Towards Anomaly Detection on Text-Attributed Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Graph anomaly detection (GAD), which aims to identify abnormal nodes that differ from the majority in graphs, has attracted considerable research attention. In real-world GAD scenarios, such as reviews in e-commerce platforms, the original features in graphs are raw text. Existing methods only treat these texts with a simple context embedding, without a comprehensive understanding of semantic information. In this work, we propose TAGAD, a novel Text-Attributed Graph Anomaly Detection framework that jointly trains the context feature and the semantic feature of texts with graph structure to detect the anomaly nodes. TAGAD consists of a global GAD module and a local GAD module, respectively for detecting global anomaly nodes and local anomaly nodes. In the global GAD module, we employ a contrastive learning strategy to jointly train the graph-text model and an autoencoder to compute the global anomaly scores. In the local GAD module, an ego graph and a text graph are constructed for each node. Then, we devise two different methods to compute local anomaly scores based on the difference between the two subgraphs, respectively for the zero-shot settings and the few-shot settings. Extensive experiments demonstrate the effectiveness of our model under both zero-shot and few-shot settings on text-attributed GAD scenarios. Codes are available at `https://anonymous.4open.science/r/TAGAD-1223`.

1 Introduction

Graph anomaly detection (GAD) aims to identify abnormal nodes that exhibit significant deviation from the majority in the graph, which has attracted much interest due to its wide applications, such as financial fraud detection Huang et al. (2022), anti-money-laundering Weber et al. (2019), and review management Dou et al. (2020). In real-world scenarios, node labeling is often costly, making the low-resource GAD, where there are few or no labeled nodes, a critical and challenging research problem.

In the GAD lecture, nodes often carry rich textual information, such as the identification of fraudulent reviews on platforms like Amazon. To address anomaly detection on such text-attributed graphs (TAGs), both the context features capturing the statistical properties of texts and the semantic features inflecting the deep linguistic meaning are critical to detect the anomaly nodes. Therefore, it is essential to design a model that jointly learns contextual features, semantic features, and the graph structure.

However, existing GAD methods handle textual features in a simplistic way. Simple bag-of-words (BOW) representations Sennrich et al. (2016) or shallow embedding vectors Mikolov et al. (2013) are fed into GAD models as node features. While these techniques enable basic handling of textual data, they fail to capture its full semantic and contextual richness.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Recent works on Text-Attributed Graphs (TAGs) Yan et al. (2023) have explored joint training of the graph structure and the text embedding for the node classification task. They categorize nodes with similar text features and similar neighbors into one class. Some of these methods, like G2P2 Wen and Fang (2023) and P2TAG Zhao et al. (2024), utilize the text of the class due to the high similarity between the text feature and the text of the class. However, in the GAD problem, anomalous nodes often exhibit diverse and irregular textual and structural patterns, making them difficult to classify based on similarity. Moreover, it is meaningless to compute the similarity between the node feature and the text of the class, "anomalous" or "normal". Consequently, existing TAG-based methods developed for node classification cannot be applied to the GAD problem.

There are two main challenges on TAGs towards the anomaly detection problem. (1) Joint training of the graph-text model. While some recent works explore joint training of the graph-text model for tasks like node classification, they are not designed to detect anomaly nodes and thus cannot directly address the requirements of GAD. (2) Detection of both global and local anomaly nodes. There are both global and local anomaly nodes in GAD problem. Global anomaly nodes are those whose features deviate from the majority of the nodes, while local anomaly nodes exhibit abnormal features within their immediate neighborhood or subgraph. Thus, a key challenge is how to detect both the global and local anomaly nodes.

In this paper, we propose a **T**ext-**A**ttributed **G**raph **A**nomaly **D**etection model called TAGAD, which jointly trains the context feature and the semantic feature of texts with the graph structure to find both global and local anomaly nodes. Two modules are composed in TAGAD: a global GAD module and a local GAD module, designed to identify global and local anomaly nodes, respectively. In the global GAD module, our model first obtains the semantic embedding by LM and the context graph feature by BOW and GNN, then aligns the GNN and the LM using a contrastive learning based loss function. Then, the autoencoder based technique is employed to find the anomaly nodes. In the local GAD module, two subgraphs are constructed for each node: the ego graph capturing the local graph structure and the text graph indicating the similarity of the semantic embedding between neighboring nodes. Then, we devise two different methods to compute the local anomaly scores, respectively for zero-shot settings and few-shot settings. Under zero-shot settings, the difference between the ego graph and the text graph is computed as the local anomaly score. However, due to the globally shared feature of nodes, textual similarities are uniformly high, thereby hiding some local anomaly nodes. In few-shot settings, we introduce a common embedding that captures the common feature of nodes. By removing this common feature, the similarity between anomalous and normal nodes is reduced, amplifying local deviations and improving the model's ability to detect local anomaly nodes.

Accordingly, our main contributions can be summarized as follows:

1. To the best of our knowledge, this is the first attempt towards anomaly detection problem on the text-attributed graphs.
2. We propose a novel framework TAGAD, that jointly trains context and semantic features of text with the graph structure.
3. We design two GAD methods based on comparing each node's ego graph with its corresponding text graph, respectively for the zero-shot settings and few-shot settings.
4. Our proposed TAGAD archives an improvement with $+7.8\% \sim +36.9\%$ compared to GAD methods under low-resource settings.

2 Related Work

2.1 Graph Anomaly Detection

Existing GAD methods are divided into two groups based on different settings: supervised and unsupervised. Under the supervised setting, GAD is formulated as a binary classification task. Various GNN-based supervised detectors have been devised in the lecture Tang et al. (2024), such as BWGNN Tang et al. (2022), AMNet Chai et al. (2022), PC-GNN Liu et al. (2021a), H2FDetector Liu et al. (2020).

Apart from these supervised detectors, there are numerous unsupervised GAD techniques Liu et al. (2022) aiming to detect anomalies without labeled data. As a typical approach in unsupervised graph learning, Graph Auto-Encoder (GAE) has been widely used in the GAD models. For example, DOMINANT Ding et al. (2019) uses GCN to reconstruct graph data of both topological structure and node attributes. ANOMALYDAE Fan et al. (2020) employs the attention mechanism to learn the importance between a node and its neighbors. There are also many methods using contrastive learning to compute the anomaly score, such as CONAD Liu et al. (2021b), COLA Liu et al. (2021b), and NLGAD Duan et al. (2023). Others like SCAN Roy et al. (2024), RADAR Li et al. (2017), and ANOMALOUS Peng et al. (2018) identify the anomaly nodes by using traditional shallow methods.

However, all these methods overlook the textual information associated with nodes in graphs, only relying on node attributes. To the best of our knowledge, this paper is the first work to explore graph anomaly detection towards text-attributed graphs.

2.2 Graph Pre-training and Prompt Learning

Recently, there has been a boom in the research of graph pre-training Jin et al. (2020), which aims to learn the general knowledge of the graphs. Numerous effective graph pre-training models have been introduced in this area. Among these models, GCA Zhu et al. (2021) adopts the node-level comparison method, while GraphCL You et al. (2020) and SimGRACE Xia et al. (2022) focus on the graph-level contrastive learning.

