HyGROVE: Generative View Learning with Hypergraph Transformation for Unsupervised Graph-Level Anomaly Detection

Xiyu Hu[1], Jindong Li[2], Yali Fu[1], Qi Wang[1*]
[1] School of Artificial Intelligence, Jilin University, Changchun, China
[2]The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
{xyhu23, fuyl23}@mails.jlu.edu.cn, qiwang@jlu.edu.cn, jli839@connect.hkust-gz.edu.cn

***Abstract*—Graph-level anomaly detection (GLAD) aims to identify abnormal graphs that deviate from typical structural or attribute patterns, playing a vital role in cybersecurity, bioinformatics, and fraud detection. While contrastive learning has recently been introduced to GLAD, existing methods heavily rely on heuristic augmentations such as node deletion or feature masking, which risk disrupting key semantics or obscuring subtle anomalies. Moreover, most methods that employ shallow GNNs may struggle to capture the long-range dependencies and high-order interactions. To address these issues, we propose HyGROVE, a novel framework that combines HyperFormer and GeneRative–COntrastiVE learning for unsupervised GLAD. HyGROVE leverages a generative model to explicitly learn node-level latent distributions and sample multiple embeddings to form contrastive views. It then incorporates a hypergraph transformation to capture high-order local interactions beyond pairwise links, while a transformer models global semantic dependencies. Finally, hierarchical contrastive learning at both node and graph levels, together with an adaptive scoring module, detects anomalies from multiple perspectives. To our knowledge, this is the first framework that introduces a graph generation–contrastive learning paradigm for unsupervised GLAD. Experiments on 11 datasets and 10 baselines verify HyGROVE's superiority. The code is available at https://github.com/xyhu1/HyGROVE.**

***Index Terms*—Graph-level anomaly detection, Unsupervised learning, Graph neural networks, Generative Learning, Contrastive Learning**

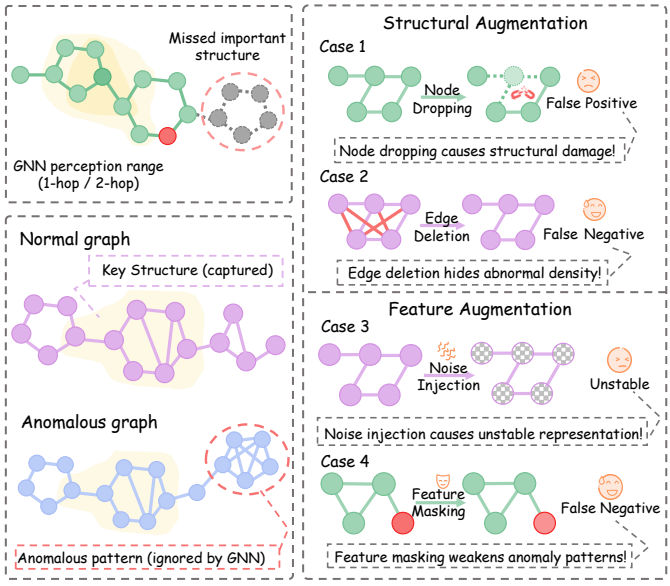


Fig. 1: Limitations of GCL-based UGLAD. Left: Shallow GNNs miss long-range and group patterns. Right: View perturbations distort semantics, causing false or missed anomalies.

I. Introduction

Graphs are a fundamental data structure in many real-world applications such as cybersecurity, bioinformatics, and social network analysis [1]–[3]. Among various graph learning tasks, Graph-level anomaly detection (GLAD) aims to identify graphs that deviate from the majority in structure or attributes. Such anomalies often indicate rare but meaningful events, such as toxic molecular compounds, fraudulent transactions, or compromised communication patterns [4].

In many real-world applications, anomalous graphs are difficult to label, which necessitates the use of unsupervised methods for graph-level anomaly detection. Recently, graph contrastive learning (GCL) has emerged as a promising solution [5]–[8]. By generating multiple augmented views of each graph and maximizing agreement between their representations, GCL encourages the model to learn informative features without relying on labeled data. Common augmentation strategies include structural augmentations and feature augmentations [9]. These strategies have shown effectiveness in improving representation quality for downstream anomaly detection. Despite these advancements, existing GCL-based methods still face two major limitations.

Firstly, the construction of contrastive views mainly relies on heuristic perturbations, such as node or edge deletion, feature masking, and random noise injection. These approaches assume that small perturbations preserve graph semantics, but this lacks theoretical support and cannot ensure the views follow the same distribution as the original graph. Once such perturbations disrupt key anomalous patterns, the model performance may degrade. For example, as illustrated in Figure 1, removing a node from a normal graph may destroy its global topology (Case 1), leading to false positives, while applying feature masking to an anomalous graph may obscure critical anomaly cues (Case 4), resulting in false negatives. These

* Corresponding author.

issues become more severe when anomaly signals are weak or globally dispersed, as perturbations further compromise semantic consistency. Fundamentally, current view generation strategies do not model the relationship between different views from the perspective of latent distribution, resulting in a lack of semantic alignment across views. This raises a critical necessity of modeling the latent graph distribution instead of relying on manual perturbations to generate semantically consistent contrastive views.

