Next-Generation GRAM: From Lightweight Efficiency to Temporal Graph Understanding

Zahra Chizari [*], Zahra Maleki [*], Seyed Moein Ayyoubzadeh[*]
[*]Computer Engineering, Sharif University of Technology

***Abstract*—Graph Anomaly Detection (GAD) plays a vital role in identifying abnormal nodes, edges, or substructures across domains such as cybersecurity, financial fraud detection, and social network analysis. While Graph Neural Networks (GNNs) have significantly advanced GAD, challenges remain in balancing *interpretability*, *computational efficiency*, and handling *temporal dynamics*. This paper extends the interpretable GRAM framework through three novel variants: (i) Faster-GRAM, a lightweight, scalable architecture employing Transformer-based convolutions, bilinear/MLP decoders, and latent normalization to achieve up to $26\times$ parameter reduction and $2.7\times$ speed-up without sacrificing detection accuracy; (ii) Temporal-GRAM, integrating a GCN-LSTM encoder and Variational Autoencoder (VAE) to capture evolving structural patterns and detect anomalies in dynamic graphs while preserving gradient-based interpretability; and (iii) a multi-label classification extension for domains such as protein–protein interaction networks, enabling per-class attention maps and adaptive decision thresholds. Extensive experiments demonstrate that Faster-GRAM consistently outperforms baseline and state-of-the-art models in Recall@10, AP, AUC, NDCG, and Precision@10, while Temporal-GRAM effectively models time-dependent anomalies and the multi-label extension delivers competitive micro- and macro-level performance on imbalanced datasets. Finally, we present Hybrid-GRAM (PTC), an encoder-only variant that repurposes the GRAM encoder as a feature extractor and couples it with conventional classifiers for graph-level prediction on the PTC dataset. While this hybrid abandons gradient-based interpretability and decoder reconstruction, it attains up to 96.7% accuracy (Logistic Regression) with 5-fold CV F1 of 0.963 $\pm$ 0.009. We analyze the trade-offs and threats to validity when pretraining is absent and interpretability is sacrificed. The proposed methods collectively advance interpretable, efficient, and temporally aware GAD, offering practical deployment in large-scale, real-time, and multi-label settings.**

***Index Terms*—Graph Anomaly Detection, Graph Neural Networks, Interpretability, Temporal Graphs, TransformerConv, Variational Autoencoder**

I. Introduction

Graph Anomaly Detection (GAD) is a critical task across diverse application domains, aiming to identify anomalous instances—such as nodes, edges, subgraphs, or entire graphs—that significantly deviate from established normal patterns. Applications of GAD span a wide range, including cybersecurity, financial fraud detection, social network analysis, and fault diagnosis in complex systems. The emergence of deep learning, particularly Graph Neural Networks (GNNs), has introduced a powerful paradigm for GAD by leveraging their ability to model intricate structural and attribute information in graph data. However, applying GNNs to GAD remains challenging; phenomena such as over-smoothing can result in indistinguishable representations for normal and anomalous instances, thereby limiting their effectiveness in real-world deployments [1].

The Gradient Attention Map (GRAM) model represents a notable step forward in interpretable GAD, employing gradient-based attention mechanisms to highlight influential graph components. This work extends GRAM through two novel variants: **Fast GRAM**, a lightweight and efficient anomaly detection framework utilizing TransformerConv, bilinear decoding, and contrastive learning; and **Temporal GRAM**, designed for dynamic or temporal graphs, incorporating a GCN-LSTM encoder and a Variational Autoencoder (VAE) framework [2]. Beyond anomaly detection, we also explore a pragmatic encoder-only variant we call *Hybrid-GRAM (PTC)*: a GRAM-style encoder used purely for feature extraction, combined with a bank of traditional classifiers for graph-level prediction on the PTC dataset. This hybrid deliberately removes decoder paths and gradient-based explanations in exchange for a simpler pipeline and strong accuracy. We report stateful performance and, crucially, discuss threats to validity stemming from the lack of a pre-trained GRAM encoder.

Advancing GAD models along the dimensions of *interpretability*, *efficiency*, and *temporal dynamics* is essential. Interpretability fosters trust and enables actionable insights—particularly in high-stakes scenarios—by revealing the reasoning behind predictions and identifying critical substructures. Efficiency and scalability are vital for deploying GAD models in large-scale, resource-constrained, real-time environments. Finally, as many real-world graphs are inherently dynamic, the ability to capture temporal evolution is crucial, since anomalies may manifest differently over time [3].

This work provides a focused review of relevant literature, emphasizing methods aligned with the conceptual motivations and architectural design of the proposed GRAM extensions. Special attention is given to temporal GAD approaches, Grad-CAM extensions for GNNs, and classification techniques expanded from the same architecture, while Fast GRAM focuses on efficiency rather than interpretability. For reproducibility, we release the implementation at [1].

II. Related Work

Graph Anomaly Detection (GAD) has witnessed significant advancements not only in improving detection accuracy but

[1]https://github.com/smoeina/NxtGRAM

also in enhancing the interpretability and efficiency of models. This section reviews relevant literature on interpretable GAD methods, gradient- and attention-based explainability, self-interpretable models, evaluation frameworks for explanations, and lightweight architectures designed for scalable and practical deployment.

A. Interpretable Graph Anomaly Detection

Interpretable GAD methods aim to provide human-understandable reasoning behind anomaly detection decisions, addressing the "black-box" nature of deep graph models. Recent studies propose prototype-based and causality-aware frameworks to improve transparency and trustworthiness. For example, Yang et al. [4] introduce *GLADPro*, a global interpretable graph-level anomaly detection framework leveraging prototype representations to identify anomalous patterns while offering precise, case-level explanations. Similarly, Luo et al. [5] develop *CA-GCN*, a causality-aware graph convolutional network for vehicular network anomaly detection, explicitly modeling causal relationships to enhance interpretability.

Model architectures incorporating structural priors have also been explored. Liu et al. [6] propose *U-GraphFormer*, an unsupervised transformer-based approach for structural damage detection, producing interpretable severity assessments based on reconstruction errors. Chandra and Manhas [7] employ meta-learning with dependency graphs for scalable, interpretable anomaly detection in self-healing databases. Jiang and Wu [8] leverage spatio-temporal causal inference to improve interpretability and performance in distributed systems.

Survey works, such as Chunawala et al. [9], comprehensively cover interpretable GAD methods, highlighting the trade-offs between accuracy, scalability, and explainability. Lightweight designs, like Wang et al.'s [10] *ProvGOutLiner*, focus on process behavior features in provenance graphs for interpretable yet computationally efficient detection. Moreover, domain-specific interpretable GAD has been applied in logistics (Aldhahri et al. [11]), medical anomaly detection (Wu [12]), and general graph database monitoring (Chetoui et al. [13]).

Overall, interpretable GAD research trends converge towards hybrid approaches that integrate causal reasoning, prototype learning, and attention mechanisms, enabling not only accurate detection but also actionable and comprehensible explanations.

B. Lightweight and Efficient Models for Graph Anomaly Detection

Lightweight and efficient models for Graph Anomaly Detection (GAD) aim to deliver high detection accuracy while reducing computational and memory overhead, enabling deployment in resource-constrained environments such as IoT devices, edge computing platforms, and large-scale dynamic graphs. These methods typically adopt compact neural architectures, efficient graph filtering techniques, and federated or distributed learning paradigms to maintain scalability without sacrificing interpretability or robustness.

Ye et al. [14] introduce *HCT*, a hierarchical contrastive learning framework that reuses pre-trained model weights while adding a lightweight, trainable GCN layer to support transferable and efficient GAD across heterogeneous graph domains. Wo et al. [15] propose *LH-GNN*, a local homophily-aware GNN with adaptive polynomial filters designed for scalable anomaly detection, achieving significant reductions in computational cost for large-scale graphs.

For provenance graph analysis, Wang et al. [10] present *ProvGOutLiner*, a lightweight detection model leveraging process behavior features for efficient host-based intrusion detection. In industrial IoT contexts, Athar et al. [16] propose *DuAtt*, a dual-layer attestation scheme combining dependency graph modeling with efficient anomaly detection for programmable logic controllers (PLCs). Son et al. [17] extend this to healthcare IoT with *AutoKAN*, a federated lightweight autoencoder-based GAD framework for constrained diabetes monitoring systems.

Edge-oriented solutions include Li [18], who employs tiny machine learning for real-time opinion anomaly detection using micro-GNNs, and Asad et al. [19] who design a probabilistic adversarial autoencoder with a lightweight reconstruction-based novelty detection mechanism. In IoT security, Gouda and Ariunaaa [20] apply graph-based machine learning with reduced model complexity to maintain effectiveness under strict resource constraints.

Augmentation-based optimization is explored by Raj et al. [21], who integrate DCGAN-driven minority class augmentation with a lightweight YOLO framework for photovoltaic defect localization in edge deployment scenarios. Furthermore, Kapileswar and Simon [22] propose *PrivAudit*, a federated transformer-driven GAD method with differential privacy, designed for secure and efficient anomaly detection in cloud infrastructures.

Collectively, these approaches demonstrate that lightweight architectures, combined with graph-aware optimizations and distributed learning, can enable efficient, scalable, and accurate GAD suitable for both centralized and resource-constrained deployment environments.

III. Methodology

A. Faster-GRAM

The proposed *Faster-GRAM* model is a lightweight and computationally efficient extension of the original GRAM architecture [23], specifically designed to address the computational bottlenecks of graph anomaly detection (GAD) in large-scale and real-time settings. While GRAM integrates dual decoder paths and GCN-based reconstruction to maximize interpretability, its complexity and parameter count hinder fast deployment. Faster-GRAM retains the core idea of a variational graph autoencoder with separate μ and $\log \sigma$ parameterizations, but introduces several architectural simplifications and optimizations without sacrificing detection accuracy.