With the increasing interest in the large language model (LLM), utilizing node texts in graphs has gained growing attention. Many works incorporate pre-trained language models (PLMs), such as BERT Devlin (2018), into graph learning by leveraging node texts. Most of these works follow the paradigm of pre-training and prompt learning. For example, Prog Sun et al. (2023) unifies the graph prompt and language prompts. G2P2 Wen and Fang (2023) pretrains a Graph-Text model by aligning the graph structure with the corresponding text representation. In the prompt learning phase, the label texts are used to generate the prompt and jointly train the pre-trained Graph-LLM model. Similarly, P2TAG Zhao et al. (2024) introduces a language masking strategy for pretraining and utilizes both the label texts and the node texts to build a prompt graph. Nevertheless, these methods can't be applied to graph anomaly detection problems, as anomaly nodes vary significantly across different domains.

3 Preliminaries

In this section, we introduce the background of our paper including the definition of text-attributed graph and the text-attributed graph anomaly detection problem.

Definition 1 (Text-Attributed Graph) *A text-attributed graph (TAG) is a graph $G = (V, E, D)$, where each node $u \in V$ is associated with a text sequence $d_u \in D$ and E represents the set of edges between nodes.*

In graph anomaly detection, each node has a label $y_v \in \{0, 1\}$, where 0 represents normal and 1 represents anomaly. V_n and V_a represent the normal node set and anomaly node set, respectively. We denote Y as the labels assigned to the nodes. The whole graph contains two types of nodes, the training nodes V_{train} and the testing nodes V_{test}, labeled with Y_{train}, and Y_{test}. Y_{test} are inaccessible during the training.

Given the above definition, we formally define our problem, text-attributed graph anomaly detection.

Definition 2 (Text-Attributed Graph Anomaly Detection) *Given a text-attributed graph $G = (V, E, D)$, the observed nodes V_{train} with label Y_{train}, the Text-Attributed Graph Anomaly Detection problem aims to learn a function f that measures node abnormalities by calculating their anomaly scores S:*

$$f(G, Y_{train}) \to S, \tag{1}$$

where $S \in \mathbb{R}^n$ indicates the anomaly score matrix, and $n = \mid V \mid$ is the node number in the graph.

Low-resource Graph Anomaly Detection. In the low-resource lecture, the number of Y_{train} is small or even zero. In the K-shot graph anomaly detection problem, the number of anomaly nodes and normal nodes is K. As a special case, the problem with $K = 0$ is known as zero-shot classification, which means that there are no labeled nodes.

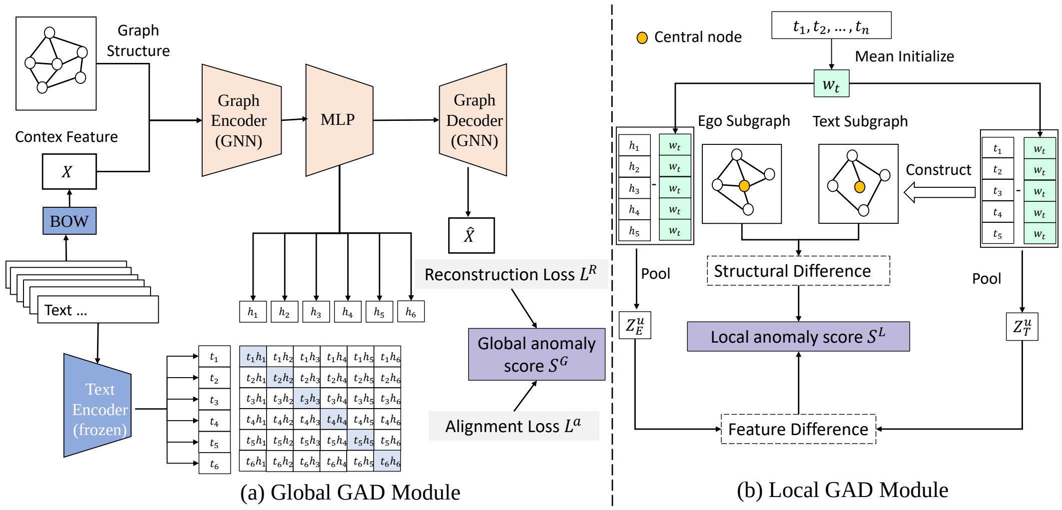


Figure 1: Our proposed framework TAGAD. (a) We first align the GNNs and the LM using a contrastive learning based objective. Then, the GNN decoder is introduced to compute the global anomaly scores. (b) Next, the common embedding is initialized as the mean embedding of all semantic embeddings. The local semantic embeddings are then obtained by subtracting the semantic embedding. Then, for each node, the ego graph is built based on the graph structure, while the text graph is formed by computing the similarity of the local semantic embedding. The local anomaly score is finally computed by comparing the two subgraphs. The figure only shows the local anomaly score under few-shot settings, while zero-shot inference adopts a simplified scheme.

4 Method

As shown in Figure 1, our TAGAD model consists of two modules: (a) Global GAD module, which aligns the GNNs and the LM using a contrastive learning based objective and calculates the global anomaly scores by the autoencoder. (b) Local GAD module, which computes the local anomaly scores by comparing the ego graph and the text graph of each node. The pseudocode of the algorithms and complexity analysis of TAGAD can be found in Appendix A.

4.1 Global GAD module

In this part, we introduce our proposed global GAD module in detail. The goal of the global GAD module is to detect the anomaly nodes that deviate from the major distribution. We first introduce the triple encoders to encode the context embedding by BOW, the semantic embedding by LM, and the graph structure by GNNs. However, GNNs are randomly initialized, not directly suitable for detecting global anomaly nodes, and the graph embedding space is different from the semantic embedding space. Therefore, we divide the global GAD module into two stages. First, we align the GNNs and LM embedding spaces using the contrastive learning based strategy. Then, we use an autoencoder based approach to detect global anomaly nodes.

4.1.1 Triple Encoders

In the TAG, text encoding requires capturing both deep semantic information and shallow context patterns to identify both global and local anomalies. Therefore, along with the GNN to encode the graph structure, triple encoders are introduced in our global GAD module. The triple encoders comprise: (1) BOW encoder for shallow context text encoding, (2) LM encoder for deep semantic text encoding, and (3) GNN encoder for graph structural encoding.

Shallow context Encoder To capture shallow context features of the texts, we first employ the BOW (Bag of Words) technique to obtain the context embedding. For each text d_u, we compute $x_u \in \mathbb{R}^{d_V}$ as $x_u = \text{BOW}(d_u)$, where d_V is the vocabulary size. These context features show distributional anomalies that may not appear in the deep semantic space.

Deep Semantic Encoder While the BOW can capture the context feature, it may miss the contextual semantic information of the TAG. Therefore, we use a typical pre-trained language model, BERT Devlin (2018) with 110M parameters. The BERT model is trained using the masked language modeling objective. We use the starting token ([CLS]) to represent a summary of the input text. For a text d_u, its semantic embedding is denoted as $t_u \in \mathbb{R}^{d_\mathrm{L}}$, where $t_u = \mathrm{LM}(d_u)$. Let T represent the semantic embedding matrix. Since BERT has already been optimized on large corpora, we freeze its parameters and only train the GNN component.