Secondly, most models adopt shallow graph neural networks (GNNs) as encoders, such as GCN or GIN. These models primarily aggregate local neighborhood information, whereas graph-level anomalies often arise from sparse yet informative patterns, such as long-range dependencies or high-order structural interactions, which shallow GNNs struggle to capture. For example, as shown in Figure 1, shallow GNN-based methods may fail to detect anomalies that rely on remote node connections (top-left) or group-wise interactions within densely connected substructures (bottom-left). In addition, simply stacking more GNN layers often leads to over-smoothing and degraded expressive power. Therefore, effectively modeling high-order interactions and long-range dependencies is essential for graph-level anomaly detection.

To address these issues, we propose **Hy**perFormer and **G**ene**R**ative–C**O**ntrasti-**VE** learning (HyGROVE) for unsupervised graph-level anomaly detection (HyGROVE). We first design a generative contrastive view construction module that models node-level latent distributions and samples multiple node embeddings to form semantically consistent views, avoiding heuristic perturbations (e.g., node deletion or feature masking). We then introduce a hypergraph transformation module to capture high-order interactions beyond pairwise relations, combined with a transformer to model global dependencies and long-range semantics. Finally, a multi-level graph contrastive learning module performs hierarchical contrastive learning at both node and graph levels, together with an adaptive scoring scheme for anomaly detection. **Our major contributions are summarized as follows**:

(1) We propose HyGROVE, a novel framework that introduces generative contrastive view learning for unsupervised graph-level anomaly detection. Instead of relying on heuristic perturbations, our method explicitly models node-level latent distributions and samples multiple semantically consistent views, fundamentally ensuring the preservation of critical anomaly signals during view generation.

(2) We design a hypergraph-based transformer to capture high-order interactions beyond pairwise relations and model long-range dependencies, addressing the limitations of shallow GNNs in representing complex structural patterns for graph-level anomaly detection.

(3) We conduct extensive experiments on 11 real-world datasets against 10 representative baselines, demonstrating the effectiveness and robustness of HyGROVE for unsupervised graph-level anomaly detection.

II. Related Work

A. Graph-Level Anomaly Detection

Graph-level anomaly detection aims to identify abnormal graphs by modeling global structures and node attributes. OCGIN [10] uses one-class GIN to compact normal graph representations, while OCGTL [11] extends this via neural transformation learning. GlocalKD [3] applies knowledge distillation to compute anomaly scores, and GOOD-D [5] leverages multi-level contrastive learning. TUAF [12] models triple-level edge–node relations, CVTGAD [13] exploits transformers for multi-view interactions, and SIGNET [14] introduces dual hypergraphs for interpretability. GOODAT [15] focuses on data-centric test-time detection via graph masking.

B. Graph Contrastive Learning

Contrastive learning learns expressive representations by maximizing mutual information between similar instances and is widely used in graph-level anomaly detection. DGI [16] introduces local mutual information. GCA [9] and GraphCL [17] enhance robustness via adaptive and multi-type augmentations. GLADC [2] applies contrastive objectives for anomaly discrimination. Building on this, GOOD-D [5] employs non-perturbative augmentation and multi-level contrast, and CVTGAD [13] further extends it with transformer-based multi-view modeling to capture complex relational dependencies.

C. Generative Methods in Graph Anomaly Detection

Generative methods improve anomaly detection by modeling latent distributions and generating synthetic samples. VGAE [18] and DOMINANT [19] learn unsupervised graph representations via reconstruction, while GAN-based models such as GraphGAN [20] and NetRA [21] capture connectivity patterns through adversarial training. Building on these ideas, AEGIS [22], GAAN [23], and GGAD [24] further enhance graph anomaly detection by creating pseudo-anomalies or structure-aware outliers, demonstrating the effectiveness of generative learning in modeling complex anomalies.

III. Preliminaries

A graph is denoted as $G = (V, E)$, where V is the set of nodes and E is the set of edges, with $N = |V|$ being the number of nodes, $M = |E|$ being the number of edges. The topology information of G is represented by an adjacency matrix $A \in \mathbb{R}^{N \times N}$, where N is the number of nodes. $A_{i,j} = 1$ if there is an edge between node v_i and node v_j, otherwise, $A_{i,j} = 0$. An attributed graph is denoted as $G = (V, E, X)$, where $X \in \mathbb{R}^{N \times d}$ represents the feature matrix of node features. Each row of X represents a node's feature vector with d dimensions. The graph set is denoted as $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$, where m is the total number of graphs. We study unsupervised graph-level anomaly detection, where HyGROVE is trained on normal graphs only and detects anomalous graphs at inference. The overall framework is shown in Figure 2 and Algorithm 1.