1) Motivation: The original GRAM employs:

- **GCN Encoder:** A stack of graph convolutional layers to encode structural and attribute information.
- **Dual μ and $\log \sigma$ Encoders:** Linear transformations producing the mean and log-variance vectors.
- **Dual Decoders:** Attribute decoder (*Linear* + *GCN*) and structure decoder (*Linear* + *GCN*) paths, reconstructing node features $\hat{\mathbf{X}}$ and adjacency structure $\hat{\mathbf{A}}$.

Although this design enhances interpretability through gradient-based saliency maps, the dual GCN decoders introduce considerable inference latency and increase the memory footprint, particularly when processing dense or large-scale graphs.

2) Architectural Enhancements: Faster-GRAM modifies the above design through the following enhancements:

a) Optimized GNN Encoder: The multi-layer GCN encoder in GRAM is replaced by a modular `OptimizedGNN` block, which supports *GATv2*, *GraphSAGE*, or *Transformer*-based convolution layers. This flexibility allows practitioners to choose an encoder variant that balances accuracy and computational efficiency for a given application domain.

b) Simplified Attribute Reconstruction: The attribute decoder path is reduced from a multi-stage *Linear* + *GCN* design to a *single linear layer*:

$$\hat{\mathbf{X}} = \mathbf{Z}\mathbf{W}_{\text{attr}} + \mathbf{b}_{\text{attr}} \tag{1}$$

where $\mathbf{Z}$ is the latent representation and $\mathbf{W}_{\text{attr}}$ is a trainable projection matrix. This reduces the decoder's complexity, resulting in lower inference time and parameter count.

c) Efficient Edge Reconstruction: The structure decoder is replaced by a bilinear or multi-layer perceptron (MLP) edge decoder:

$$\hat{A}_{ij} = \begin{cases} \mathbf{z}_i^\top \mathbf{W}_{\text{bil}} \mathbf{z}_j, & \text{(bilinear form)} \\ \text{MLP}([\mathbf{z}_i \,\|\, \mathbf{z}_j]), & \text{(MLP form)} \end{cases} \tag{2}$$

This modification eliminates the need for GCN decoding in the structural path, which is a major contributor to GRAM's computational load.

d) Latent Space Normalization: Before decoding, $\mathbf{Z}$ is normalized in the latent space via:

$$\mathbf{z}_i \leftarrow \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2} \tag{3}$$

L2 normalization improves stability, controls embedding scale, and often enhances contrast in the feature space, which benefits both reconstruction and anomaly scoring.

e) Integrated Negative Sampling: Negative sampling for edge reconstruction is moved directly into the output stage. This ensures that only a subset of non-existent edges is considered during training, significantly reducing computational cost for large graphs.

3) Variational Encoding Process: As in GRAM, Faster-GRAM employs a variational encoding scheme:

$$\boldsymbol{\mu} = \text{Linear}_{\mu}(\mathbf{H}) \tag{4}$$

$$\log \boldsymbol{\sigma} = \text{Linear}_{\log \sigma}(\mathbf{H}) \tag{5}$$

$$\mathbf{z}_i = \mu_i + \epsilon_i \cdot \exp(\log \sigma_i), \quad \epsilon_i \sim \mathcal{N}(0, I) \tag{6}$$

where $\mathbf{H}$ denotes the encoder output. The stochastic latent code $\mathbf{Z}$ is then L2-normalized before being passed to the decoders.

4) Training Objective: The total training loss combines attribute reconstruction loss, structure reconstruction loss, and KL divergence:

$$\mathcal{L} = \mathcal{L}_{\text{attr}} + \mathcal{L}_{\text{struct}} + \beta \cdot \mathcal{L}_{\text{KL}} \tag{7}$$

where:

$$\mathcal{L}_{\text{attr}} = \|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 \tag{8}$$

$$\mathcal{L}_{\text{struct}} = - \sum_{(i,j) \in E \cup E^-} \Big[A_{ij} \log \hat{A}_{ij} + (1 - A_{ij}) \log(1 - \hat{A}_{ij}) \Big] \tag{9}$$

$$\mathcal{L}_{\text{KL}} = -\frac{1}{2} \sum_i \left(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2\right) \tag{10}$$

Here E is the set of observed edges and E^- is the set of negatively sampled edges.

5) Architectural Comparison: A side-by-side comparison between GRAM and Faster-GRAM is shown in Fig. 1. The diagram illustrates the following major changes:

- Removal of GCN layers in the decoder paths.
- Addition of L2 normalization in the latent space.
- Flexible encoder supporting *GATv2*, *GraphSAGE*, and TransformerConv.
- Simplified attribute reconstruction via a single linear projection.
- Replacement of GCN structure decoder with bilinear/MLP-based edge reconstruction.
- Integrated negative sampling for efficient training.

6) Advantages: Through these changes, Faster-GRAM achieves:

1) **Parameter Reduction:** $\sim 26\times$ fewer trainable parameters compared to GRAM.
2) **Speed-up:** $\sim 2.7\times$ faster training and inference in our benchmarks.
3) **Memory Efficiency:** Reduced GPU memory usage due to simpler decoders.
4) **Scalability:** Practical for graphs with millions of nodes and edges.

Despite these reductions, Faster-GRAM maintains comparable detection accuracy, making it a viable choice for real-world GAD applications where both speed and accuracy are critical.

B. Temporal GRAM: Extending GRAM for Dynamic Graphs

While the original **GRAM** architecture focuses on static graphs and explains anomalies through gradient-based attention maps over a single snapshot, many real-world systems (e.g., communication networks, financial transactions, IoT device interactions) exhibit *temporal* evolution. Anomalies in such settings may only emerge when considering how the graph structure and attributes evolve over time. To address this limitation, we propose **Temporal GRAM**, a temporal

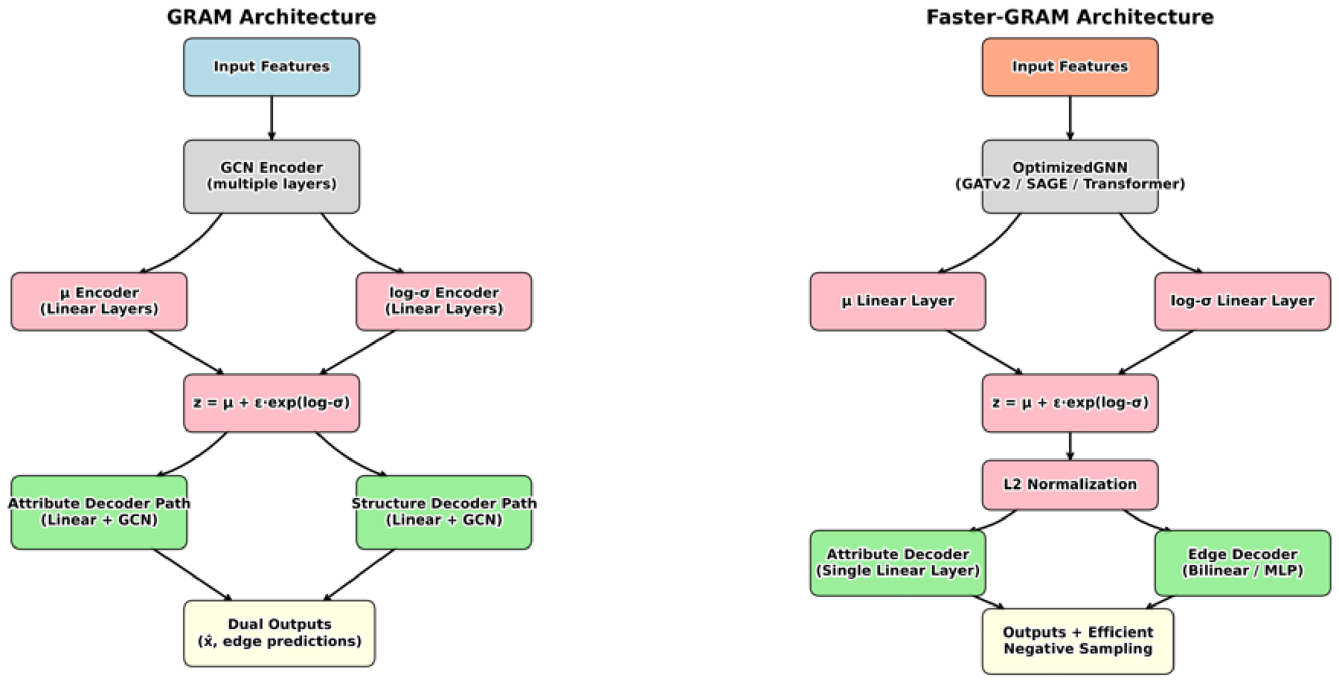

GRAM Architecture
Input Features
GCN Encoder (multiple layers)
μ Encoder (Linear Layers)
log-σ Encoder (Linear Layers)
z = μ + ε·exp(log-σ)
Attribute Decoder Path (Linear + GCN)
Structure Decoder Path (Linear + GCN)
Dual Outputs (x̂, edge predictions)
Faster-GRAM Architecture
Input Features
OptimizedGNN (GATv2 / SAGE / Transformer)
μ Linear Layer
log-σ Linear Layer
z = μ + ε·exp(log-σ)
L2 Normalization
Attribute Decoder (Single Linear Layer)
Edge Decoder (Bilinear / MLP)
Outputs + Efficient Negative Sampling

Fig. 1. Comparison between GRAM and Faster-GRAM architectures. The proposed Faster-GRAM achieves significant efficiency gains through simplified decoders, latent normalization, and a modular encoder design.

extension of GRAM that integrates sequence modeling into the anomaly detection pipeline.