Structural Graph Encoder For the GNN encoder, we choose the classic GCN Kipf and Welling (2016) module, which effectively integrates the feature of graphs with the graph structure. For each node u, the graph embedding $h_u^\mathrm{g} \in \mathbb{R}^{d_\mathrm{H}}$ is encoded by GNNs, $h_u^\mathrm{g} = \mathrm{GNN}(x_u)$, where d_H is the encoder size. Likewise, let H^g be the graph embedding matrix encoded by GNNs. We use context (BOW-based) embedding rather than semantic embeddings as the GNN input, as the GNN operates over the entire graph structure.

4.1.2 Text-Graph alignment

In this stage, we align the graph encoder with the text encoder. In the triple-encoders, the space of the graph embedding H^g is different from the semantic embedding space T. Therefore, we first feed the feature encoded by GNNs to an MLP to align the space:

$$h_u = \mathrm{MLP}(h_u^\mathrm{g}), \tag{2}$$

where h_u indicates the decoded context feature by the MLP. We denote H as the projected graph feature. Then, the scaled cosine similarities $\Lambda \in \mathbb{R}^{n\times n}$ between the semantic embeddings T and the decoded feature embeddings H are computed:

$$\Lambda = T \cdot H^\top \times e^\tau, \tag{3}$$

where τ indicates the hyperparameter temperature to scale the similarity values.

Then, in the first stage, we use a contrastive learning based loss function to align the semantic embeddings and the projected graph embeddings:

$$L^\mathrm{a} = \frac{1}{2}(\mathrm{CE}(\Lambda, y_P) + \mathrm{CE}(\Lambda^\top, y_P)), \tag{4}$$

where $y_P = (1, 2, \ldots, n)^T$ is the pseudo label vector for contrastive training and CE denotes the cross entropy loss function.

4.1.3 Graph Decoder

As discussed before, GAEs have been proven to be effective in GAD task. The features of global anomaly nodes deviate significantly from the majority, making them difficult to reconstruct using GNNs. In contrast, normal nodes tend to be more easily reconstructed. Therefore, after alignment for some epochs, a graph decoder is introduced to reconstruct the context feature and detect the anomaly nodes. The decoded feature $\hat{x}_u \in \mathbb{R}^\mathrm{V}$ is obtained by GNN:

$$\hat{x}_u = \mathrm{GNN}(h_u). \tag{5}$$

Let $\hat{X}$ be the decoded embedding matrix. The loss function L_G of the second stage combines the reconstruction loss and the alignment loss:

$$L^\mathrm{G} = (1-\alpha)\|\hat{X} - X\|_2 + \alpha L^\mathrm{a}, \tag{6}$$

where α balances the reconstruction loss and the alignment loss. Let L_u^G be the loss score of node u. We reconstruct the context feature rather than the semantic feature, as they capture more the statistical distribution, thus more effective to identify global anomaly nodes. Experiments in Section 5 also show the context features are more important than semantic embeddings in TAGs towards the anomaly detection problem.

Finally, the global anomaly score s_u^G are computed by the loss score of each node, $s_u^\mathrm{G} = \textsc{Norm}(L_u^\mathrm{G})$, where the min-max Normalization is employed to normalize the global anomaly score. The alignment loss scores are also critical in TAGs toward the anomaly detection problem, because for anomaly nodes, their context and semantic features may be inconsistent, making it difficult to align the GNN and LM, resulting in a high alignment loss. In contrast, for the normal nodes, there tends to be coherent, thus easier to align.

4.2 Local GAD module

In this stage, we propose a novel local GAD module to compute the local anomaly score of nodes. As discussed in Section 1, there is a distinct distribution difference between the local anomaly node and its neighbors. Therefore, TAGAD leverages the local subgraph of each node to compute the local anomaly score.

Specifically, for each node u, we construct two subgraphs: the *ego graph* G_E^u and the *text graph* G_T^u. The ego graph captures the original local graph structure, while the text graph G_T^u reflects node similarity within the local neighborhood based on semantic features. For an anomaly node whose neighbor features differ substantially, the text similarity with its neighbors is low, leading to a significant mismatch between G_T^u and G_E^u.

Therefore, we define the local anomaly score of node u as the differences between G_T^u and G_E^u. In the zero-shot settings, the final anomaly score is computed by combining the local and global anomaly scores directly. In the few-shot settings, instead of training the full model, we learn a common embedding that captures the shared semantics among nodes. By subtracting this common embedding from the semantic features, we amplify the distinction between the ego and text graphs, thereby making anomalies more detectable. Theoretical justifications of the proposed local GAD module can be found in Appendix B.

4.2.1 Zero-shot detection.

Under the zero-shot settings, we first construct two subgraphs for each node u: the ego graph G_E^u and the text graph G_T^u. To build the ego graph, we select up to W first-order neighbors of the node u, along with u itself, to form the node set V_u of the ego graph ($W = 100$ in practice). The induced subgraph over V_u from the original graph then forms the ego graph G_E^u.

In the text graph construction, we aim to capture semantic similarity among nodes in V_u using their semantic embeddings T. For each pair of nodes $i, j \in V_u$, we compute their similarity based on the semantic embeddings. An edge $(i, j) \in E_u^T$ is added if the similarity exceeds a threshold ϵ:

$$A_T^u(i,j) = \begin{cases} 1, & \text{if } \textsc{Sim}(t_i, t_j) \geq \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where A_T^u denotes the adjacency matrix in the text graph and $\textsc{Sim}$ is the cosine similarity function.

In the message passing, for the local anomaly node, the feature is always different from its neighbors. Therefore, we use the difference between G_E^u and G_T^u to indicate the local anomaly score of a node u. First, we get the summary embeddings Z_E^u and Z_T^u of two subgraphs G_E^u and G_T^u:

$$Z_E^u = \text{READOUT}(h_i; i \in V_u), Z_T^u = \text{READOUT}(t_i; i \in V_u), \tag{8}$$

where READOUT means the pooling operation, such as mean pooling and max pooling.

The differences between the ego graph and the text graph consist of feature differences and structural differences. We measure the feature difference using the distance between their respective summary embeddings, and the structural difference using the distance between their adjacency matrices:

$$s_u^{\mathrm{L}} = \textsc{Norm}((1-\beta)\|Z_E^u - Z_T^u\|_2 + \beta\|A_E^u - A_T^u\|_2), \tag{9}$$

where A_E^u and A_T^u indicate the adjacency matrix of two subgraphs, and $\beta \in (0, 1)$ is the hyperparameter to control the importance of the structural difference. Similarly, Min-Max Normalization is also used here as the $\textsc{Norm}$ function.

Finally, the summary score consists of two parts: the local anomaly score reflecting the local discrepancy and the global anomaly score indicating the common anomaly likelihood:

$$s_u = (1-\lambda)s_u^{\mathrm{G}} + \lambda s_u^{\mathrm{L}}, \tag{10}$$

where $\lambda \in (0, 1)$ indicates the hyperparameter to control the importance of the local anomaly score.

4.2.2 Few-shot detection

In subgraph construction, semantic features often contain excessive common information, which leads to uniformly high similarity among nodes and hides the local anomaly nodes. Therefore, it becomes critical to determine an appropriate value for the sensitivity parameter ϵ. In the few-shot settings, we intend to remove the common information from the local subgraph to amplify the structural differences for anomaly nodes. Consequently, a trainable parameter $w_t \in \mathbb{R}^{d_{\mathrm{L}}}$ with common knowledge is learned. We use the mean embedding of all the features to initialize:

$$w_t = \textsc{Mean}(t_u; u \in V) \tag{11}$$

Then, the common embedding is removed from the graph embedding and the semantic embedding:

$$h_i^l = h_i - w_t, t_i^l = t_i - w_t, \tag{12}$$

where h_i^l and t_i^l denote the local graph embedding and the local semantic embedding of node i.