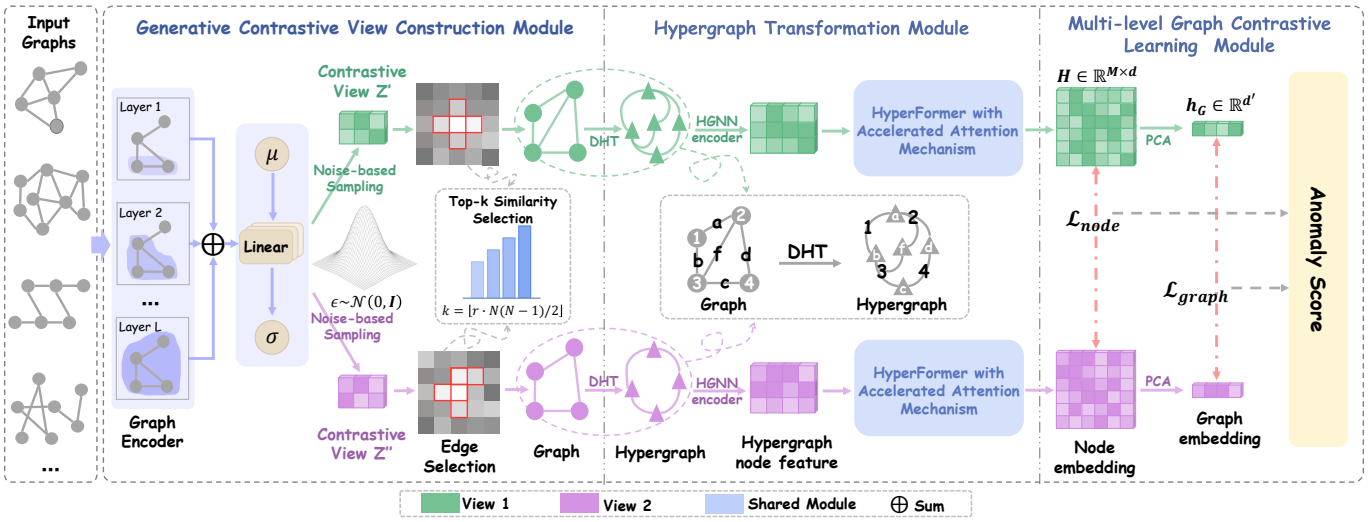


Fig. 2: The overall pipeline of the proposed HyGROVE model. The detailed structure of the HyperFormer with Accelerated Attention Mechanism is shown in Figure 3.

IV. Methodology

A. Generative Contrastive View Construction

To generate contrastive views for each graph, we draw inspiration from the modeling paradigm of the Variational Autoencoder (VAE) [25] by introducing a latent variable distribution learning mechanism.

Given the adjacency matrix A of a graph and the initial node embeddings E_0, our objective is to learn a probabilistic latent representation Z that can reconstruct the input graph structure: $\hat{A} \sim p_\theta(A \mid Z)$. Similar to the variational autoencoder (VAE) framework, we adopt variational inference $q_\phi(Z \mid A, E_0) = \prod_{i=0}^{n-1} q_\phi(z_i \mid A, E_0)$ to approximate the posterior distribution of $p_\theta(A \mid Z)$. To enforce a structured and continuous contrastive space, we add a KL regularizer:

$$\mathcal{L}_{\mathrm{KL}} = \sum\nolimits_i \mathrm{KL}[q_\phi(z_i \mid A, E_0) \,\|\, p(z_i)], \tag{1}$$

where $q_\phi(z_i|A, E_0)$ denotes the approximate posterior distribution parameterized by the encoder and $p(z_i)$ is the standard Gaussian prior. We encode each node i as a multivariate Gaussian distribution conditioned on the graph structure and initial node features as:

$$q_\phi(\mathbf{z}_i \mid A, E_0) = \mathcal{N}(\mathbf{z}_i \mid \mu_\phi(i), \mathrm{diag}(\sigma_\phi^2(i))), \tag{2}$$

where $\mu_\phi(i)$ and $\sigma_\phi(i)$ are the mean and standard deviation vectors for node i. To exploit graph structure, we employ Graph Neural Networks (GNNs) to estimate them: $\mu = GNN(A, E_0, \phi_\mu), \sigma = GNN(A, E_0, \phi_\sigma)$ where ϕ_μ and ϕ_σ denote the learnable parameters on graph inference. Then, we adopt a GNN encoder to perform structure-aware aggregation over node features, updating the mean vectors for each node i as follows:

$$\mu_i^l = \sum\nolimits_{j \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_j|}} \mu_j^{l-1}, \tag{3}$$

where μ_i^l and μ_j^{l-1} are corresponding means on l^{th} and $(l-1)^{th}$ graph convolution layer, $\mathcal{N}_i$ and $\mathcal{N}_j$ are the connected neighbors for node i and node j. We take E_0 as the initial mean embedding μ_0. Through L layers of graph convolution, we obtain $L+1$ outputs $[\mu_0, \mu_1, ..., \mu_L]$. These outputs are fused to compute the final mean and variance as follows:

$$\mu = \frac{1}{L} \sum\nolimits_{l=1}^{L} \mu^l, \sigma = MLP(\mu), \tag{4}$$

the variance is computed by passing the fused mean through a single-layer MLP: $\sigma = exp(\mu \mathrm{W} + \mathrm{b})$, where $\mathrm{W} \in \mathbb{R}^{d \times d}$ and $\mathrm{b} \in \mathbb{R}^d$ are learnable parameters. After obtaining μ_i and σ_i, we sample $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$. However, since the sampling operation is non-differentiable, we adopt the reparameterization trick to make the training process end-to-end optimizable:

$$\mathbf{z}_i = \mu_i + \sigma_i \cdot \epsilon, \tag{5}$$

where $\varepsilon \sim \mathcal{N}(0, \mathrm{I})$ is a normal Gaussian noise. These sampled latent vectors are then used to construct contrastive views.

Instead of manual augmentations, we construct contrastive views by sampling from the learned distribution. Concretely, for each node i, we generate $\mathbf{z}'$ and $\mathbf{z}''$ as follows:

$$\mathbf{z}_i' = \mu_i + \sigma_i \cdot \epsilon', \mathbf{z}_i'' = \mu_i + \sigma_i \cdot \epsilon'', \tag{6}$$

where $\epsilon', \epsilon'' \sim \mathcal{N}(0, \mathbf{I})$ are two random normal noises. These sampled embeddings serve as contrastive views for the downstream contrastive learning objective.

