Edge-Level Graph Neural Network Architectures for Network Intrusion Detection: Advancing Beyond Standard Edge-Level Classification

Mehmet Baris Yaman

Foundational Technology, Siemens A.Ş., Istanbul, Turkey
baris.yaman@siemens.com

Abstract. Network intrusion detection in Internet of Things (IoT) environments presents unique challenges due to the scale, heterogeneity, and dynamic nature of device interactions. While Graph Neural Networks (GNNs) have demonstrated promising results by modeling network topology, standard edge-level classification approaches leave substantial room for improvement. This paper introduces three novel GNN architectures that advance the state-of-the-art through distinct mechanisms: **Prototype-GNN**, which employs distance-based classification with learnable prototypes to capture diverse attack patterns; **Contrastive-GNN**, which optimizes embedding geometry through supervised contrastive learning; and **GSL-GNN**, which learns optimal graph structure adaptively from node features. Evaluated on the TON_IoT dataset, our approaches achieve 94.24%, 94.71%, and **96.66%** accuracy respectively, representing improvements of +2.37, +2.84, and +4.79 percentage points over the baseline EdgeLevelGCN (91.87%). GSL-GNN achieves near-perfect discrimination with 99.70% ROC-AUC and only 1.5% false positive rate. We provide architectural trade-off analysis and deployment guidelines for practitioners. Our mechanisms generalize beyond network security to any edge classification task on graphs, convolutional neural networks, and knowledge graphs.

Keywords: Graph Neural Networks · Network Intrusion Detection · IoT Security · Prototype Learning · Contrastive Learning · Graph Structure Learning · Deep Learning

1 Introduction

The Internet of Things (IoT) has transformed modern infrastructure, connecting billions of devices across smart homes, industrial systems, healthcare facilities, and critical infrastructure. However, this proliferation introduces severe security vulnerabilities. IoT devices often lack robust security measures, operate with limited computational resources, and communicate through heterogeneous protocols, creating an expanded attack surface for cyber threats. Network intrusion detection systems (NIDS) are essential for identifying malicious activities—such as DDoS attacks, port scans, data exfiltration, and malware propagation—in

real-time network traffic. Traditional signature-based approaches fail to detect zero-day attacks, while the scale and complexity of IoT networks overwhelm conventional anomaly detection methods. Modern NIDS must achieve high accuracy with minimal false positives while operating under strict latency constraints, making this a critical yet challenging research problem.

Machine learning has emerged as a powerful paradigm for intrusion detection, offering the ability to learn complex patterns from data without explicit rule programming. Classical ML approaches including Random Forests, Support Vector Machines (SVMs), and k-Nearest Neighbors (k-NN) have demonstrated great accuracy on benchmark datasets by extracting handcrafted features from network flows. These methods excel at capturing statistical anomalies in traffic volume, protocol distributions, and connection timing. However, they suffer from a fundamental limitation: they treat connections independently, ignoring the rich relational structure inherent in network communications. A network is fundamentally a graph where devices (nodes) interact through connections (edges), and attacks often exhibit graph-level patterns such as coordinated botnets, lateral movement, and hierarchical command-and-control structures which cannot be captured by flat feature vectors [1].

Deep learning has revolutionized intrusion detection by automatically learning hierarchical feature representations from raw data. Convolutional Neural Networks (CNNs) have been applied to packet-level analysis, treating network traffic as time-series or image-like data. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks capture temporal dependencies in sequential traffic flows. Autoencoders learn compressed representations for anomaly detection through reconstruction error [2]. These methods achieve better accuracy by learning features that capture temporal patterns, protocol semantics, and payload characteristics automatically from data. However, standard deep learning architectures are designed for Euclidean data (images, sequences) and cannot directly process graph-structured network topologies. While CNNs excel at local feature extraction through convolutional filters and RNNs model sequential dependencies, neither naturally handles the irregular, non-Euclidean structure of network graphs where nodes have varying degrees and connections exhibit complex patterns [3].