Then, we build the ego graph and the text graph similarly. When building the text graph, the binary indicator in Eq 7 is non-differentiable, making the Neural Network hard to train. To address this issue, we approximate the binary indicator with the Gumbel softmax trick Jang et al. (2017) to build the text graph. Specifically, the text graph is computed by:

$$A_T^u(i,j) = \text{Sigmoid}((\text{Sim}(t_i^l, t_j^l) + \log\delta - \log(1-\delta))/\tau_g), \tag{13}$$

where $\delta \sim \text{Uniform}(0, 1)$ is the sampled Gumbel random variate and $\tau_g > 0$ is the temperature hyperparameter of Gumbel softmax, which is closer to 0. In this way, the $A_T^u(i,j)$ tends to be closer to 0 or 1.

After that, we use the same functions as Eq. 8 to get the summary embeddings Z_E^u and Z_T^u. Finally, the Cross Entropy Loss is used as the loss function of the local GAD module:

$$L_{\mathrm{L}} = \sum_{u \in V_{\text{train}}} \text{CE}(y_u, s_u^{\mathrm{L}}) \tag{14}$$

5 Experiments

5.1 Experiment Setup

Datasets The experiments were performed on three synthetic datasets, including Cora, Arxiv, and Pubmed. We use a commonly used method Sen et al. (2008) in GAD to inject the anomaly nodes into the graph. This method introduces two types of anomaly nodes into the graph: structural anomaly nodes, created by forming densely connected subgraphs with probabilistic edge deletion; and contextual anomaly nodes, generated by altering node features to maximize dissimilarity from the randomly chosen nodes. A detailed description of each dataset and the anomaly injection process is provided in Appendix C.1.

Baselines We compare TAGAD with both unsupervised and supervised learning methods. These methods can only deal with the numeric feature, so we use the feature obtained by BOW and LM, respectively. We also compare the performance of baselines by concatenating the feature obtained by BOW and LM in Appendix D.1.

Unsupervised learning methods include traditional shallow methods SCAN Xu et al. (2007), Radar Li et al. (2017) and ANOMALOUS Peng et al. (2018), reconstruction based methods, DOMINANT Ding et al. (2019), AnomalyDAE Fan et al. (2020), and GAD-NR Roy et al. (2024), contrastive learning based methods, CONAD Xu et al. (2022), NLGAD Duan et al. (2023), and CoLA Liu et al. (2021b) .

Supervised learning methods include two conventional GNNs, GCN Kipf and Welling (2016) and GAT Veličković et al. (2017), five state-of-the-art GNNs specifically designed for GAD, i.e., GATSEP Platonov et al. (2023), PC-GNN Liu et al. (2021a), AMNET Chai et al. (2022), and

BWGNN Tang et al. (2022), and two decision-tree based GAD methods, XGBGRAPH and RFGRAPH Tang et al. (2024). For detailed information, refer to Appendix C.2.

We also conduct experiments by removing the key components of TAGAD on all datasets. Specifically, we evaluate four variants, namely TAGAD(A), TAGAD(R), TAGAD(G), and TAGAD(L). In TAGAD(A), only the alignment loss is used as the anomaly score, without incorporating the reconstruction loss and the local GAD module. Similarly, in TAGAD(R), the alignment stage is removed, and the reconstruction loss alone is used to compute the anomaly score. TAGAD(G) removes the local GAD module entirely and relies on the global anomaly score for prediction. Conversely, TAGAD(L) eliminates the global GAD module, using only the summary representations from the LM as node features in the local subgraph for anomaly detection.

Evaluation and Implementation Following the benchmark Tang et al. (2024), we employ Area Under ROC (AUC) as our evaluation metric for GAD. We report the average AUC across 5 trials. More implementation details can be found in Appendix C.3. All experiments were run on an Ubuntu 18.04 LTS server with six Intel Xeon 6130 CPUs (13 cores, 2.10GHz), 256GB of main memory, and two NVIDIA GeForce RTX V100 GPUs.

5.2 Performance of GAD

Zero-shots We first compare TAGAD with unsupervised baseline methods. The results are shown in Table 1 (more results in Appendix D.1). We have the following observations: (1) The proposed TAGAD performs best on most datasets, with an average improvement of $+7.8\% \sim +36.9\%$. In the Arxiv dataset, most of the models can't work due to the limited GPU memory, while our model can perform well because only two simple GCN and MLP are trained in the global module. (2) We can also find a huge improvement of TAGAD compared with the four variants of TAGAD. Specifically, TAGAD achieves an improvement in AUC of 17% and 22% compared to TAGAD(G) and TAGAD(L) in the Cora dataset. This improvement is due to the combination of both the global anomaly score and the local anomaly score. The TAGAD(G) method also performs better than TAGAD(A) and TAGAD(R) because of the two stages of alignment and reconstruction. (3) Most models perform better using Bag-of-Words (BOW) based context features as input features than using LM-based semantic representations, indicating that in GAD tasks, context features play a more critical role than semantic features.

Few-shots Table 2 shows the comparison results of TAGAD with supervised methods under two few-shot settings: 2-shots, and 5-shots. The global GAD module of TAGAD is unsupervised, so we don't compare TAGAD(A), TAGAD(R), and TAGAD(G) in this settings and only compare the local GAD module TAGAD(L). TAGAD consistently emerges as the top performer, outperforming the best baseline by around $0.3\% \sim 18\%$. TAGAD performance is remarkably stable, varying by no more than 2% across two different settings. The stability is due to the effectiveness of the simple common embedding, which can be reliably trained with very limited labeled data. In contrast, the decision-tree-based methods, such as XGBGraph and RFGraph, which perform well in the GAD problem under fully supervised settings Tang et al. (2024), suffer notable degradation under the few-shot settings. This suggests that these models are heavily reliant on labeled datasets and struggle to generalize under few-shot settings.

5.3 Ablation Studies

To better analyze the impact of LMs, we explore other LMs such as e5-v2-base Wang et al. (2022) with 110M parameters. We also try larger LMs such as e5-v2-large with 335M parameters and DeBERTa-large with 350M parameters. An external experiment is conducted to assess whether to fine-tune the LM. The LM is mainly used in the global module, so we only report the performance achieved with P2TAG(G) under zero-shot settings. The results are reported in Table 3. Generally, the results of LMs are quite similar, with differences within 4%. We also observe that training with the fine-tuned language model (LM) is significantly slower than using the frozen LM. More critically, fine-tuning results in suboptimal performance, for example, achieving only 0.511 AUC on the Cora dataset, whereas the frozen LM attains much higher accuracy. This performance gap arises because the pretrained LM has already learned rich semantic representations. When the LM is jointly trained

Table 1: Performance Comparison under zero-shot settings. The highest performance is highlighted in boldface; the second highest performance is underlined. "—" indicates that the algorithm cannot complete on large datasets due to limited GPU memory.