Unlike VAEs that reconstruct a dense probabilistic adjacency matrix, we retain only the top-ranked edges based on embedding similarity, which preserves salient structure and reduces noise. Specifically, we compute a pairwise similarity matrix $S = \mathrm{sigmoid}(ZZ^\top)$, where Z is the matrix of latent node embeddings.

For each graph in the batch, we retain only the top-k edges with the highest similarity scores, where $k = \lfloor r \cdot N(N-1)/2 \rfloor$,

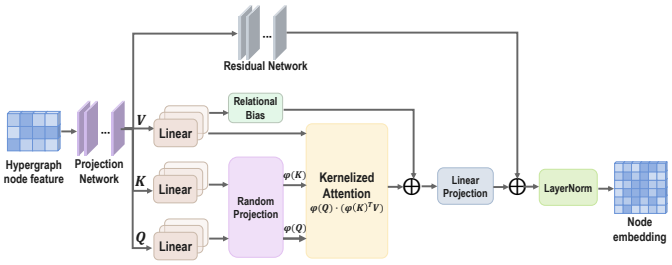


Fig. 3: The detailed structure of the HyperFormer with Accelerated Attention Mechanism.

$r \in [0, 1)$ controls edge sparsity. with r empirically set per dataset (mostly $r = 0.7$). The training procedure optimizes the model without using labels in gradient updates. The resulting edge set is then used to construct a new adjacency matrix $\hat{A}$ for each view. We encourage the two views to reconstruct the original topology, and define the reconstruction loss as:

$$\mathcal{L}_{\text{recon}} = \frac{1}{2}\left(\|A - \hat{A}'\|_F^2 + \|A - \hat{A}''\|_F^2\right), \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm, and A is the original adjacency matrix. Finally, the overall loss is given by:

$$\mathcal{L}_{view} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{KL}}. \tag{8}$$

B. Hypergraph Transformation

To capture high-order dependencies, given an original graph G, we construct a dual hypergraph G^* via DHT [14], where each edge becomes a dual node and each original node forms a hyperedge linking its incident edges (Fig. 2). This design enables efficient information exchange among multi-hop nodes, thereby capturing long-range dependencies with fewer propagation steps.

The node features of G^*, denoted as $X^* \in \mathbb{R}^{M \times d_f}$, are initialized using original edge attributes or features derived from node or geometric properties. Unlike a conventional graph transformer operating on nodes, our model applies attention over the dual nodes (i.e., original edges), facilitating higher-order dependency modeling across multi-hop connections.

To obtain richer feature representations, we employ a Hypergraph Transformer [26] to capture global correlations. This model utilizes a kernel-based approximation of multi-head attention, enabling efficient information aggregation within an implicitly constructed graph space. Let $\mathbf{H}^{(l)}$ denote the embeddings at the l-th hypergraph attention layer, where $h_i^{(l)}$ represents the hidden representation of instance i, and $\mathcal{A}^{(l)} \in \mathbb{R}^{M \times M}$ is the softmax attention matrix at the l-th layer. The attention matrix $\tilde{A}^{(l)}$ is computed as follows:

$$\mathcal{A}_{ij}^{(l)} = \frac{\exp\left(\left(h_i^{(l)} W_Q^{(l)}\right)\left(h_j^{(l)} W_K^{(l)}\right)^\top\right)}{\sum_{k=1}^{M} \exp\left(\left(h_i^{(l)} W_Q^{(l)}\right)\left(h_k^{(l)} W_K^{(l)}\right)^\top\right)}, \tag{9}$$

Here, $W_Q^{(l)}$, $W_K^{(l)}$, and $W_V^{(l)}$ are learnable projection matrices for queries, keys, and values at the l-th layer. After computing attention scores between inputs, the representation of instance l is updated as follows:

$$h_i^{(l+1)} = \sum_{j=1}^{M} \mathcal{A}_{ij}^{(l)} \cdot (h_j^{(l)} W_V^{(l)}). \tag{10}$$

We introduce a structure-aware link loss to regularize the attention weights, encouraging them to align with the connectivity patterns of the hypergraph by:

$$\mathcal{L}_{\text{link}} = -\frac{1}{|C|} \sum_{(i,j)\in C} \log \mathcal{A}_{ij} \cdot \frac{1}{\sqrt{d_j}}, \tag{11}$$

where C denotes node pairs connected in the hypergraph. We stack L layers with residual connections, layer normalization, and activation to obtain $\mathbf{h}_i$.

C. Multi-level Graph Contrastive Learning

After obtaining the node representations $\mathbf{h}_i$, the embeddings of all nodes in each graph form a matrix $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_M] \in \mathbb{R}^{M \times d}$, where M is the number of nodes in the graph and d is the embedding dimension.To obtain more compact graph-level representations, we apply Principal Component Analysis (PCA) to the learned embeddings, denoted as $\mathbf{H}' = \text{PCA}(\mathbf{H}) \in \mathbb{R}^{M \times d'}$, where $d' < d$ is the reduced dimension after PCA. Finally, we compute the mean vector of all nodes in the reduced space as the graph-level representation:

$$\mathbf{h}_G = \frac{1}{M} \sum_{i=1}^{M} \mathbf{h}_i', \tag{12}$$

where $\mathbf{h}_i'$ is the PCA-reduced embedding of node i, i.e., the i-th row of $\mathbf{H}'$. Following [5], [13], we design an adaptive strategy considering both node and graph-level loss to calculate the anomaly score.