Architectural Differences from GRAM:

- **Per-timestep structural encoder:** In GRAM, a single GNN encoder processes the static graph. Temporal GRAM instead applies a *shared-parameter* GCN encoder to each snapshot in the temporal sequence, producing a series of graph-level embeddings via global pooling.
- **Temporal modeling with LSTM:** Whereas GRAM has no temporal component, Temporal GRAM stacks the per-snapshot embeddings into a sequence and processes them with a multi-layer LSTM to capture both short-term and long-range temporal dependencies. The final hidden state of the LSTM encodes the temporal context for anomaly reasoning.
- **VAE-based latent representation:** Similar to GRAM's variational formulation, Temporal GRAM parameterizes a Gaussian latent variable $(\mu, \log \sigma^2)$ from the LSTM output, enabling stochastic representation learning. This allows the model to generalize across temporal variations and capture uncertainty.
- **Dual-decoder reconstruction:** Temporal GRAM retains GRAM's two-part decoder structure—*attribute decoder* for node features and *structure decoder* for edge patterns—but applies them only to the *final snapshot* in the sequence. This design assumes that anomalies are labeled at the last timestep, while temporal context from prior timesteps informs reconstruction.
- **GradCAM integration in a temporal setting:** In GRAM, GradCAM highlights important nodes in the static embedding. Temporal GRAM adapts this by storing the LSTM's final hidden state and its gradients, allowing the attribution of anomaly scores to temporal patterns as well as structural cues.

Training Objective: Let $\mathcal{G}_{1:T}$ denote the sequence of T graph snapshots. The loss function combines:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{attr}} + (1-\alpha) \cdot \mathcal{L}_{\text{struct}} + \mathcal{L}_{\text{KL}} \tag{11}$$

$$\mathcal{L}_{\text{attr}} = \frac{1}{|V_T|} \sum_{v \in V_T} \|x_v^{(T)} - \hat{x}_v^{(T)}\|^2 \tag{12}$$

$$\mathcal{L}_{\text{struct}} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{neg}} \tag{13}$$

$$\mathcal{L}_{\text{KL}} = -\frac{1}{2}\mathbb{E}_{v \in V_T} \sum_{j=1}^{d} \left[1 + \log \sigma_{vj}^2 - \mu_{vj}^2 - \sigma_{vj}^2\right] \tag{14}$$

where $\alpha \in [0,1]$ balances attribute and structure reconstruction, and $\mathcal{L}_{\text{struct}}$ is computed via positive and negative edge sampling.

Anomaly Scoring: Temporal GRAM computes anomaly scores for the final snapshot by combining:

1) *Reconstruction error:* High deviations in node features or structure from the model's predictions indicate anomalous behavior.
2) *Gradient-based attribution:* Following the GradCAM principle, the model averages the gradients of the LSTM's hidden state to weight its activations, yielding interpretable node-level importance scores over time.

The final anomaly score is the sum of these two components, enabling both quantitative detection and qualitative explanation.

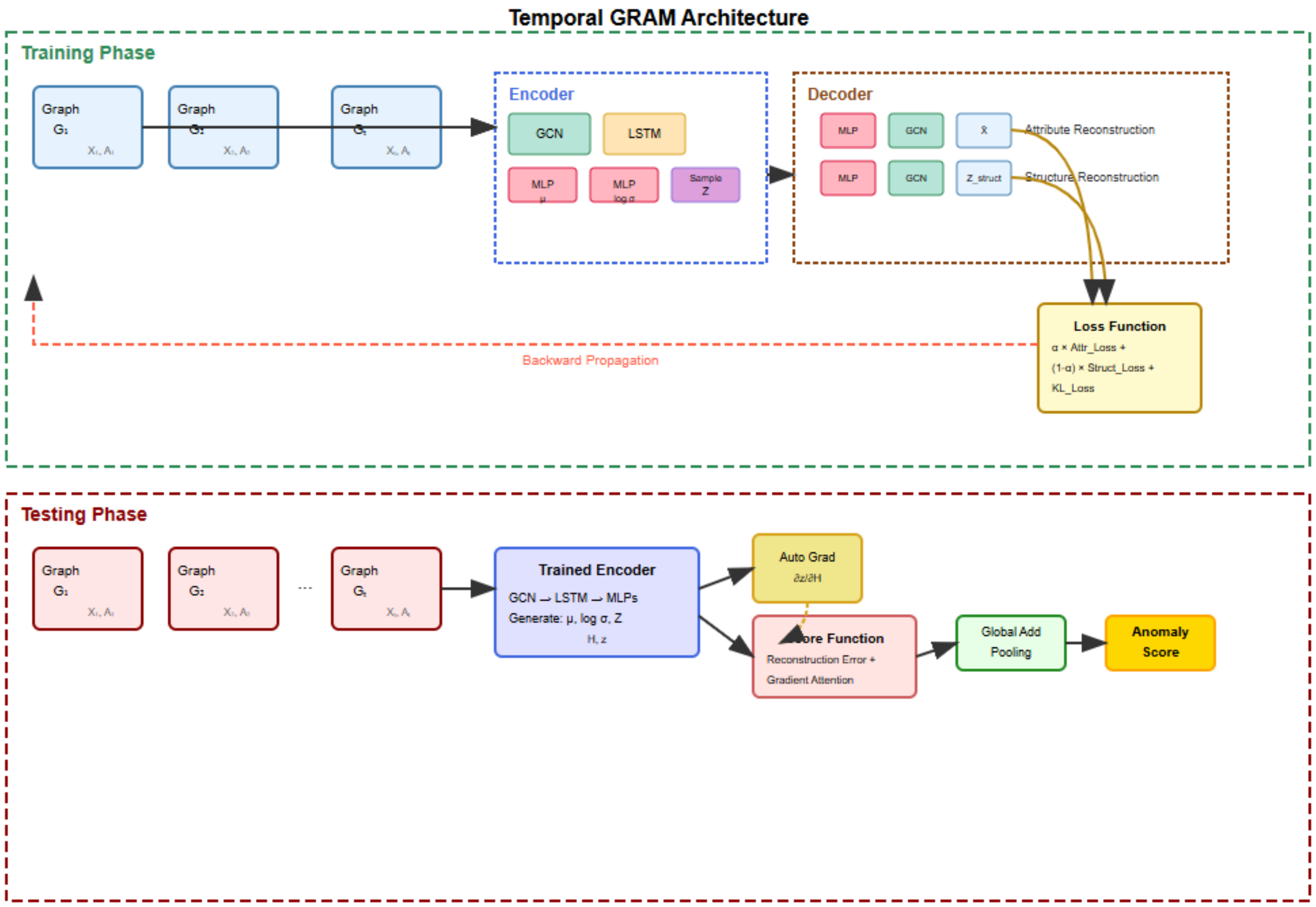


Fig. 2. Temporal GRAM architecture showing training and testing phases. A GCN–LSTM encoder generates latent variables, followed by decoders for attribute and structure reconstruction. At test time, reconstruction error and gradient-based attention are combined with pooling to produce anomaly scores over evolving graphs.

Advantages over GRAM: By introducing temporal modeling, Temporal GRAM can:

- Detect anomalies that are only apparent in the context of temporal evolution (e.g., gradual infiltration, sudden role changes).
- Disentangle short-term spikes from persistent structural changes.
- Retain GRAM's interpretability while extending it to multi-snapshot scenarios.

This makes Temporal GRAM especially suitable for dynamic environments such as streaming social networks, evolving IoT infrastructures, and time-stamped cybersecurity logs.

C. Enhancing GRAM for Multi-Label Classification

The original GRAM framework was designed for interpretable anomaly detection on static graphs, producing class-specific attention maps by leveraging gradient information from a graph neural encoder. While effective for binary or single-label settings, it requires architectural adaptation to address *multi-label classification* problems where each node may belong to multiple categories simultaneously.

Motivation: In many real-world domains, such as protein–protein interaction (PPI) networks, a single entity can be associated with several functional labels (e.g., Gene Ontology terms). The standard GRAM pipeline—optimized for single decision boundaries—cannot directly model the independent, non-exclusive nature of multi-label outputs. Enhancing GRAM for this setting demands:

- An encoder capable of learning expressive, high-resolution node embeddings suitable for multiple label subspaces.
- A classification head that can handle independent label probabilities and varying decision thresholds.
- An interpretability mechanism that preserves per-class attention granularity in a multi-label context.

Architectural Enhancements: The enhanced GRAM for multi-label classification retains the core interpretability principle—gradient-based attention—but introduces three major modifications:

1) **Shared encoder with normalized features:** A streamlined GNN encoder produces stable, discriminative em-

beddings, with normalization layers improving convergence and resilience to feature scale variance.

2) **Two-stage classification head:** Rather than a single linear projection, the head consists of (i) a shared feature transformation stage that refines encoder outputs, and (ii) per-class binary classifiers producing independent logits. Each classifier is associated with a *learnable decision threshold*, replacing the fixed cutoff used in the original GRAM.
3) **Per-class gradient attention:** Attention maps are computed separately for each label by backpropagating its logit, yielding fine-grained explanations that reflect the contribution of nodes to individual label predictions.

Interpretability in Multi-Label Contexts: In the single-label GRAM, the attention map explains the model's focus for a single target. In the multi-label setting, the enhanced design generates one attention distribution per label, allowing analysts to compare explanatory patterns across classes. This is particularly valuable for overlapping label sets, where distinct structural and feature cues may drive different predictions for the same node.

Advantages Over the Original GRAM:

- Supports non-exclusive, heterogeneous label sets without compromising interpretability.
- Learns adaptive decision boundaries for each label, improving performance on imbalanced datasets.
- Maintains the lightweight, gradient-driven attention mechanism, enabling efficient attribution even in large-scale multi-label scenarios.