Method	Cora		Arxiv		Pubmed	
	BOW	LM	BOW	LM	BOW	LM
SCAN	0.705	0.705	0.668	0.668	0.721	0.721
Radar	0.578	0.566	–	–	0.480	0.497
ANOMALOUS	0.550	0.582	–	–	0.463	0.465
DOMINANT	0.780	0.618	0.709	0.522	0.771	0.773
AnomalyDAE	0.773	0.737	–	–	0.850	0.844
GAD-NR	0.742	0.739	–	–	0.686	0.694
CONAD	0.827	0.583	0.685	0.481	0.796	0.740
NLGAD	0.665	0.676	–	–	0.741	0.709
COLA	0.536	0.6327	–	–	0.489	0.696
TAGAD	**0.905**		**0.747**		**0.874**	
TAGAD(A)	0.804		0.671		0.727	
TAGAD(R)	0.777		0.704		0.705	
TAGAD(G)	0.834		0.714		0.849	
TAGAD(L)	0.685		0.507		0.708	

Table 2: Comparison of Classification Performance in few-shot settings

Method	Cora				Arxiv				Pubmed			
	2-shot		5-shot		2-shot		5-shot		2-shot		5-shot	
	BOW	Text	BOW	Text	BOW	Text	BOW	Text	BOW	Text	BOW	Text
GCN	0.757	0.656	0.818	0.599	0.617	0.688	0.741	0.685	0.626	0.672	0.706	0.658
GAT	0.722	0.582	0.646	0.528	0.695	0.487	0.702	0.497	0.699	0.524	0.690	0.515
GATSep	0.689	0.529	0.810	0.519	0.677	0.497	0.706	0.508	0.696	0.510	0.718	0.501
PC-GNN	0.787	0.632	0.815	0.604	0.644	0.624	0.745	0.622	0.678	0.619	0.735	0.618
AMNet	0.591	0.673	0.525	0.653	0.657	0.526	0.696	0.663	0.739	0.519	0.739	0.566
BWGNN	0.744	0.557	0.768	0.558	0.649	0.524	0.672	0.529	0.730	0.676	0.762	0.672
XGB-Graph	0.5	0.5	0.615	0.483	0.5	0.5	0.596	0.501	0.5	0.5	0.592	0.511
RF-Graph	0.749	0.535	0.782	0.553	0.683	0.675	0.744	0.724	0.615	0.582	0.686	0.526
TAGAD	**0.930**		**0.941**		**0.748**		**0.757**		**0.875**		**0.877**	
TAGAD(L)	0.748		0.764		0.738		0.741		0.704		0.707	

with a randomly initialized graph neural network (GNN), its parameters may have substantial changes, thereby disrupting its ability to represent the semantic features.

6 Conclusions

In this paper, we study the problem of anomaly detection on the TAG. We propose a novel framework named TAGAD, which consists of two modules, respectively Contrastive learning based global GAD and Subgraph comparison based local GAD. The global GAD module utilizes a contrastive learning based method to align the GNN and LM, then employs the GAE technique to compute the global anomaly scores. In the local GAD module, we compute the local anomaly score by comparing the ego graph and the text graph for each node. Extensive experiments on three datasets demonstrate the effectiveness of our model compared to existing approaches.

LM	Cora		Arxiv		Pubmed	
	AUC	Time(s)	AUC	Time(s)	AUC	Time(s)
DeBERTa-base	**0.834**	**13.76**	0.714	**700.38**	**0.849**	**61.27**
e5-v2-base	0.828	20.98	**0.728**	850.17	0.813	62.95
DeBERTa-large	0.812	42.14	0.727	1923.86	0.793	198.83
e5-v2-large	0.825	34.81	0.725	1671.43	0.829	155.28
DeBERTa-base (FT)	0.518	561.25	–	–	0.562	1981.44
e5-v2-base (FT)	0.671	564.77	–	–	0.486	3941.81
DeBERTa-large (FT)	0.511	549.87	–	–	0.572	1672.92
e5-v2-large (FT)	0.582	1671.43	–	–	0.493	1675.40

Table 3: Ablation study of language models on the datasets. We choose various LMs, then report the performance achieved with P2TAG(G). The highest performance and the shortest time are highlighted in boldface.

References

Ziwei Chai, Siqi You, Yang Yang, Shiliang Pu, Jiarong Xu, Haoyang Cai, and Weihao Jiang. 2022. Can abnormality be detected by graph neural networks?. In *IJCAI*. 1945–1951.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM international conference on data mining*. SIAM, 594–602.

Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM international conference on information & knowledge management*. 315–324.

Jingcan Duan, Pei Zhang, Siwei Wang, Jingtao Hu, Hu Jin, Jiaxin Zhang, Haifang Zhou, and Xinwang Liu. 2023. Normality learning-based graph anomaly detection via multi-scale contrastive learning. In *Proceedings of the 31st ACM International Conference on Multimedia*. 7502–7511.

Haoyi Fan, Fengbin Zhang, and Zuoyong Li. 2020. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5685–5689.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

Xuanwen Huang, Yang Yang, Yang Wang, Chunping Wang, Zhisheng Zhang, Jiarong Xu, Lei Chen, and Michalis Vazirgiannis. 2022. Dgraph: A large-scale financial dataset for graph anomaly detection. *Advances in Neural Information Processing Systems* 35 (2022), 22765–22777.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparametrization with Gumble-Softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.

Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141* (2020).

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

Jundong Li, Harsh Dani, Xia Hu, and Huan Liu. 2017. Radar: Residual analysis for anomaly detection in attributed networks.. In *IJCAI*, Vol. 17. 2152–2158.

Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. 2022. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems* 35 (2022), 27021–27035.

Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021a. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*. 3168–3177.

Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis. 2021b. Anomaly detection on attributed networks via contrastive self-supervised learning. *IEEE transactions on neural networks and learning systems* 33, 6 (2021), 2378–2392.

Zhiwei Liu, Yingtong Dou, Philip S Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1569–1572.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

Zhen Peng, Minnan Luo, Jundong Li, Huan Liu, Qinghua Zheng, et al. 2018. ANOMALOUS: A Joint Modeling Approach for Anomaly Detection on Attributed Networks.. In *IJCAI*, Vol. 18. 3513–3519.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A critical look at the evaluation of GNNs under heterophily: Are we really making progress? *arXiv preprint arXiv:2302.11640* (2023).

Amit Roy, Juan Shu, Jia Li, Carl Yang, Olivier Elshocht, Jeroen Smeets, and Pan Li. 2024. Gad-nr: Graph anomaly detection via neighborhood reconstruction. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 576–585.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 1715.

Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. 2023. All in One: Multi-Task Prompting for Graph Neural Networks.(2023). (2023).

Jianheng Tang, Fengrui Hua, Ziqi Gao, Peilin Zhao, and Jia Li. 2024. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems* 36 (2024).

Jianheng Tang, Jiajin Li, Ziqi Gao, and Jia Li. 2022. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*. PMLR, 21076–21089.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533* (2022).

Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).

Zhihao Wen and Yuan Fang. 2023. Augmenting low-resource text classification with graph-grounded pre-training and prompting. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 506–516.

Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. 2022. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM Web Conference 2022*. 1070–1079.

Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. 2007. Scan: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. 824–833.

Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. 2022. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 444–457.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, et al. 2023. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. *Advances in Neural Information Processing Systems* 36 (2023), 17238–17264.