Node-level Contrast. For a given input graph G, we define a node-level contrastive loss to maximize consistency across views:

$$L_{node} = \frac{1}{|\mathcal{B}|} \sum_{G_j \in \mathcal{B}} \frac{1}{2|\mathcal{V}_{G_j}|} \sum_{v_i \in \mathcal{V}_{G_j}} \left[l\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right) + l\left(\mathbf{h}_i^{v_2}, \mathbf{h}_i^{v_1}\right)\right], \tag{13}$$

$$l\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right) = -\log \frac{e^{sim\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right)/\tau}}{\sum_{v_k \in \mathcal{V}_{G_j} \setminus v_i} e^{sim\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right)/\tau}}. \tag{14}$$

In Eq. (15), $\mathcal{B}$ denotes the training/testing batch and $\mathcal{V}_{G_j}$ represents the node set of graph G_j. The computations of $l\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right)$ and $l\left(\mathbf{h}_i^{v_2}, \mathbf{h}_i^{v_1}\right)$ are symmetric. We present the calculation of $l\left(\mathbf{h}_i^{v_1}, \mathbf{h}_i^{v_2}\right)$ in Eq. (16), using cosine similarity as the similarity function between views.

Graph-level Contrast. Similarly, we define a graph-level contrastive loss to maximize agreement between graph representations across views:

$$L_{graph} = \frac{1}{2|\mathcal{B}|} \sum_{G_i \in \mathcal{B}} \left[l\left(\mathbf{h}_{G_i}^{v_1}, \mathbf{h}_{G_i}^{v_2}\right) + l\left(\mathbf{h}_{G_i}^{v_2}, \mathbf{h}_{G_i}^{v_1}\right)\right], \tag{15}$$

$$l\left(\mathbf{h}_{G_i}^{v_1}, \mathbf{h}_{G_i}^{v_2}\right) = -\log \frac{e^{sim\left(\mathbf{h}_{G_i}^{v_1}, \mathbf{h}_{G_i}^{v_2}\right)/\tau}}{\sum_{G_j \in \mathcal{B} \setminus G_i} e^{sim\left(\mathbf{h}_{G_i}^{v_1}, \mathbf{h}_{G_i}^{v_2}\right)/\tau}}. \tag{16}$$

Similar to the node-level loss, the computation of $l\left(\mathbf{h}_{G_i}^{v_1}, \mathbf{h}_{G_i}^{v_2}\right)$ is the same as that of $l\left(\mathbf{h}_{G_i}^{v_2}, \mathbf{h}_{G_i}^{v_1}\right)$. We employ an adaptive loss function during training to guide the optimization process:

$$\mathcal{L} = (\sigma_{node})^{\alpha} L_{node} + (\sigma_{graph})^{\alpha} L_{graph} + \mathcal{L}_{view} - \mathcal{L}_{link}, \tag{17}$$

where σ_{node} and σ_{graph} are the standard deviations of the prediction errors at the node and graph levels, respectively. α is a hyperparameter that satisfies $\alpha \geq 0$. and the subtraction of $\mathcal{L}_{link}$ aligns with its negative definition to encourage structure-aware regularization. During inference, we adopt a normalization strategy to obtain the final anomaly score:

$$score_{G_i} = \frac{L_{nodeG_i} - \mu_{node}}{\sigma_{node}} + \frac{L_{graphG_i} - \mu_{graph}}{\sigma_{graph}}, \tag{18}$$

where μ_{node} and μ_{graph} are the mean values of prediction errors at the node level and graph level.

D. Time Complexity Analysis

While both data augmentation and hypergraph construction are performed once during the pre-processing stage, their cost is negligible compared to the encoder and loss computation. Let m be the number of graphs, n/e the average nodes/edges, M the average hyperedges, B the batch size, d the latent dimension, d_{in} the input dimension, and L_g, L_h, L_t the numbers of graph, hypergraph, and Transformer layers. The graph, hypergraph and hypergraph transformer encoders have complexities $O(mL_g ed + mL_g nd^2 + mndd_{in})$, $O(mL_h Md + mL_h nd^2 + mMdd_{in})$, and $O(mL_t ed + mL_t nd^2 + mndd_{in})$. The node- and graph-level contrastive losses cost $O(mn^2 d)$ and $O(mBd)$. Ignoring constants and lower-order terms, the overall complexity is $O\big(mLd(e + nd + M) + md(nd_{in} + Md_{in} + n^2 + B)\big)$ with $L = L_g + L_h + L_t$. While these introduce additional computational cost compared to simpler baselines, it remains manageable for typical datasets and can be further optimized in future work.

V. Experiments

A. Experimental Settings

Datasets and Baselines. We use 11 publicly available datasets from the TUDataset [1], covering bioinformatics, social networks, and small molecules. Following [3], [5], minority or truly anomalous classes are treated as anomalies, and the remaining classes are considered normal, with only normal graphs used for training [3], [5], [10]. Dataset statistics are reported in Table I. We compare HyGROVE with 10 representative baselines, including kernel-based methods (WL-OCSVM, WL-iF, PK-OCSVM, PK-iF) [27]–[30], contrastive learning methods (InfoGraph-iF, GraphCL-iF) [17], [31], test-time training method (GOODAT) [15], and end-to-end models (OCGIN, GLocalKD, GOOD-D) [3], [5], [10].