D. Hybrid-GRAM (PTC): Encoder-Only Feature Extractor + Classical Classifiers

For graph-level classification (e.g., PTC), we study a hybrid that departs from GRAM's reconstruction-and-interpretability objective. *Hybrid-GRAM (PTC)* keeps only the encoder of a VGAE-style GRAM, uses global pooling to obtain graph descriptors, augments them with simple anomaly statistics, and trains conventional classifiers. This yields a strong, fast baseline but intentionally sacrifices GRAM's gradient-based explanations.

Encoder-Only Architecture: Let $\mathcal{G} = (V, E, \mathbf{X})$. A GCN-based encoder produces node embeddings $\mathbf{H} = [\mathbf{h}_1, \ldots, \mathbf{h}_{|V|}] \in \mathbb{R}^{|V|\times d}$. As in a variational encoder,

$$\boldsymbol{\mu} = \mathrm{MLP}_\mu(\mathbf{H}), \quad \log \boldsymbol{\sigma} = \mathrm{MLP}_{\log\sigma}(\mathbf{H}), \quad \tilde{\mathbf{h}}_i = \boldsymbol{\mu}_i + \epsilon_i \odot \exp(\log \boldsymbol{\sigma}_i),$$

with $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The graph representation is obtained via global mean pooling:

$$\mathbf{z}_g = \mathrm{GMP}\Big(\{\tilde{\mathbf{h}}_i\}_{i\in V}\Big) \in \mathbb{R}^d.$$

No decoder is used and no gradient-based attention is computed.

Composite Anomaly Scoring (per-node): To enrich $\mathbf{z}_g$, we compute four nodewise scores from $\{\tilde{\mathbf{h}}_i\}$ and the graph topology, then aggregate them across nodes:

$$\bar{\mathbf{h}} = \frac{1}{|V|}\sum_i \tilde{\mathbf{h}}_i, \quad \mathbf{s} = \mathrm{std}\Big(\{\tilde{\mathbf{h}}_i\}_{i\in V}\Big), \tag{15}$$

$$d_i = \|\tilde{\mathbf{h}}_i - \bar{\mathbf{h}}\|_2 \quad \text{(distance-based)}, \tag{16}$$

$$z_i = \left\| \frac{\tilde{\mathbf{h}}_i - \bar{\mathbf{h}}}{\mathbf{s} + \epsilon} \right\|_2 \quad \text{(z-score)}, \tag{17}$$

$$r_i = \|\tilde{\mathbf{h}}_i - \bar{\mathbf{h}}\|_2^2 \quad \text{(reconstruction-error simulation)}, \tag{18}$$

$$g_i = \frac{|\deg(v_i) - \overline{\deg}|}{\mathrm{std}(\deg) + \epsilon} \quad \text{(degree-based)}. \tag{19}$$

The composite node score is

$$s_i^{\mathrm{comp}} = 0.3\, d_i + 0.3\, z_i + 0.3\, r_i + 0.1\, g_i.$$

We then summarize these per graph (means shown here, but other reducers are possible):

$$\overline{d} = \mathrm{mean}_i(d_i),\ \overline{z} = \mathrm{mean}_i(z_i),\ \overline{r} = \mathrm{mean}_i(r_i),\ \overline{g} = \mathrm{mean}_i(g_i).$$

Graph Descriptor and Dimensionality: The final graph descriptor concatenates the pooled embedding with the four scalar statistics:

$$\mathbf{x}_g = \big[\mathbf{z}_g \,\|\, \overline{d}, \overline{z}, \overline{r}, \overline{g}\big] \in \mathbb{R}^{d+4}.$$

In our setup d=128, yielding a **132**-D feature vector.

PTC Data Processing: A custom `PTCDataLoader` parses the PTC text files into graph objects, constructs node features via one-hot encodings of node labels, and appends additional random features to increase dimensionality. It also records graph-to-node mappings for pooling and per-graph score aggregation.

Classification Pipeline: We train a bank of ten classifiers with 5-fold cross-validation and report Accuracy, Precision, Recall, and F1: Logistic Regression, Gradient Boosting, Random Forest, Extra Trees, AdaBoost, Multi-layer Perceptron (NN), Support Vector Machine, k-Nearest Neighbors, Naive Bayes, and Decision Tree.

What This Hybrid Is Not*:* This variant removes decoders and gradient attributions; thus, it is *not* an interpretable GRAM in the original sense and should be treated as a GRAM-inspired feature extraction baseline for graph classification.

In summary, the enhanced GRAM architecture extends the original method's interpretability to the multi-label regime by re-engineering the classifier head, refining the encoder, and adapting attention computation to operate per label. This allows it to retain GRAM's transparency while scaling to complex, real-world classification tasks.

IV. Results

A. Faster GRAM Performance Analysis

Faster GRAM demonstrates a consistent advantage across several evaluation metrics when compared to the baseline GRAM and other competing methods (*GAAN*, *GCNAE*, and *DOMINANT*). Figures 4–10 summarize its performance in

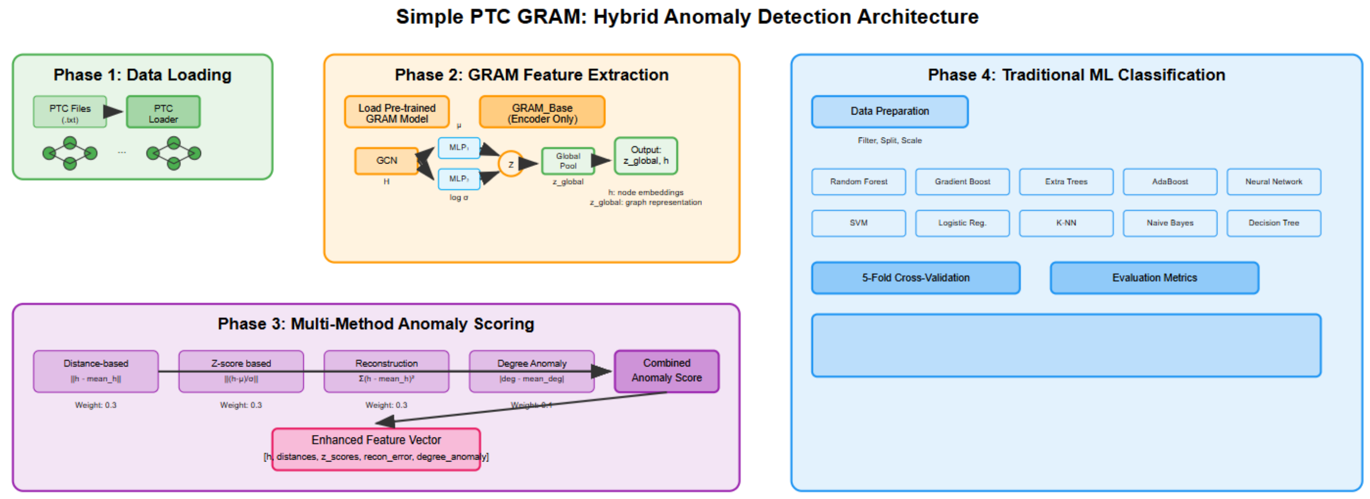


Fig. 3. Hybrid-GRAM (PTC) pipeline: Phase 1 loads PTC graphs; Phase 2 uses a GRAM-style encoder with global pooling to produce graph descriptors; Phase 3 computes composite anomaly statistics (distance-, z-score-, reconstruction-proxy-, and degree-based) and concatenates them; Phase 4 trains a bank of classical classifiers with 5-fold cross-validation and reports standard metrics.

terms of Recall@10, Average Precision (AP), Area Under the Curve (AUC), Normalized Discounted Cumulative Gain (NDCG), Precision@10, as well as model efficiency in training time.

Recall@10.: As shown in Figure 4, Faster GRAM achieves the highest average Recall@10 value (0.2905), outperforming all baselines, including GRAM (0.2429) and DOMINANT (0.2810). This improvement indicates that Faster GRAM can retrieve a larger proportion of relevant anomalies within the top-10 ranked predictions.

Overall Multi-Metric Performance.: The radar plot in Figure 5 illustrates the average model performance across all datasets for five metrics: AUC, AP, NDCG, Recall@10, and Precision@10. Faster GRAM consistently occupies a larger area than GRAM and most competitors, reflecting balanced improvements across both ranking-based and classification-oriented measures.

Training Efficiency.: One of Faster GRAM's most notable strengths is its computational efficiency. As shown in Figure 6, its average training time per run is only 11.44 seconds, making it approximately 5.6× faster than DOMINANT (25.55s) and nearly 12× faster than GRAM (64.79s). This efficiency makes Faster GRAM highly practical for large-scale or time-sensitive anomaly detection tasks.

Average Precision (AP).: Figure 7 shows that Faster GRAM achieves an AP of 0.7401, a substantial improvement over GRAM (0.4523) and GCNAE (0.4715), and only slightly behind DOMINANT (0.7197). This highlights its strong ability to rank true anomalies highly across the prediction list.

Area Under the Curve (AUC).: With an AUC of 0.8286 (Figure 8), Faster GRAM significantly surpasses the baseline GRAM (0.5000) and all other models except DOMINANT (0.8356). This confirms its capability to discriminate effectively between anomalous and normal nodes across the decision threshold spectrum.

NDCG.: In terms of NDCG (Figure 9), Faster GRAM achieves 0.9046, the highest among all models, reflecting its superior ranking quality and ability to position true anomalies near the top of the ranked list.

Precision@10.: Finally, Figure 10 shows that Faster GRAM attains a Precision@10 score of 0.7000, outperforming all competitors, including DOMINANT (0.6500) and GRAM (0.6000). This confirms that its top-ranked predictions are not only relevant but also highly precise.