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

Huanjing Zhao, Beining Yang, Yukuo Cen, Junyu Ren, Chenhui Zhang, Yuxiao Dong, Evgeny Kharlamov, Shu Zhao, and Jie Tang. 2024. Pre-training and prompting for few-shot node classification on text-attributed graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4467–4478.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021*. 2069–2080.

Algorithm 1: TAGAD(G)

Input : A TAG $\mathcal{G} = (V, E, D)$, the total training epoch N, the first stage training epoch M, the scaled temperature τ, the similarity threshold ϵ, the hyperparameter α
Output : Global anomaly score S of all nodes.
1 $T = \mathrm{LM}(D), X = \mathrm{BOW}(D)$;
2 $p =$ True ;
3 **for** $epoch = 1, \ldots, N$ **do**
4 $\quad H^{\mathrm{g}} = \mathrm{GNN}(X; \theta_E)$;
5 $\quad H = \mathrm{MLP}(H^{\mathrm{g}}; \theta)$;
6 $\quad \hat{X} = \mathrm{GNN}(H; \theta_D)$;
7 $\quad \Lambda = T \cdot H^T \times e^{\tau}$;
8 $\quad L^a = (CE(\Lambda, y) + CE(\Lambda^T, y))/2$;
9 $\quad L^R = (1-\alpha)\|\hat{X} - X\|_2$;
10 $\quad$ **if** p **then**
11 $\quad\quad L^{\mathrm{G}} = L^a$;
12 $\quad$ **else**
13 $\quad\quad L^{\mathrm{G}} = (1-\alpha)L^a + \alpha L^R$;
14 $\quad$ **if** $epoch \geq M$ **then**
15 $\quad\quad p =$ False
16 $\quad$ Update the weight parameters θ, θ_E, and θ_D by using gradient descent
17 $S^G = \mathrm{NORM}(L)$;
18 **return** S^G;

Algorithm 2: TAGAD(L)

Input : A TAG $\mathcal{G} = (V, E, D)$, a set of training nodes V_{train}, the class label y_i of the node $v_i \in V_{\text{train}}$, the training epochs N, the projected features H, the semantic embeddings T, the similarity threshold ϵ, the hyperparameter β
Output : Anomaly score S_L of all nodes.
1 $w_t = \mathrm{MEAN}\{t_u; u \in V\}$;
2 **for** $epoch = 1, \ldots, N$ **do**
3 $\quad H = H - w_t$;
4 $\quad T = T - w_t$;
5 $\quad$ **for** $v \in V_{train}$ **do**
6 $\quad\quad$ Sampling W first-order nodes of v to form the ego graph A_E^v ;
7 $\quad\quad$ Build the text graph A_T^v using Eq. 13 ;
8 $\quad\quad$ Compute ego graph embedding Z_E^v and text graph embedding Z_T^v and using Eq. 8;
9 $\quad\quad s_v^L = \mathrm{NORM}(\|Z_E^u - Z_T^u\| + \beta\|A_E^u - A_T^u\|)$
10 $\quad$ Compute the loss L using Eq. 14 ;
11 $\quad$ Update w_t by using gradient descent
12 **for** $v \in V$ **do**
13 $\quad$ Compute the local score S_L^v using a similar way to Lines 18–21.
14 **return** S_L;

A Algorithm and Complexity

A.1 Algorithmic description

The global GAD module of TAGAD, the local GAD module of TAGAD are presented in Algorithm 1 and Algorithm 2, respectively.

A.2 Complexity Analysis

In the global module, Lines 1–2 are pre-processing. For each epoch, the time complexity of the GNN encoder (line 4) is $O(nLd_{\mathrm{V}}d_{\mathrm{H}})$, where L is the layer number of the GNN. In line 5, it takes

$O(nd_{\mathrm{H}}d_{\mathrm{L}})$ in the feature projection. In line 6, the GNN decoder process takes $O(nd_{\mathrm{L}}d_{\mathrm{V}})$ Overall, the time complexity of Algorithm 1 is bounded by $O(Nn(Ld_{\mathrm{V}}d_{\mathrm{H}} + Ld_{\mathrm{L}}d_{\mathrm{V}} + d_{\mathrm{H}}d_{\mathrm{L}}))$

In the local module, it takes $O(nd_{\mathrm{L}})$ for initialization (line 1). Then, for each epoch and each training node v, it takes $O(W)$ to sample an ego graph A_E^v (line 6) and takes $O(W^2d_{\mathrm{L}})$ to build the text graph A_T^v (line 7). Then, the time complexity of Eq. 8 is $O(Wd_{\mathrm{L}})$. In the few-shot settings, there are few nodes in V_{train}. Therefore, it takes $O(NW^2d_{\mathrm{L}})$ to train the local module of TAGAD (lines 2–11). Similarly, computing the local anomaly score S_l (lines 12–13) takes $O(nW^2d_{\mathrm{L}})$. Overall, the time complexity of Algorithm 2 is bounded by $O(nW^2d_{\mathrm{L}})$

B Theoretical Analyze

B.1 Local anomaly score

Theorem 1 *Given a TAG $G = (V, E, D)$, in the local GAD module, the expected local anomaly score for anomalous nodes is greater than that for normal nodes.*

To prove this theorem, we make the following assumptions:

1. For a local anomaly node, the feature deviation is random, not correlated to its neighbors.
2. For normal nodes, the structural and textual similarities are positively correlated, as their text content and graph neighbors are semantically coherent.
3. For a normal node, the feature deviation from the overall distribution is similar to that of its neighboring nodes.

We first consider the structural difference between G_E and G_T:

$$\begin{aligned} E(\|A_E^u - A_T^u\|) &= \sum_{i,j} E[(A_E^u(i,j) - A_T^u(i,j))^2] \\ &= \sum_{i,j\in V_u} \mathrm{Var}(A_E^u(i,j)) + \mathrm{Var}(A_T^u(i,j)) - \mathrm{Cov}(A_E^u(i,j)), A_T^u(i,j))) \end{aligned} \tag{15}$$

According to our assumption, for anomaly nodes u, the ego graph and the text graph are less correlated the normal nodes v, so $\mathrm{Cov}(A_E^u(u,j)), A_T^u(u,j)) < \mathrm{Cov}(A_E^u(u,j)), A_T^u(u,j))$. Meanwhile, the feature of anomaly nodes is more random, hence, the variance of the text graph for anomaly nodes is also large. Overall, the expected structural difference between the ego graph and the text graph is larger for anomaly nodes than for normal nodes.

As discussed in Section 4.1, the feature embedding difference for anomaly nodes is also higher due to the hard alignment. Therefore, the total difference between the ego graph and the text graph—comprising both feature and structural components as defined in Eq. 10—serves as an effective local anomaly score, particularly sensitive to the presence of anomaly nodes.

B.2 Remove embeddings

Theorem 2 *Given a TAG $G = (V, E, D)$, and the common embedding vector w_t representing the shared semantic information among node features, in the local GAD module, removing w_t from the semantic embeddings in the local GAD module amplifies the structural differences between the ego graph and the text graph for anomalous nodes.*

We denote the semantic embedding for the node i as $t_i = w_t + \delta_i$, where δ_i is a deviation. δ_i is a significant deviation for anomaly nodes, while δ_i is a small noise, related to the structure of the normal nodes. For the original similarity between node i and node j,

$$sim(t_i, t_j) = \frac{t_i \cdot t_j}{|t_i||t_j|} = \frac{(w_t + \delta_i)\cdot(w_t + \delta_j)}{\|w_t + \delta_i\|\|w_t + \delta_j\|}. \tag{16}$$

It can be easily found that the common feature hides local anomalies by inducing high similarities. In the special case, if $w_t \gg \delta_i$, the similarity $\mathrm{sim}(t_i, t_j) \approx 1$ for any pair of nodes, regardless of whether they are normal or anomalous.