Evaluation and Implementation. We evaluate performance using AUC, a standard metric for graph-level anomaly detection [2], [3], [5], [13], [15], where higher values indicate better detection. HyGROVE is implemented in PyTorch.

Algorithm 1: HyGROVE

Input: Graph set: $\mathcal{G} = \{G_1, G_2, \ldots, G_m\}$
Output: The anomaly scores for each graph $score_G$

Initialize:
Load dataset $\mathcal{G}$ and split into training/testing sets;
Initialize model modules and training settings.
Training Phase: for *epoch* $= 1$ **to** *epochs* **do**
  foreach *batch in train loader* **do**
    Compute mean μ and variance σ of node embeddings (Eq. 4);
    Sample two contrastive node embeddings $\mathbf{z}_i'$ and $\mathbf{z}_i''$ (Eq. 6);
    Generate adjacency matrices $\hat{A}'$ and $\hat{A}''$ by top-k similarity selection and compute $\mathcal{L}_{\text{view}}$ (Eq. 8);
    Construct a dual-hypergraph for each view using DHT and compute initial hypergraph node features (Sec. IV-B);
    Apply the Hypergraph Transformer to update node representations and compute link loss $\mathcal{L}_{\text{link}}$ (Eq. 10–11);
    Compute graph-level embeddings h_G from node embeddings using PCA by (Eq. 12);
    Compute node-level loss L_{node}(Eq. 13);
    Compute graph-level loss L_{graph}(Eq. 15);
    Compute final loss $\mathcal{L}$ with adaptive weights (Eq. 17);
Inference Phase: for G_i *in Graph set* $\mathcal{G}$ **do**
  Calculate anomaly scores (Eq. 18).

TABLE I: Statistics of the TUDataset [1] datasets used.

Dataset	PROTEINS-full	ENZYMES	AIDS	DHFR	BZR
Graphs	1113	600	2000	467	405
Avg.Nodes	30.06	32.63	15.69	42.43	35.75
Avg.Edges	72.82	62.14	16.20	44.54	38.36

Dataset	COX2	DD	COLLAB	HSE	p53	PPAR-gamma
Graphs	467	1178	5000	8417	8903	8451
Avg.Nodes	41.22	284.32	74.49	16.89	17.92	17.38
Avg.Edges	43.45	715.66	2457.78	17.23	18.34	17.72

B. Overall Performance

The overall results on 11 datasets are summarized in Table II. HyGROVE achieves the best performance among all methods, ranking first on five datasets and second on four, and obtains the best average rank overall.

We observe that graph kernel-based methods perform the worst among all baselines, likely due to their limited ability to capture informative graph features. In contrast, graph contrastive learning-based methods show stronger performance in anomaly detection tasks. The results show that integrating generative contrastive view construction with hypergraph-based structural modeling significantly improves the semantic consistency and expressiveness of the learned representations. These findings validate the design motivation of HyGROVE, demonstrating that it can effectively learn the underlying regularities of normal graphs and achieve superior performance in unsupervised graph-level anomaly detection.

C. Ablation Study

To validate HyGROVE, we conduct ablations on 11 datasets by removing the generative view construction (w/o G; replaced

TABLE II: The performance comparison in terms of AUC (in percent, mean value ± standard deviation). The best performance is highlighted in bold, and the second-best performance is underlined. †: we report the result from [5], [15].

Method	PK-OCSVM†	PK-iF†	WL-OCSVM†	WL-iF†	InfoGraph-iF†	GraphCL-iF†	OCGIN†	GLocalKD†	GOOD-D†	GOODAT†	HyGROVE
PROTEINS-full	50.49±4.92	60.70±2.55	51.35±4.35	61.36±2.54	57.47±3.03	60.18±2.53	70.89±2.44	77.30±5.15	71.97±3.86	**77.92±2.37**	76.47±1.54
ENZYMES	53.67±2.66	51.30±2.01	55.24±2.66	51.60±3.81	53.80±4.50	53.60±4.88	58.75±5.98	61.39±8.81	63.90±3.69	52.33±4.74	**65.57±7.59**
AIDS	50.79±4.30	51.84±2.87	50.12±3.43	61.13±0.71	70.19±5.03	79.72±3.98	78.16±3.05	93.27±4.19	97.28±0.69	95.50±0.99	**98.79±0.90**
DHFR	47.91±3.76	52.11±3.96	50.24±3.13	50.29±2.77	52.68±3.21	51.10±2.35	49.23±3.05	56.71±3.57	62.67±3.11	61.52±2.86	**63.07±1.96**
BZR	46.85±5.31	55.32±6.18	50.56±5.87	52.46±3.30	63.31±8.52	60.24±5.37	65.91±1.47	69.42±7.78	**75.16±5.15**	64.77±3.87	70.27±6.64
COX2	50.27±7.91	50.05±2.06	49.86±7.43	50.27±0.34	53.36±8.86	52.01±3.17	53.58±5.05	59.37±12.67	**62.65±8.13**	59.99±9.76	62.57±8.13
DD	48.30±3.98	71.32±2.41	47.99±4.09	70.31±1.09	55.80±1.77	59.32±3.92	72.27±1.83	80.12±5.24	73.25±3.19	77.62±2.88	**80.39±1.61**
COLLAB	49.59±2.24	50.49±1.72	52.60±2.56	50.69±0.32	46.27±0.73	47.61±1.29	60.70±2.97	52.94±0.85	**72.08±0.90**	44.91±0.87	57.62±1.17
HSE	57.02±8.42	56.87±10.51	62.72±10.13	53.02±5.12	53.56±3.98	51.18±2.71	64.84±4.70	59.48±1.44	**69.65±2.14**	63.05±0.90	67.46±6.89
p53	46.74±4.88	50.69±2.02	54.59±4.46	50.85±2.16	52.66±1.95	53.29±2.32	58.50±0.37	64.20±0.81	62.99±1.55	63.27±0.04	**66.47±2.16**
PPAR-gamma	53.94±6.94	45.51±2.58	57.91±6.13	49.60±0.22	51.40±2.53	50.30±1.56	**71.19±4.28**	64.59±0.67	67.34±1.71	68.23±1.54	69.72±2.79
AVG.RANK	9.27	8.45	8.18	8.27	7.54	7.82	4.54	3.45	2.36	4.36	**1.73**