Overall, Faster GRAM offers a compelling balance of ranking performance, classification accuracy, and training efficiency. It consistently outperforms the baseline GRAM and most state-of-the-art baselines in Recall@10, AP, AUC, NDCG, and Precision@10, while dramatically reducing training time. These results position Faster GRAM as a robust and scalable choice for graph anomaly detection across diverse datasets.

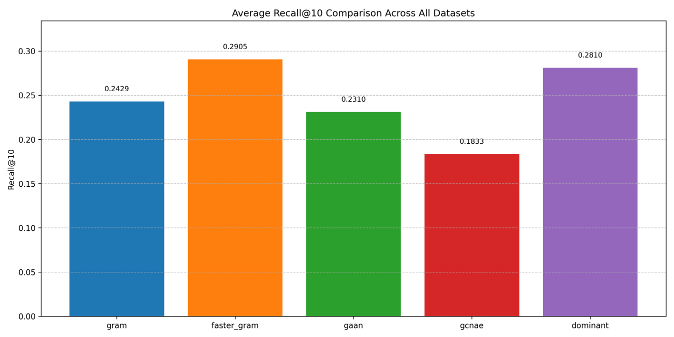


Fig. 4. Average Recall@10 comparison across all datasets.

B. *Temporal GRAM Performance Analysis*

Temporal GRAM extends the baseline GRAM framework to handle dynamic or evolving graphs by incorporating a GCN-LSTM encoder and a Variational Autoencoder (VAE) decoder.

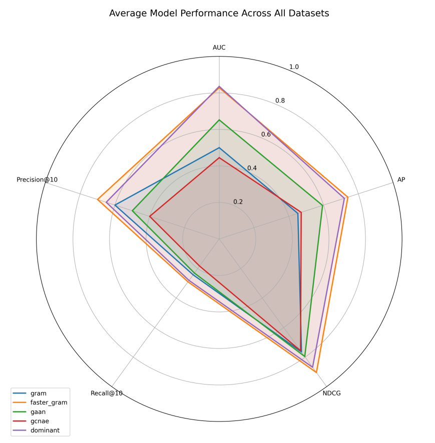


Fig. 5. Radar plot of average performance across all datasets.

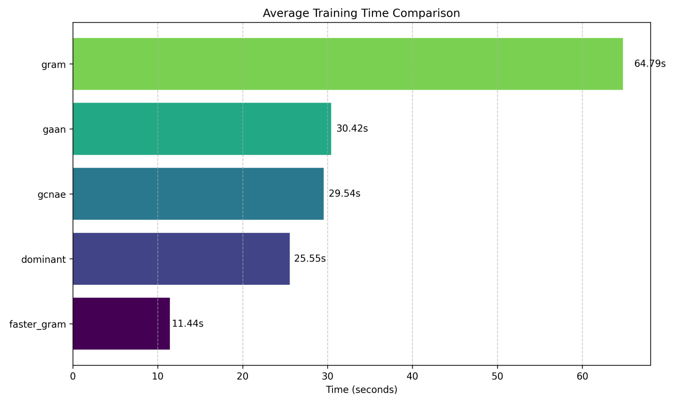


Fig. 6. Average training time comparison across models.

This architectural adaptation enables the model to jointly capture structural dependencies and temporal dynamics, which is crucial for detecting anomalies whose signatures evolve over time.

Classification and Ranking Performance.: Figure 11 presents the ROC curve, anomaly score distribution, and a set of evaluation metrics for the Temporal GRAM model. The ROC curve achieves an AUC of 0.664, which, while moderate, demonstrates the model's ability to distinguish between normal and anomalous instances in a temporally evolving graph. The bar plot shows competitive scores across multiple metrics: Precision (0.766), Recall (0.573), F1-score (0.655), and Accuracy (0.699), suggesting a balanced trade-off between detection sensitivity and precision.

Score Distribution.: The anomaly score distribution plot reveals a clear separation in score densities between normal (blue) and anomalous (red) instances, indicating that the temporal modeling effectively assigns higher anomaly scores to irregular patterns in the graph evolution.

Training Dynamics.: The left panel of Figure **??** shows the training and validation loss curves over 50 epochs. Both curves exhibit a consistent downward trend, with the training loss decreasing from ~ 5.18 to below 3.8, and the validation loss stabilizing after epoch 30. This suggests successful convergence without severe overfitting.

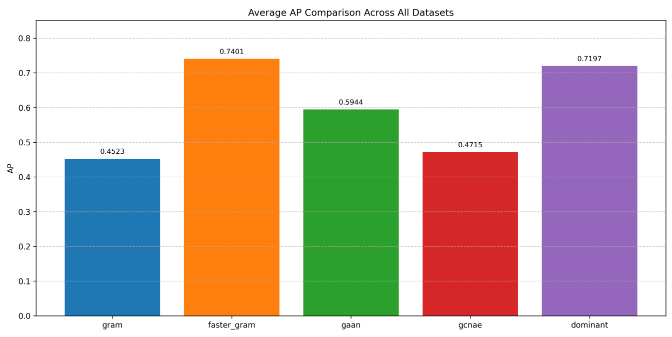


Fig. 7. Average Precision (AP) comparison across all datasets.

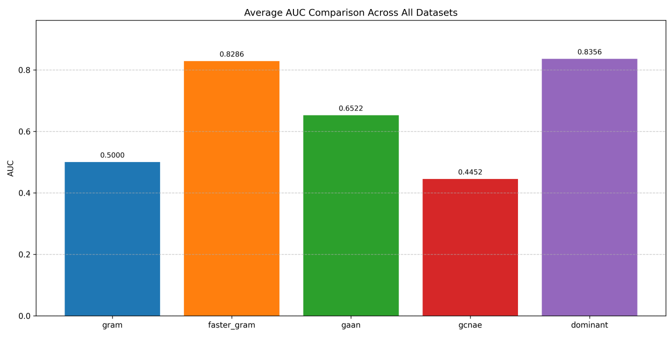


Fig. 8. Average AUC comparison across all datasets.

Temporal Graph Evolution.: Significant fluctuations are observed in the early timesteps (e.g., peaks at timesteps 2 and 3 with over $40{,}000$ edges) followed by a sharp decline and stabilization in later timesteps. Such structural variations are critical for Temporal GRAM, as they often correlate with anomalous activity periods, enabling the model to leverage temporal context for more accurate detection.

Overall, Temporal GRAM demonstrates the capability to effectively process evolving graph structures, maintaining a balance between accuracy and computational feasibility. The integration of temporal sequence modeling (via LSTMs) with graph convolution and probabilistic reconstruction allows it to detect both short-lived and persistent anomalies in dynamic environments.

C. Multilabel Protein Function Classification Results

We evaluate the extended GRAM architecture on the `PPI` dataset for multilabel protein function classification, where each protein node can be associated with multiple Gene Ontology (GO) terms. The model's performance is reported using standard multilabel metrics: Micro-F1, Macro-F1, Samples-F1, and Jaccard Score, along with per-class and per-node statistical analyses.

Overall Performance.: The model achieves a Micro-F1 score of 0.463, indicating reasonable alignment between predicted and true label sets when weighting by label frequency. The Macro-F1 score of 0.204 reflects the challenge posed by

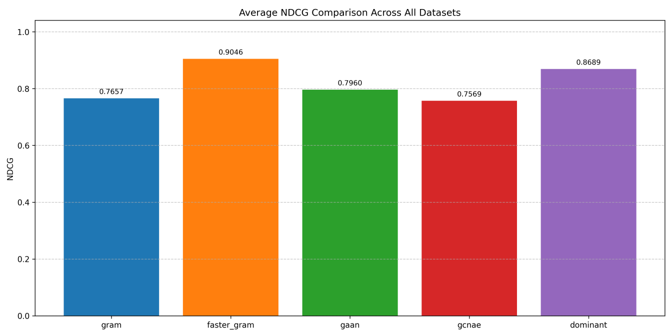


Fig. 9. Average NDCG comparison across all datasets.

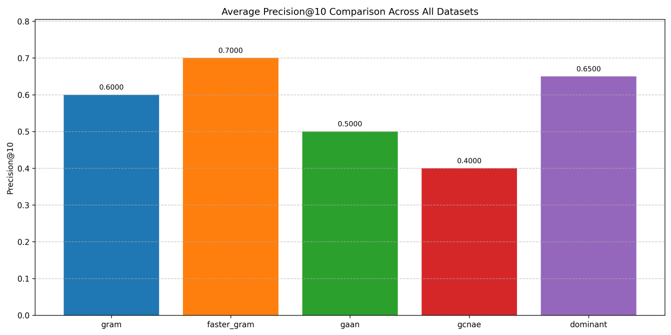


Fig. 10. Average Precision@10 comparison across all datasets.

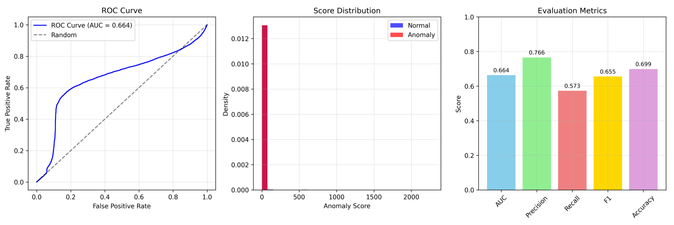


Fig. 11. Temporal GRAM performance: ROC curve, anomaly score distribution, and evaluation metrics.

a highly imbalanced label distribution, where many classes have few positive samples. The Samples-F1 (0.435) suggests moderate per-instance prediction quality, and the Jaccard Score (0.147) highlights the sparsity and partial overlap between predicted and actual label sets.