Dataset	#Node	#Edges	#Attributes	#Anomalies (Rate)
Cora	2.2K	8.1K	1361	194(8.51)
Arxiv	169K	1.4M	128	10K(6.14)
Pubmed	19K	112K	500	963(4.89)

Table 4: Statistics of datasets.

After removing w_t, the local embedding is δ_i for node i. Therefore, the new similarity is:

$$sim(t_u^l, t_v^l) = \frac{\delta_i \cdot \delta_j}{\|\delta_i\| \cdot \|\delta_j\|} \tag{17}$$

From the above assumptions, for a normal node, δ_i and δ_j are similar, leading to high similarity. However, for anomalous nodes, δ_i is uncorrelated with neighbors' deviations, leading to significantly lower similarity scores. Consequently, the structure of the text graph diverges more strongly from that of the ego graph for anomalous nodes, thereby amplifying the local anomaly signal.

Overall, removing the common embedding w_t from the semantic embeddings amplifies the structural differences between the ego graph and the text graph for anomalous nodes.

C Details of Experiment Setup

C.1 Description of Datasets

The statistics of the datasets are shown in Table 4.

We make a slight modification to a widely used approach Liu et al. (2022) to inject anomaly nodes in the TAG. We use two techniques in the method, injecting structural anomaly nodes and contextual anomaly nodes. The method is described below.

Injecting structural anomaly nodes. In this technique, we create g densely connected groups of nodes to inject the structural outliers. Each group contains m nodes, resulting in a total of $m \times g$ structural anomaly nodes. Specifically, for each group, we first randomly sample m nodes without replacement to form this group. Then, for these nodes, we make them fully connected and then drop each edge independently with probability p. In experiments, we set $p = 0.2$.

Injecting contextual anomaly nodes. In this technique, we inject o contextual anomaly nodes. First, we sample o nodes as contextual anomaly nodes from the node set V without replacement. These selected nodes are denoted as V_c, where $|V_c| = o$. The remaining nodes $V_r = V \setminus V_c$ form the reference set. Then, for each node $v \in V_c$, we randomly choose q nodes without replacement from V_r. Among these q reference nodes, we identify the most dissimilar node u to v by computing Euclidean distances and then modify $s_v = s_u$.

C.2 Description of Baselines

The following unsupervised learning methods are compared to highlight the effectiveness of the proposed TAGAD under zero-shot settings.

- SCAN Xu et al. (2007): A structural clustering method to detect clusters and anomaly nodes based on a structural similarity measure.
- RADAR Li et al. (2017): A learning framework that characterizes the residuals of attribute information.
- ANOMALOUS Peng et al. (2018): A joint framework to conduct attribute selection and anomaly detection jointly based on CUR decomposition and residual analysis.
- DOMINANT Henaff et al. (2015): GNN that reconstructs the features and structure of the graph using the auto-encoder.
- ANOMALYDAE Fan et al. (2020): GAE that reconstructs both node embeddings and attribute embeddings.

- GAD-NR Roy et al. (2024): GAE that incorporates neighborhood reconstruction.
- CONAD Xu et al. (2022): GNN that uses a data augmentation strategy to model prior human knowledge.
- NLGAD Duan et al. (2023): Normality learning-based GNN via multi-scale contrastive learning.
- COLA Liu et al. (2021b): A contrastive learning based GNN that captures the relationship between each node and its neighboring structure.

The following supervised learning methods are compared to highlight the effectiveness of the proposed TAGAD under few-shot settings.

- GCN Kipf and Welling (2016): Standard graph convolution network (GCN).
- GAT Veličković et al. (2017): Standard graph attention network (GAT).
- GATSEP Platonov et al. (2023): GNN that deals with the heterophilous graphs.
- PC-GNN Liu et al. (2021a): GNN that handles imbalanced classes.
- AMNET Chai et al. (2022): GNN that analyzes anomalies via the lens of the graph spectrum.
- BWGNN Tang et al. (2022): GNN using graph spectral filters to detect fraudsters.
- RF-GRAPH Tang et al. (2024): Tree-ensembled method using random forest and neighbor aggregation.
- XGB-GRAPH Tang et al. (2024): Tree-ensembled method using XGBoost and neighbor aggregation.

C.3 Details of Implementation

We implemented TAGAD in PyTorch 2.2.0 Paszke et al. (2019) and Python 3.11. For our model, the selection of LMs and GNNs is flexible. In our experiment, we choose a representative LM–DeBERTa-base and a powerful GCN model for the main experiments. The DeBERTa-base is a pre-trained language model with 100M parameters. The hidden size of the DeBERTa-base model is 768. We keep the same hidden size of the GCN model with DeBERTa-base. We use AdamW optimizer with learning rate $lr = 1e-3$ and weighting decay $5e-4$ for model optimization. For all datasets, we run 200 epochs in the global GAD module and 50 epochs in the local GAD module. In the global GAD module, the scaled temperature τ is 0.07. In the local GAD module, the similarity threshold ϵ is 0.95. The hyperparameters α, β, λ are set to be $0.6, 0.4, 0.4$. Additionally, we apply Early Stopping with a patience of 10 to prevent overfitting and terminate training when performance stops improving.

In unsupervised settings, for NLGAD and COLA, we use the default parameters described in the original papers. For other methods, we use the codes and parameters provided in the PyGOD library Liu et al. (2022). In supervised settings, we use the codes and parameters provided in the GAD benchmark Tang et al. (2024). The links to their source codes are as follows:

- PyGoD: `https://github.com/pygod-team/pygod`
- GADBench: `https://github.com/squareRoot3/GADBench`
- NLGAD: `https://github.com/FelixDJC/NLGAD`
- COLA: `https://github.com/TrustAGI-Lab/CoLA`

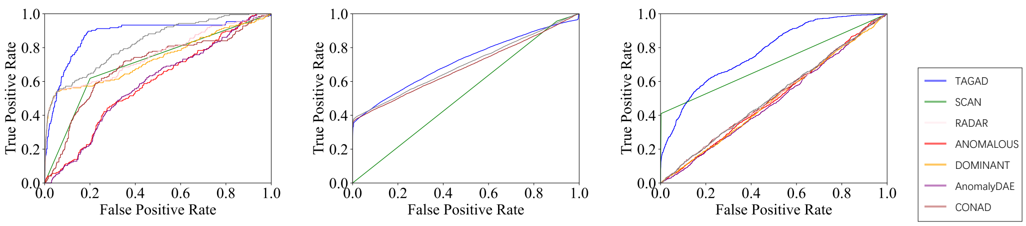
D Supplemental Experiments

D.1 Performance with combined feature

In this section, we compare our model performance with unsupervised baselines using the combined feature of the BOW feature and the LM feature as input. As shown in Table 5, our model continues to outperform the baselines, although both global and local perspectives of textual features are provided. Notably, the baselines, despite both types of features input, perform worse than when using only one perspective (either BOW or LM) in most cases. This highlights the importance of joint modeling rather than simple feature concatenation for effectively leveraging textual information in GAD.