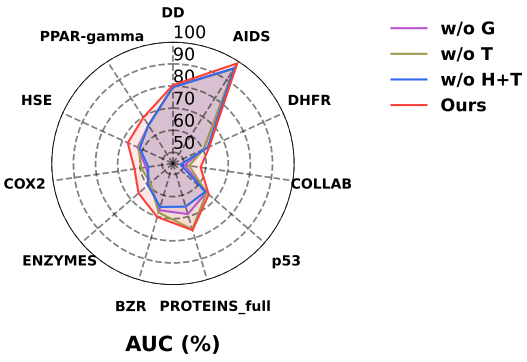


Fig. 4: Component-wise ablation results across all datasets.

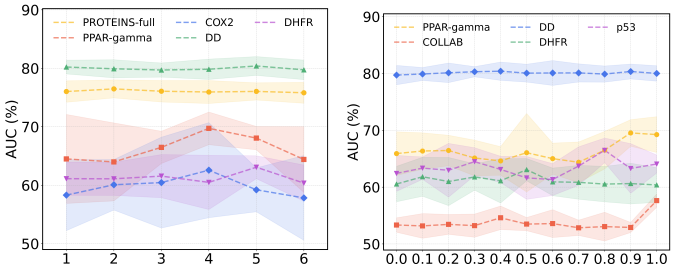
with SimGCL feature perturbation [32]), removing the transformer in hypergraph transformation (w/o T), and removing the entire hypergraph transformation module (w/o H+T).

As shown in Figure 4, HyGROVE performs best overall. Removing generative view construction (w/o G) degrades performance, indicating that sampling views from a learned latent distribution is more effective than heuristic perturbations. Excluding the transformer (w/o T) causes a moderate decline, suggesting that global modeling helps capture long-range dependencies. Removing hypergraph transformation (w/o H+T) leads to the largest drop, highlighting the necessity of hypergraph-based reasoning for high-order interaction modeling.

D. Hyperparameter Analysis and Visualization

The Number of Encoder Layers. To study the effect of encoder depth in the Generative Contrastive View Construction module, we evaluate HyGROVE on five representative datasets. As shown in Figure 5(a), the model performs best when using 2–5 layers, while further increasing or decreasing the depth brings no clear gains. With only 1 layer, the encoder has limited capacity to capture graph features, whereas 6 layers may cause performance degradation due to over-smoothing.

Self-adaptiveness Strength α. We analyze the sensitivity of HyGROVE to α, which controls the adaptive weighting between graph-level and node-level contrastive losses, by varying it from 0 to 1. Figure 5(b) shows that disabling adaptiveness ($\alpha = 0$) slightly degrades performance. As α increases, HyGROVE achieves the best AUC when α is in the range of 0.3–0.9, while larger values lead to a plateau or slight decline, indicating reduced generalization. Overall, these results demonstrate the effectiveness and robustness of the adaptive loss weighting.



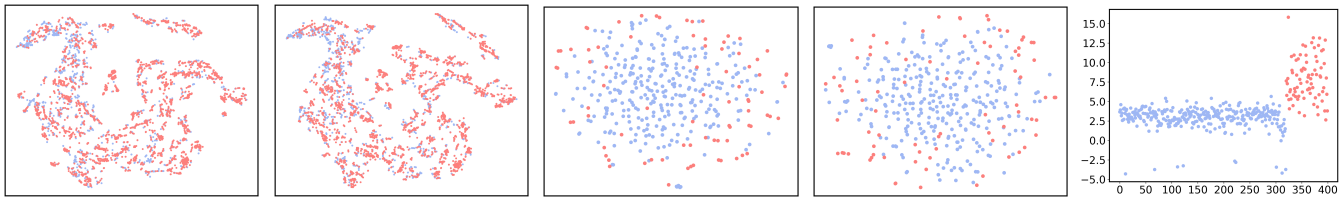
(a) The number of encoder Layers (b) Self-adaptiveness strength α

Fig. 5: AUC (%) performance with (a) varying encoder depth and (b) self-adaptiveness strength α on representative datasets.

Visualization. To further verify the effectiveness of HyGROVE, we visualize the node-level and graph-level embeddings on the AIDS using t-SNE. As shown in Figure 6, normal and anomalous graphs still show noticeable overlap in the embedding space, making them hard to separate by distribution alone. However, the resulting anomaly scores produce a clear decision boundary, indicating that HyGROVE can capture anomaly-related features through generative view construction and hypergraph-aware modeling.