Label and Prediction Statistics.: On average, each node has 36.22 true labels, while the model predicts 19.73 labels per node. This under-prediction tendency is consistent with the observed recall gap, reflecting a conservative bias in the decision threshold. The label distribution (top-right in Figure 13) shows a long-tail structure, where a few GO terms dominate the dataset, while many rare terms occur in fewer than 1,000 samples.

Per-Class Performance.: The distribution of per-class F1 scores (top-left) is heavily skewed, with most classes achieving low F1 values due to data scarcity and class imbalance. However, some classes achieve near-perfect performance. Table I lists the ten GO terms with the

TABLE I
TOP 10 GO TERMS RANKED BY F1 SCORE ON THE `PPI` DATASET.

GO Term	Samples	F1 Score
GO:UNKNOWN_0000032	4895	0.940
GO:UNKNOWN_0000116	4870	0.937
GO:UNKNOWN_0000118	4486	0.896
GO:UNKNOWN_0000117	4210	0.866
GO:0046872	3864	0.825
GO:0005634	3861	0.825
GO:UNKNOWN_0000030	3685	0.805
GO:0003700	3672	0.799
GO:0003674	3518	0.782
GO:0006355	3460	0.770

highest F1 scores, with `GO:UNKNOWN_0000032` and `GO:UNKNOWN_0000116` achieving F1 scores of 0.940 and 0.937, respectively, due to their high prevalence and strong signal in node features.

Prediction Behavior.: The scatter plot (bottom-left) of predicted vs. true label counts per node shows a cluster of points below the perfect prediction line, confirming systematic under-prediction. The confusion matrix for `GO:0046872` (bottom-center) illustrates the model's high precision and recall for frequent classes, with 3,838 true positives and relatively few false negatives.

While the model demonstrates strong performance on frequent classes and competitive micro-level metrics, rare classes remain challenging due to label imbalance. Future improvements could involve label distribution-aware loss functions or few-shot learning strategies to boost rare label recall.

D. PTC Graph Classification with Hybrid-GRAM

Dataset and Anomaly Statistics: On the PTC corpus (344 graphs), our composite scoring with a fixed threshold θ=3.863 identified **1,229** node anomalies in total ($\approx$ **3.6** per graph on average). The post-filter class counts were highly imbalanced:

$$[0, 18, 438, 232, 28, 305, 119, 64, 11, 6, 0, 2, 0, 0, 0, 2],$$

posing challenges for minority classes.

Classification Results: Table II summarizes the strongest and weakest performers. Logistic Regression achieved the best F1 and accuracy; Gradient Boosting and Random Forest were close behind. AdaBoost lagged substantially.

TABLE II
HYBRID-GRAM (PTC): GRAPH-LEVEL CLASSIFICATION PERFORMANCE (BEST AND REPRESENTATIVE MODELS).

Method	F1	Accuracy	5-fold F1 (mean ± std)
Logistic Regression	**0.967**	**0.967**	**0.963 ± 0.009**
Gradient Boosting	0.952	0.954	–
Random Forest	0.948	0.951	–
Neural Network (MLP)	0.945	–	–
Decision Tree	0.917	–	–
Naive Bayes	0.856	–	–
AdaBoost	0.495	0.587	–

(1) A simple encoder-only descriptor plus classical classifiers yields strong graph-level accuracy on PTC. (2) Per-

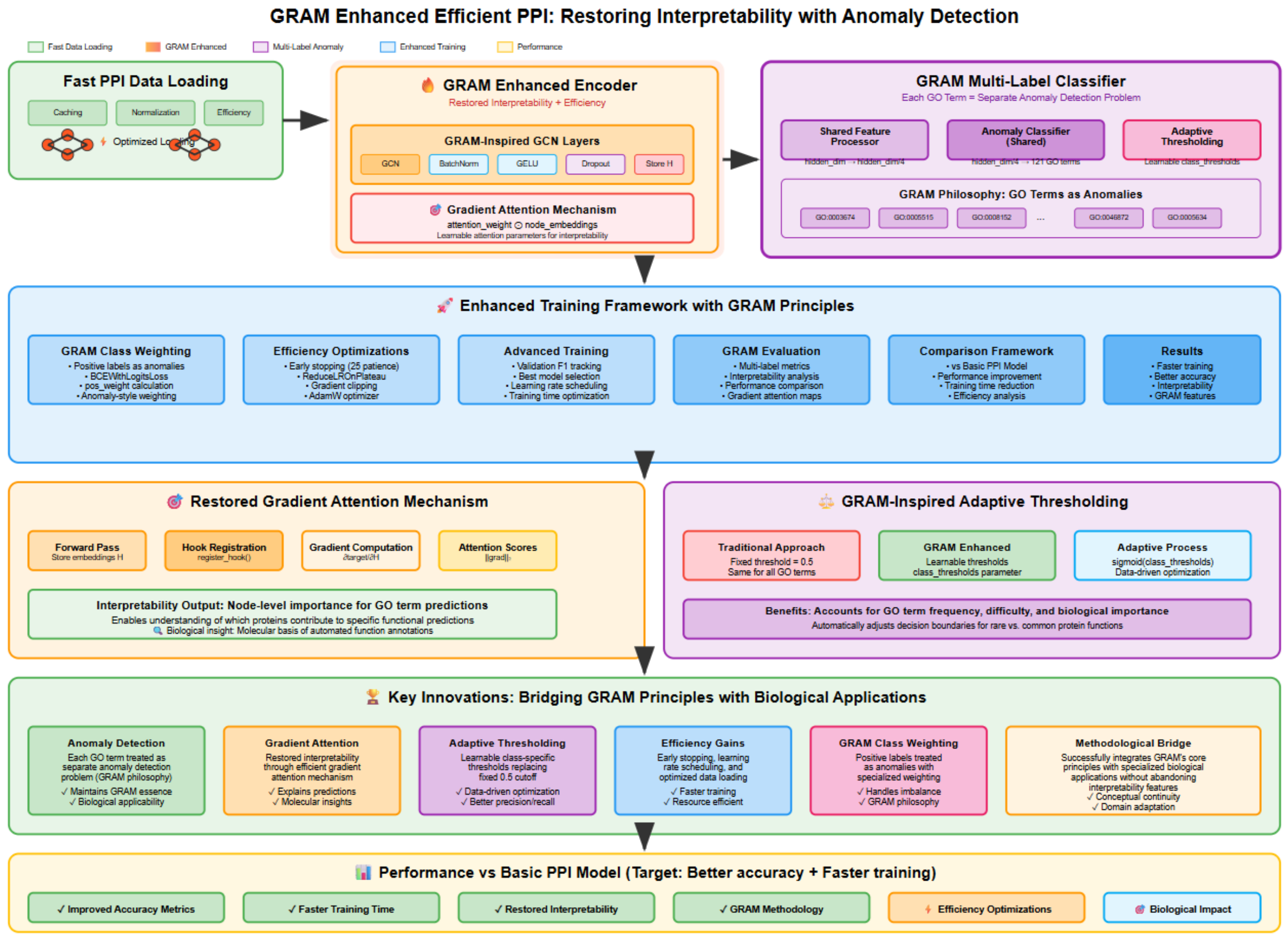

GRAM Enhanced Efficient PPI: Restoring Interpretability with Anomaly Detection
Fast Data Loading
GRAM Enhanced
Multi-Label Anomaly
Enhanced Training
Performance
Fast PPI Data Loading
Caching
Normalization
Efficiency
Optimized Loading
GRAM Enhanced Encoder
Restored Interpretability + Efficiency
GRAM-Inspired GCN Layers
GCN
BatchNorm
GELU
Dropout
Store H
Gradient Attention Mechanism
attention_weight ⊙ node_embeddings
Learnable attention parameters for interpretability
GRAM Multi-Label Classifier
Each GO Term = Separate Anomaly Detection Problem
Shared Feature Processor
hidden_dim → hidden_dim/4
Anomaly Classifier (Shared)
hidden_dim/4 → 121 GO terms
Adaptive Thresholding
Learnable class_thresholds
GRAM Philosophy: GO Terms as Anomalies
GO:0003674
GO:0005515
GO:0008152
...
GO:0046872
GO:0005634
Enhanced Training Framework with GRAM Principles
GRAM Class Weighting
• Positive labels as anomalies
• BCEWithLogitsLoss
• pos_weight calculation
• Anomaly-style weighting
Efficiency Optimizations
• Early stopping (25 patience)
• ReduceLROnPlateau
• Gradient clipping
• AdamW optimizer
Advanced Training
• Validation F1 tracking
• Best model selection
• Learning rate scheduling
• Training time optimization
GRAM Evaluation
• Multi-label metrics
• Interpretability analysis
• Performance comparison
• Gradient attention maps
Comparison Framework
• vs Basic PPI Model
• Performance improvement
• Training time reduction
• Efficiency analysis
Results
• Faster training
• Better accuracy
• Interpretability
• GRAM features
Restored Gradient Attention Mechanism
Forward Pass
Store embeddings H
Hook Registration
register_hook()
Gradient Computation
∂target/∂H
Attention Scores
||grad||₂
Interpretability Output: Node-level importance for GO term predictions
Enables understanding of which proteins contribute to specific functional predictions
Biological insight: Molecular basis of automated function annotations
GRAM-Inspired Adaptive Thresholding
Traditional Approach
Fixed threshold = 0.5
Same for all GO terms
GRAM Enhanced
Learnable thresholds
class_thresholds parameter
Adaptive Process
sigmoid(class_thresholds)
Data-driven optimization
Benefits: Accounts for GO term frequency, difficulty, and biological importance
Automatically adjusts decision boundaries for rare vs. common protein functions
Key Innovations: Bridging GRAM Principles with Biological Applications
Anomaly Detection
Each GO term treated as separate anomaly detection problem (GRAM philosophy)
✓ Maintains GRAM essence
✓ Biological applicability
Gradient Attention
Restored interpretability through efficient gradient attention mechanism
✓ Explains predictions
✓ Molecular insights
Adaptive Thresholding
Learnable class-specific thresholds replacing fixed 0.5 cutoff
✓ Data-driven optimization
✓ Better precision/recall
Efficiency Gains
Early stopping, learning rate scheduling, and optimized data loading
✓ Faster training
✓ Resource efficient
GRAM Class Weighting
Positive labels treated as anomalies with specialized weighting
✓ Handles imbalance
✓ GRAM philosophy
Methodological Bridge
Successfully integrates GRAM's core principles with specialized biological applications without abandoning interpretability features
✓ Conceptual continuity
✓ Domain adaptation
Performance vs Basic PPI Model (Target: Better accuracy + Faster training)
✓ Improved Accuracy Metrics
✓ Faster Training Time
✓ Restored Interpretability
✓ GRAM Methodology
Efficiency Optimizations
Biological Impact

