	1,024
Polynormer	9.20	174	15.20	307	170.07	4,729
UGCFormer	8.80	106	10.60	246	69.60	2,050

E.4 Additional experiment results

Running Time and Space Consumption. To further illustrate the efficiency and scalability of the proposed UGCFormer, this experiment compares it with other GTs in terms of runtime and GPU memory usage. Common hyperparameters are uniformly applied across all models to highlight the impact of their components, particularly the attention modules. As depicted in Table 6, UGCFormer consistently has the second-lowest running time and ranks among the top three in terms of lowest GPU memory usage across three datasets. Despite utilizing linearized attention mechanisms, most linearized GTs, including GraphGPS, NodeFormer, Exphormer, and Polynormer, perform worse than UGCFormer. This highlights the lightweight and efficient design of UGCFormer.

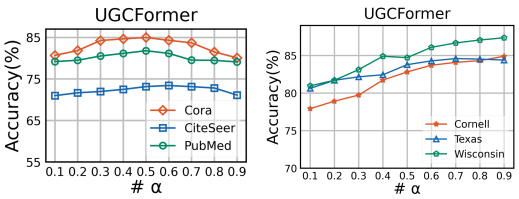


Figure 6: Performance variations for varying α.

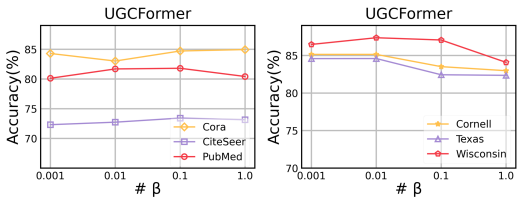


Figure 7: Performance variations for varying β.

Hyperparameter α. As depicted in the Fig. 6, UGCFormer exhibits stable performance within a specific parameter range for each dataset. For instance, on Cora, performance variation is minimal within the set $\{0.3, 0.4, 0.5, 0.6\}$, indicating that the model is robust to changes in hyperparameter α. Similar observations are made for heterophilic graphs.

Hyperparameter β. From the Fig. 7, it can be observed that the performance remains stable across the selection range of β, demonstrating that the model is not sensitive to this parameter. Even in the worst case of parameter selection, the model achieves performance that is comparable to the baseline.

F Discussion

Comparison with Edge-augmented Graph Transformer (EGT). The proposed UGCFormer shares a conceptual connection with EGT [25], yet their core mechanisms differ substantially. EGT augments pairwise attribute-based attention between nodes using graph topology, whereas UGCFormer captures the interaction between topology and attributes through a cross-attention mechanism. Although EGT introduces an additional edge channel alongside the attribute channel, this design primarily aims to utilize edge features to modulate the attention process (via addition or gating) rather than to explicitly model the interaction between the two channels. Moreover, EGT requires large-scale edge features of size $O(n^2d)$ (where n denotes the number of nodes and d the feature dimension), which significantly increases computational complexity and limits scalability. In contrast, UGCFormer adopts a linear cross-attention module that efficiently models topology-attribute interactions with linear time and space complexity.

Broader Relation to Other Categories of Graph Transformers. The proposed cross-aggregation mechanism can also provide a unified interpretation for other two types of Graph Transformers, that is, *those based on context-node sampling or edge rewriting*. Although both categories of GTs are implemented in different ways, their essence is to determine the subgraph for each node to perform local message passing (via graph convolution or self-attention). Thus, they can be uniformly expressed by obtaining the graph structure. As a result, the topology representation can be straightforwardly determined based on the eigenvalue decomposition of the adjacency matrix, as described in the manuscript. Ultimately, a formulation similar to that of cross-aggregation can be derived. Therefore, their underlying mechanism can be attributed to a cross-aggregation between topology and attribute representations.

G Limitations

The proposed UGCFormer, like other Graph Transformers (GTs), relies on a fixed set of hyperparameters, requiring significant tuning and prior knowledge to optimize results. This not only limits their applicability but also increases computational costs. Future work should explore adaptive learning mechanisms to automate hyperparameter adjustment based on input data characteristics, reducing manual intervention and enhancing generalizability.

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: The abstract and introduction clearly outline the contributions of our paper, including the motivation and design of the proposed UGCFormer.

   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 the limitations in Section G, particularly regarding the large number of hyperparameters commonly found in GTs. We have outlined potential directions for future research to address this concern.

   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: [Yes]

Justification: All theoretical results are accompanied by clearly stated assumptions and complete proofs, provided in the main paper and referenced appropriately.

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: The supplemental material contains a file of our model's code, enabling the replication of the experiments.

   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 included complete and executable code within the supplemental material, ensuring the reproducibility of our results.

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 detailed descriptions of our experimental setup in Section E, including data splits, hyperparameters, and optimizer, etc.

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: The experimental results are presented as the mean and standard deviation over 10 runs, as shown in Tabs. 1 and 2, as well as Figs. 4 and 5.

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: We have provided the computational resources used for all experiments in Section E.3.

   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 research adheres to the NeurIPS Code of Ethics, and we have ensured that all aspects of our work.

   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: Due to the nature of this work, there may be no potential negative social impact that is easily predictable.

    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: Our work does not involve releasing data or models that pose a high risk for misuse, so no specific safeguards are necessary.

    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 have accurately credited the sources and provided URLs in Section E.2.

    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: [Yes]

    Justification: The supplemental material includes the file of our model's code

    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 does not involve crowdsourcing or research with human subjects, so this information is not applicable.

    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 research does not involve human subjects, so IRB approvals or equivalent reviews are not required.

    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.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.