VI. Conclusions

In this paper, we introduce HyGROVE, an unsupervised framework for graph-level anomaly detection. It employs generative contrastive view construction to preserve subtle anomaly signals, hypergraph transformation with transformer modeling for high-order dependencies, and multi-level contrastive learning with adaptive scoring. As the first generative-contrastive paradigm for GLAD, HyGROVE consistently surpasses 10 state-of-the-art methods on 11 benchmarks, showing strong accuracy and robustness. Future work will explore more efficient designs to mitigate computational overhead.



(a) Node-level view v_1 (b) Node-level view v_2 (c) Graph-level view v_1 (d) Graph-level view v_2 (e) Anomaly Score

Fig. 6: Visualization on AIDS dataset for different view v_1 and v_2. (blue nodes denote normal, red nodes denote anomalous.)

ACKNOWLEDGMENT

This work was supported by the Excellent Young Talents Program of the Education Department of Jilin Province (Grant No. JJKH20261683KJ).

REFERENCES

[1] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," 2020. [Online]. Available: https://arxiv.org/abs/2007.08663

[2] X. Luo, J. Wu, J. Yang, S. Xue, H. Peng, C. Zhou, H. Chen, Z. Li, and Q. Z. Sheng, "Deep graph level anomaly detection with contrastive learning," *Scientific Reports*, vol. 12, no. 1, p. 19867, 2022.

[3] R. Ma, G. Pang, L. Chen, and A. van den Hengel, "Deep graph-level anomaly detection by glocal knowledge distillation," in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 704–714.

[4] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE transactions on knowledge and data engineering*, vol. 35, no. 12, pp. 12 012–12 038, 2021.

[5] Y. Liu, K. Ding, H. Liu, and S. Pan, "Good-d: On unsupervised graph out-of-distribution detection," in *Proceedings of the sixteenth ACM international conference on web search and data mining*, 2023, pp. 339–347.

[6] J. Li, Y. Liu, Q. Xing, Q. Wang, and S. Pan, "No fear of representation bias graph contrastive learning with calibration and fusion," *Available at SSRN 4774833*.

[7] Y. Fu, J. Li, J. Liu, Q. Xing, Q. Wang, and I. King, "Hc-glad: Dual hyperbolic contrastive learning for unsupervised graph-level anomaly detection," *arXiv preprint arXiv:2407.02057*, 2024.

[8] Y. Fu, J. Li, Q. Wang, and Q. Xing, "Gladmamba: Unsupervised graph-level anomaly detection powered by selective state space model," *arXiv preprint arXiv:2503.17903*, 2025.

[9] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the web conference 2021*, 2021, pp. 2069–2080.

[10] L. Zhao and L. Akoglu, "On using classification datasets to evaluate graph outlier detection: Peculiar observations and new insights," *Big Data*, vol. 11, no. 3, pp. 151–180, 2023.

[11] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," *arXiv preprint arXiv:2205.13845*, 2022.

[12] Z. Yu, X. Wang, B. Zhang, Z. Luo, and L. Duan, "Tuaf: Triple-unit-based graph-level anomaly detection with adaptive fusion readout," in *International Conference on Database Systems for Advanced Applications*. Springer, 2023, pp. 415–430.

[13] J. Li, Q. Xing, Q. Wang, and Y. Chang, "Cvtgad: simplified transformer with cross-view attention for unsupervised graph-level anomaly detection," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2023, pp. 185–200.

[14] Y. Liu, K. Ding, Q. Lu, F. Li, L. Y. Zhang, and S. Pan, "Towards self-interpretable graph-level anomaly detection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 8975–8987, 2023.

[15] L. Wang, D. He, H. Zhang, Y. Liu, W. Wang, S. Pan, D. Jin, and T.-S. Chua, "Goodat: towards test-time graph out-of-distribution detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 14, 2024, pp. 15 537–15 545.

[16] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *arXiv preprint arXiv:1809.10341*, 2018.

[17] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.

[18] T. N. Kipf and M. Welling, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[19] K. Ding, J. Li, R. Bhanushali, and H. Liu, "Deep anomaly detection on attributed networks," in *Proceedings of the 2019 SIAM international conference on data mining*. SIAM, 2019, pp. 594–602.

[20] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[21] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 2663–2671.

[22] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence*, 2021, pp. 1288–1294.

[23] Z. Chen, B. Liu, M. Wang, P. Dai, J. Lv, and L. Bo, "Generative adversarial attributed network anomaly detection," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2020, pp. 1989–1992.

[24] H. Qiao, Q. Wen, X. Li, E.-P. Lim, and G. Pang, "Generative semi-supervised graph anomaly detection," *Advances in neural information processing systems*, vol. 37, pp. 4660–4688, 2024.

[25] Y. Yang, Z. Wu, L. Wu, K. Zhang, R. Hong, Z. Zhang, J. Zhou, and M. Wang, "Generative-contrastive graph learning for recommendation," in *Proceedings of the 46th international ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 1117–1126.

[26] Z. Liu, B. Tang, Z. Ye, X. Dong, S. Chen, and Y. Wang, "Hypergraph transformer for semi-supervised classification," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 7515–7519.

[27] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[28] M. Neumann, R. Garnett, C. Bauckhage, and K. Kersting, "Propagation kernels: efficient graph kernels from propagated information," *Machine learning*, vol. 102, pp. 209–245, 2016.

[29] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[30] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *2008 eighth ieee international conference on data mining*. IEEE, 2008, pp. 413–422.

[31] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," *arXiv preprint arXiv:1908.01000*, 2019.

[32] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 1294–1303.