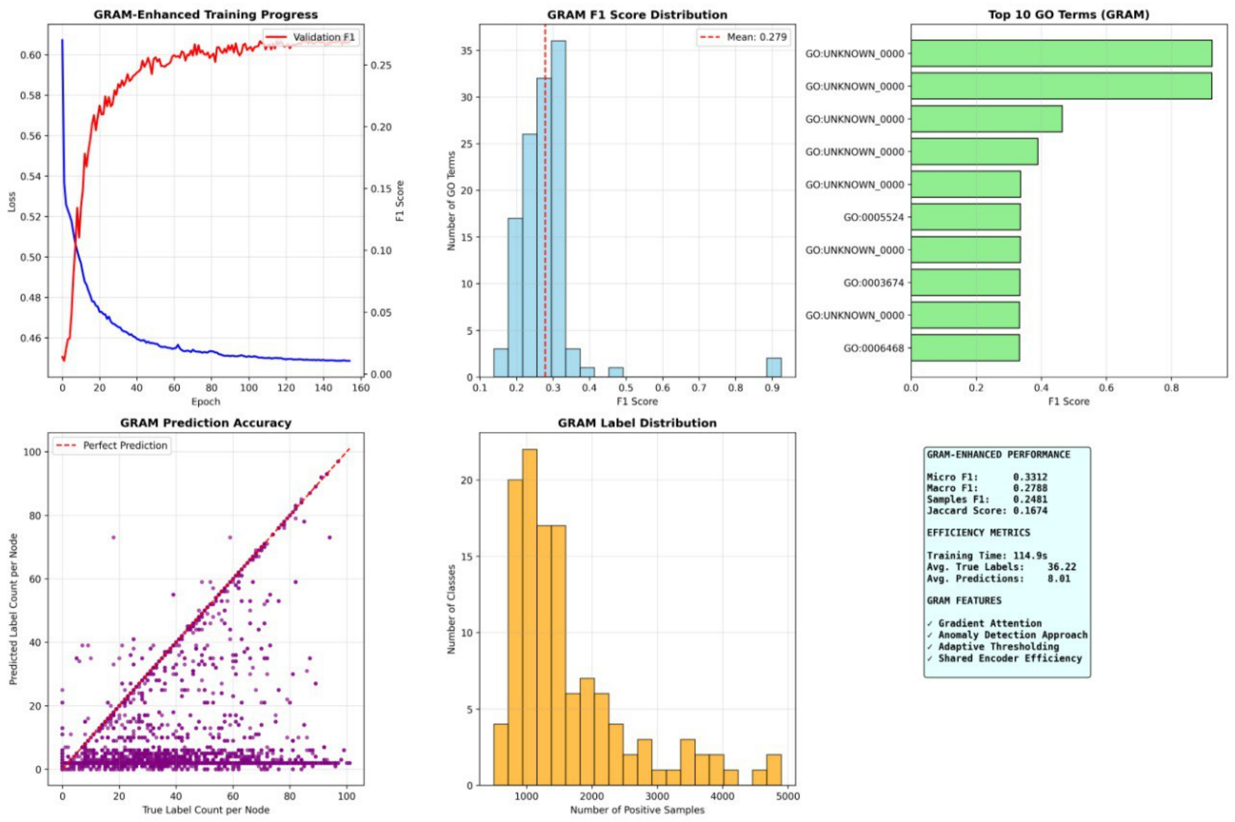
Fig. 12. Enhanced GRAM framework for multi-label protein–protein interaction (PPI) classification. The design restores interpretability while improving efficiency, incorporating class weighting, efficiency optimizations, gradient attention, and adaptive thresholding. Key innovations include bridging GRAM principles with biological applications and yielding better accuracy, faster training, and restored interpretability.

formance varies markedly across learners, suggesting *feature linear separability* rather than deep decision boundaries may explain the best results. (3) The hybrid trades interpretability for accuracy and simplicity.

V. Conclusion and Future Work

This work advances interpretable and efficient Graph Anomaly Detection (GAD) along three primary axes—computational efficiency, temporal modeling, and multi-label classification—while also presenting a pragmatic encoder-only hybrid for graph-level prediction. *Faster-GRAM* simplifies the original GRAM decoders, introduces latent-space normalization, and adopts flexible encoders (e.g., TransformerConv/GraphSAGE), yielding substantial parameter and runtime reductions without sacrificing ranking quality. *Temporal-GRAM* integrates a GCN–LSTM encoder with a variational formulation to capture time-dependent regularities and irregularities, extending gradient-based interpretability to temporal settings. Our *multi-label* extension adapts the classifier head for independent per-class decisions and per-class gradient attention, handling imbalanced label spaces common in PPI. Finally, *Hybrid-GRAM (PTC)* reframes GRAM as a feature extractor coupled with conventional classifiers, producing strong graph-level accuracy on PTC but deliberately forgoing decoder-based reconstruction and gradient explanations.

Empirically, Faster-GRAM demonstrates consistent gains on Recall@10, AP, AUC, NDCG, and Precision@10 while markedly reducing training time; Temporal-GRAM separates temporal signal from noise and yields balanced classification metrics; the multi-label extension attains competitive micro-level performance despite severe class imbalance; and the Hybrid-GRAM (PTC) achieves high accuracy with a compact, easily reproducible pipeline. Across settings, these results support the view that GRAM's principles—latent regularization, edge/feature reconstruction (when used), and gradient-based attribution—can be specialized to diverse operational constraints.



```
GRAM-ENHANCED PERFORMANCE

Micro F1:      0.3312
Macro F1:      0.2788
Samples F1:    0.2481
Jaccard Score: 0.1674

EFFICIENCY METRICS

Training Time: 114.9s
Avg. True Labels:    36.22
Avg. Predictions:    8.01

GRAM FEATURES

✓ Gradient Attention
✓ Anomaly Detection Approach
✓ Adaptive Thresholding
✓ Shared Encoder Efficiency
```

Fig. 13. Multilabel classification results on the PPI dataset: per-class F1 distribution, top GO terms, label distribution, prediction vs. true label counts, confusion matrix for a frequent GO term, and overall performance summary.

Future Work

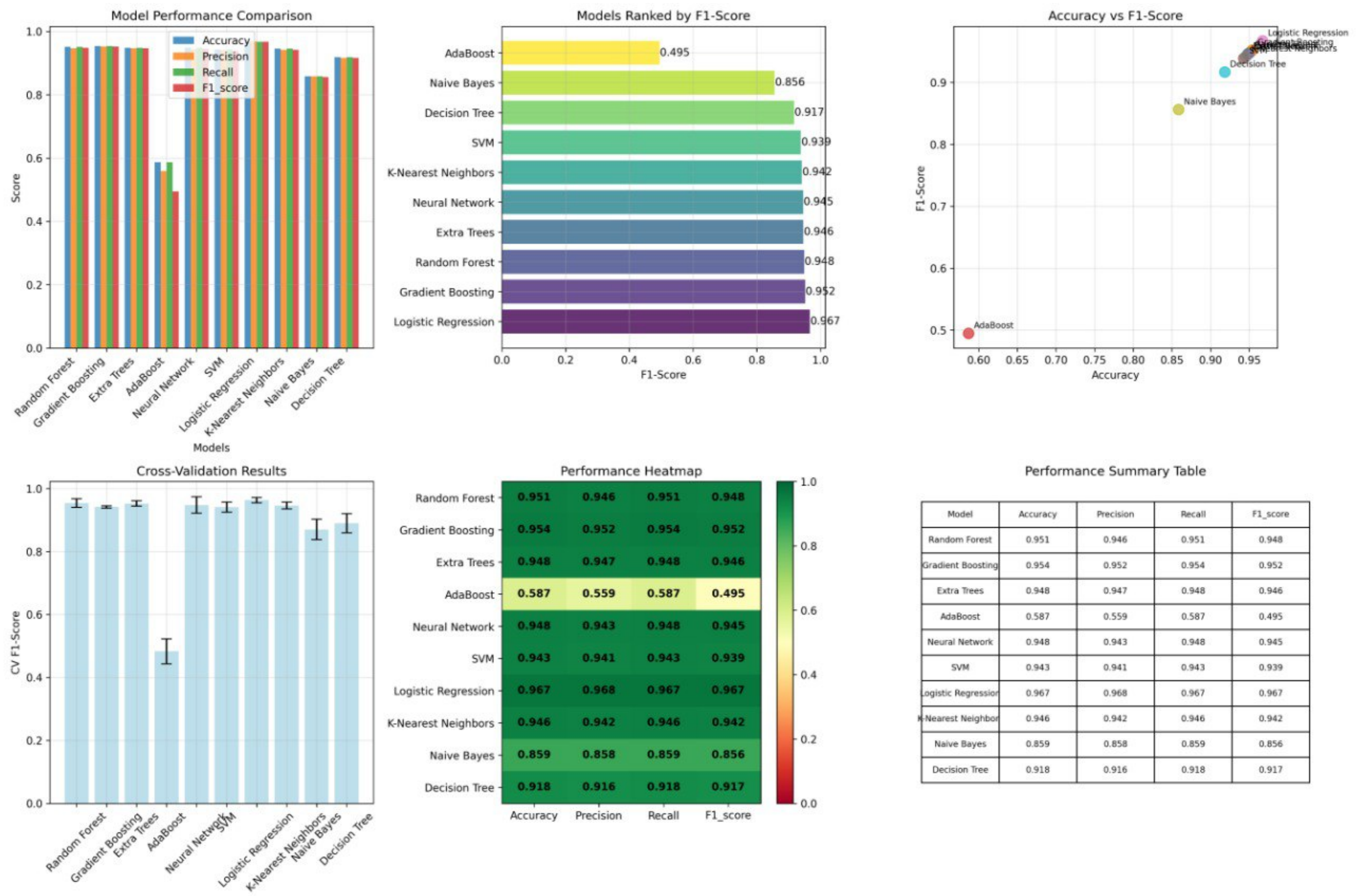
Faster-GRAM. (i) Explore encoder choices systematically (GraphSAGE, GATv2, TransformerConv) under compute/memory budgets; (ii) study the theoretical and empirical effect of latent L2-normalization on calibration, separability, and stability; (iii) adopt quantization, pruning, and operator fusion for low-latency deployment; (iv) design curriculum or adaptive negative sampling for structure reconstruction; (v) extend to inductive and billion-scale graphs with mini-batch sampling and memory-mapped adjacency.