Table 5: Performance Comparison with combined feature of the BOW feature and the LM feature under zero-shot settings. The highest performance is highlighted in boldface. "—" indicates that the algorithm cannot complete on large datasets due to limited GPU memory.

Method	Cora	Arxiv	Pubmed
SCAN	0.705	0.668	0.721
Radar	0.575	–	0.489
ANOMALOUS	0.547	–	0.358
DOMINANT	0.707	0.688	0.459
AnomalyDAE	0.713	–	0.499
GAD-NR	0.712	–	0.677
CONAD	0.688	0.692	0.694
NLGAD	0.500	–	0.571
COLA	0.590	–	0.554
TAGAD	**0.905**	**0.747**	**0.874**



(a) Cora (b) Arxiv (c) Pubmed

Figure 2: ROC curves on different datasets. The seven subplots show the True Positive Rate (TPR) vs False Positive Rate (FPR) for different algorithms across various datasets. The larger the area under the curve, the better the performance of graph anomaly detection.

D.2 Performance Comparison in Terms of AUC

We compare TAGAD with 6 unsupervised baselines in two datasets. The ROC curves on three datasets are illustrated in Fig. 3. We can find that the True Positive Rate of our model is higher than other models in most conditions.

D.3 Hyperparameter Analysis

In this part, we conduct a comprehensive analysis of four key hyperparameters α, β, λ, and ϵ to evaluate their impact on the performance of our framework. In detail, the analysis of α is performed on TAGAD(G), while others are performed on TAGAD under zero-shot settings. Figure 3 shows the AUC of our model on three datasets under zero-shot settings as one of the parameters α, β, λ, ϵ varies. By default, $\alpha = 0.5, \beta = 0.5, \lambda = 0.5, \epsilon = 0.9$.

Parameter α As shown in Figure 3a, with increasing α, the performance improves at first, but decreases later. This is due to the balance of the alignment and the reconstruction loss. Ignoring either loss will degrade the model's performance.

Parameter β From Figure 3b, we can observe that the performance of different β is stable. This is because both the structural difference and the feature difference provide overlapping insights into local anomaly nodes.

Parameter λ Figure 3c shows the performance with different λ. We can find the best performance in different λ. This is because the ratio of global and local anomaly nodes is different across different datasets. For the datasets with more global anomaly nodes, such as Pubmed, a smaller value of λ leads to better performance. For the datasets with more local anomaly nodes, such as Arxiv, the larger value of λ is better.

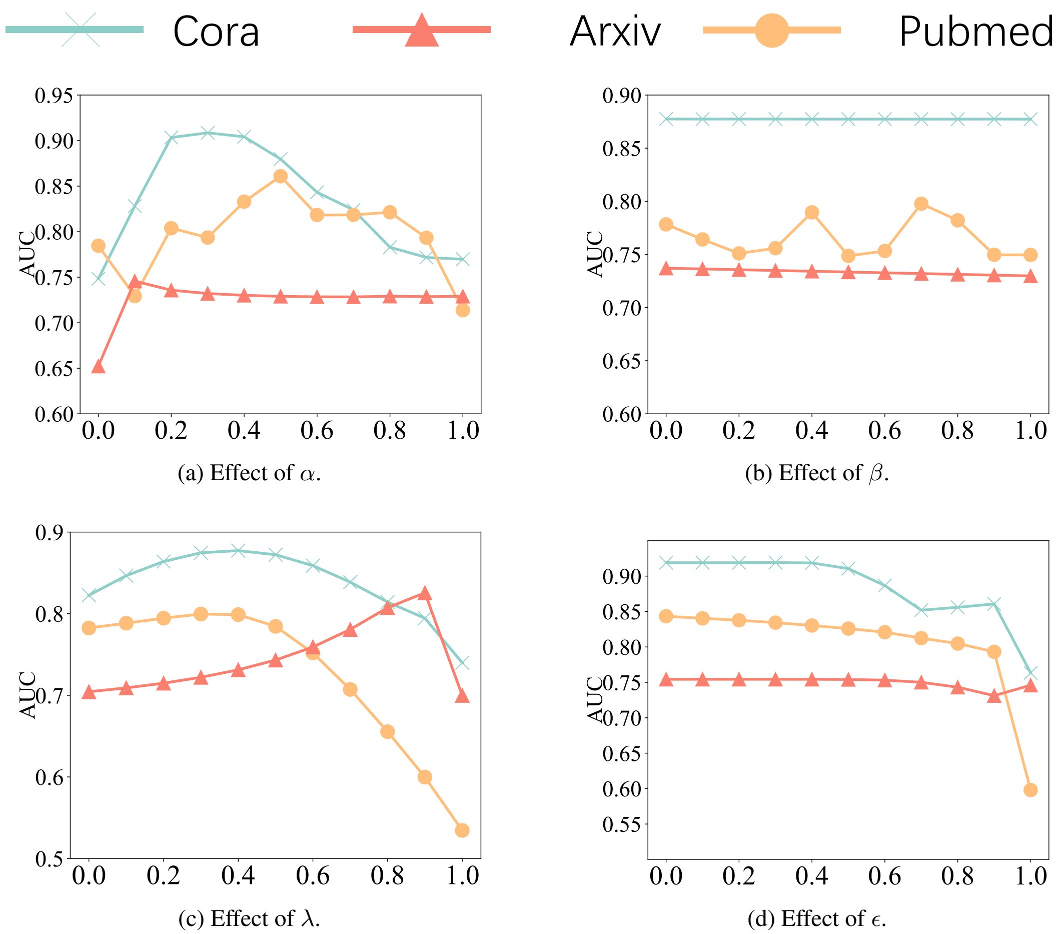


Figure 3: Parameter sensitivities of TAGAD w.r.t. five hyper-parameters on three datasets.

Parameter ϵ In order to evaluate the effectiveness of ϵ, we adopt different values of to adjust the similarity threshold when constructing the text graph under zero-shot settings. We can see that the best parameter ϵ is different in different datasets. This is related to the ratio of common features. A higher ϵ performs better when nodes share many same features, while performance is more stable when feature diversity is high.

E Limitation of Our work

One limitation of this work is the absence of publicly available real-world datasets, necessitating the use of synthetic datasets with the widely used injected anomaly nodes approach in the experiment. A potential future direction involves exploring the integration of large language models (LLMs) to enhance anomaly detection on TAGs.

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: Our main claims are elaborated in both the abstract and introduction sections.

   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: : The limitations of this paper have been discussed in Section E.

   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: The assumptions and the proof are shown in Appendix B.

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: We display the experimental instruction in the paper, provide the hyperparameter search space, and upload the source code for reproduction of the proposed method.

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: Yes, all the datasets are included along with the uploaded source code.

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: All the experimental details are given in Section 6 and Appendix C.

   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 context significance**

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

   Answer: [Yes]

   Justification: All the experimental results are acquired by multiple trials of experiments, and we report the average results.

   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 context 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: Yes, we provide the computing infrastructures in Sec 6.

   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: This research conforms with the Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    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: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

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

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - 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: The paper does not involve potential risks incurred by study participants.

    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: LLM used in this paper does not impact the core methodology, scientific rigorousness, or originality of the research.

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.