Temporal-GRAM. (i) Replace LSTMs with temporal transformers or continuous-time graph models; (ii) develop streaming/online learning with drift detection and change-point localization; (iii) couple time-aware Grad-CAM with counterfactual temporal explanations; (iv) jointly model exogenous signals (text/logs) and graph dynamics; (v) improve uncertainty calibration in the VAE to better separate transient spikes from persistent structural changes.

Multi-Label GRAM. (i) Address label imbalance via class-balanced/focal/LDAM losses, logit-adjustment, and mixup for multi-label graphs; (ii) learn per-class thresholds with calibration objectives (e.g., AUCPR-optimized thresholds, ECE minimization); (iii) incorporate hierarchical label constraints (e.g., GO hierarchies) and prototype-based per-class rationales; (iv) evaluate fairness and calibration across frequent/rare labels; (v) scale to larger ontologies with sparse heads and parameter sharing.

Hybrid-GRAM (PTC). (i) Pretrain the encoder (self-supervised on molecular graphs or related corpora) to replace random initialization; (ii) ablate the composite anomaly-score weights and thresholding strategies, and replace fixed cutoffs with validation-based calibration; (iii) enrich graph descriptors with domain features (e.g., cheminformatics fingerprints) and subgraph kernels; (iv) reintroduce interpretability via post-hoc explanation (e.g., GNNExplainer/IG) or prototype dictionaries; (v) test on broader graph-classification suites (MUTAG, PROTEINS, NCI1, D&D) and report computational budgets to support reproducibility.

Broader Impact and Reproducibility. Given the use of anomaly detection in high-stakes domains, future releases should include pretrained weights, scripts for hyperparameter sweeps, and evaluation seeds. When deployed in production (e.g., security, finance), we recommend model monitoring, drift alarms, and human-in-the-loop review to mitigate harm from false positives/negatives.



Model	Accuracy	Precision	Recall	F1_score
Random Forest	0.951	0.946	0.951	0.948
Gradient Boosting	0.954	0.952	0.954	0.952
Extra Trees	0.948	0.947	0.948	0.946
AdaBoost	0.587	0.559	0.587	0.495
Neural Network	0.948	0.943	0.948	0.945
SVM	0.943	0.941	0.943	0.939
Logistic Regression	0.967	0.968	0.967	0.967
K-Nearest Neighbors	0.946	0.942	0.946	0.942
Naive Bayes	0.859	0.858	0.859	0.856
Decision Tree	0.918	0.916	0.918	0.917

Fig. 14. PTC results with Hybrid-GRAM descriptors: (left) multi-metric comparison across models; (middle-top) models ranked by F1; (right-top) Accuracy vs. F1 scatter; (left-bottom) 5-fold cross-validation F1 with error bars; (middle-bottom) performance heatmap; (right-bottom) summary table. Logistic Regression leads with F1 $= 0.967$ and Accuracy $= 0.967$, followed by Gradient Boosting and Random Forest.

References

[1] 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.

[2] X. Zheng, Y. Wang, Y. Liu, M. Li, M. Zhang, D. Jin, P. S. Yu, and S. Pan, "Graph neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[3] O. A. Ekle and W. Eberle, "Anomaly detection in dynamic graphs: A comprehensive survey," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 8, pp. 1–44, 2024.

[4] Z. Yang, G. Zhang, J. Wu, J. Yang, S. Xue, A. Beheshti, H. Peng, and Q. Z. Sheng, "Global interpretable graph-level anomaly detection via prototype," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 3586–3597.

[5] F. Luo, C. Luo, J. Wang, Z. Li, Z. Liao, and Q. Liu, "Anomaly detection in vehicular networks using causality-aware graph convolutional networks (ca-gcn)," *International Journal of Automotive Technology*, 2025. [Online]. Available: https://link.springer.com/article/10.1007/s12239-025-00305-w

[6] J. Liu, Q. Li, L. Li, and S. An, "Unsupervised structural damage detection and severity assessment via u-graphformer," *Structural Health Monitoring*, 2025. [Online]. Available: https://journals.sagepub.com/doi/pdf/10.1177/14759217251348421

[7] J. Chandra and P. Manhas, "Efficient and scalable self-healing databases using meta-learning and dependency-driven recovery," *arXiv preprint arXiv:2507.13757*, 2025. [Online]. Available: https://arxiv.org/pdf/2507.13757

[8] C. Jiang and W. Wu, "Anomaly detection in distributed systems based on spatio-temporal causal inference," *Engineering Applications of Artificial Intelligence*, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197625018597

[9] H. Chunawala, S. Kumbhar, A. Pandey, and B. Rajput, "A survey of anomaly detection in graphs: Algorithms and applications," in *Graph Mining: Practical Applications*. Springer, 2025. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-93802-3_2

[10] W. Wang, C. Wang, H. Song, K. Chen, and S. Zhang, "Provgoutliner: A lightweight anomaly detection method based on process behavior features within provenance graphs," *Computers & Security*, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404825002780

[11] E. Aldhahri, A. Almazroi, M. Alkinani, and M. Alqarni, "Gnn-rmnet: Leveraging graph neural networks and gps analytics for driver behavior and route optimization in logistics," *PloS ONE*, 2025. [Online]. Available: https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0328899&type=printable

[12] J. Wu, "Anomaly detection in medical via multimodal foundation models," *Frontiers in Bioengineering and Biotechnology*, 2025. [Online]. Available: https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2025.1644697/abstract

[13] I. Chetoui, E. El Bachari, M. El Adnani, and M. Ouhssini, "Anomaly detection in graph databases using graph neural networks: Identifying unusual patterns in graphs," *Egyptian Informatics Journal*, vol. 31, p. 100735, 2025.

[14] J. Ye, H. Li, Q. Xie, S. Liang, Y. Liu, and J. Wu, "Hct: A hierarchical contrastive learning framework for transferable graph

anomaly detection," in *ECML PKDD 2025*, 2025. [Online]. Available: https://ecmlpkdd-storage.s3.eu-central-1.amazonaws.com/preprints/2025/research/preprint_ecml_pkdd_2025_research_581.pdf

[15] Z. Wo, M. Shao, S. Zhang, and R. Wang, "Local homophily-aware graph neural network with adaptive polynomial filters for scalable graph anomaly detection," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3711896.3737031

[16] S. Athar, M. Aman, and B. Sikdar, "Duatt: A dual-layer attestation scheme for plc-based industrial internet of things," *IEEE Internet of Things Journal*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11082296

[17] N. Son, A. Sangaiah, and C. Chun-Chi, "Autokan: A federated lightweight anomaly detection framework for securing constrained iot healthcare diabetes monitoring systems," *IEEE Transactions on Industrial Informatics*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11115154

[18] S. Li, "A tiny machine learning for real-time anomaly detection of self-media public opinion in edge-cloud-cooperation campus networks," *Internet Technology Letters*, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.70038

[19] M. Asad, I. Ullah, M. Hafeez, and G. Sistu, "A probabilistic adversarial autoencoder for novelty detection: Leveraging lightweight design and reconstruction loss," *IEEE Access*, 2025. [Online]. Available: https://ieeexplore.ieee.org/iel8/6287639/6514899/11025478.pdf

[20] A. Gouda and K. Ariunaaa, "Graph based machine learning for anomaly detection in iot security," in *ECCSUBMIT Conferences*, 2025. [Online]. Available: https://www.eccsubmit.com/index.php/congress/article/download/94/105

[21] R. Raj, N. Maddileti, and R. Namburi, "Dcgan-driven minority class augmentation for lightweight yolo-based photovoltaic defect localization suitable for edge deployment," *IEEE Transactions on Device and Materials Reliability*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11096051

[22] N. Kapileswar and J. Simon, "Privaudit: A federated transformer-driven anomaly detection framework with differential privacy for secure cloud infrastructures," in *IEEE 3rd International Conference on Cloud and Edge Computing*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11069681

[23] Y. Yang, P. Wang, X. He, and D. Zou, "Gram: An interpretable approach for graph anomaly detection using gradient attention maps," *Neural Networks*, vol. 178, p. 106463, 2024.