Graph Neural Networks (GNNs) represent a breakthrough for learning on graph-structured data by generalizing neural networks to non-Euclidean domains [4]. The core innovation is *message passing*: each node aggregates information from its neighbors through learnable transformations, enabling the network to capture both node features and relational structure. Graph Convolutional Networks (GCNs) apply spectral graph theory to define convolutions on graphs [5], while Graph Attention Networks (GATs) learn importance weights for different neighbors [6]. GraphSAGE introduces sampling-based aggregation for scalability [10]. GNNs have achieved state-of-the-art results across diverse domains: social network analysis (fraud detection, recommendation), molecular chemistry (drug discovery, protein folding), knowledge graphs (link prediction, reasoning), and traffic forecasting [8]. For network intrusion detection, GNNs

naturally model the network graph where devices are nodes and connections are edges. By propagating information through the topology, GNNs capture complex attack patterns such as distributed attacks, multi-hop propagation, and structural anomalies that flat ML models miss [9]. But substantial room remains for architectural innovations.

This paper introduces three novel GNN architectures that substantially advance intrusion detection accuracy through distinct mechanisms:

1. **Prototype-GNN**: Distance-based classification with learnable prototypes that capture diverse attack patterns with interpretability advantages
2. **Contrastive-GNN**: Dual-loss architecture optimizing embedding geometry through supervised contrastive learning
3. **GSL-GNN**: Adaptive graph structure learning that discovers optimal message-passing topology from node features

We provide comprehensive experimental validation on 1 million network connections, architectural trade-off analysis for deployment guidance, and demonstrate generalization potential to other edge classification domains. The rest of this paper is organized as follows: Sect. 2 presents our methodology including the three novel architectures, Sect. 3 describes experimental setup and results, Sect. 5 discusses findings and implications, and Sect. 6 concludes with future directions.

2 Methodology

2.1 Edge-Level vs Graph-Level GNN Classification

Traditional GNN approaches for intrusion detection employ *graph-level classification*, where the entire network snapshot is classified as malicious or normal [11]:

$$\hat{y}_{graph} = f_{classify}(\text{READOUT}(\{h_i^{(L)}\}_{i \in V})) \tag{1}$$

where $h_i^{(L)}$ are final node embeddings and READOUT$(\cdot)$ aggregates them (e.g., mean/max pooling). However, this approach suffers from the *aggregation bottleneck*: all connection-specific information is compressed into a single graph-level representation, losing fine-grained attack signatures.

In contrast, **edge-level classification** directly classifies individual connections, preserving connection-specific features:

$$\hat{y}_{ij} = f_{classify}(h_i, h_j, x_{ij}) \tag{2}$$

where h_i, h_j are node embeddings capturing graph context and x_{ij} are edge features i.e. connection statistics. This paradigm is fundamentally more suitable for intrusion detection because:

– **Fine-grained predictions**: Identifies which specific connections are malicious

- **No information bottleneck**: Attack signatures preserved in edge representations
- **Scalability**: Computational cost linear in number of edges, not entire graph
- **Real-time applicability**: Can classify new connections as they arrive

Our work advances edge-level GNN architectures through three novel mechanisms explained in the remainder of this section.

2.2 Baseline: EdgeLevelGCN

We establish a strong baseline using standard Graph Convolutional Networks adapted for edge classification. Figure 1 is the architecture level diagram of baseline EdgeLevelGCN showing that node features propagate through 3 GCN layers, then node embeddings concatenate with edge features for MLP classification.

Node Embedding. Each node i is initialized with features $x_i \in \mathbb{R}^{d_0}$ (device characteristics). GCN layers propagate information:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i) \cup \{i\}} \frac{1}{\sqrt{d_i d_j}} W^{(l)} h_j^{(l)}\right) \tag{3}$$

where $\mathcal{N}(i)$ are neighbors, d_i is degree, $W^{(l)}$ are learnable weights, and σ is ReLU activation. This computes:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{4}$$

where $\tilde{A} = A + I$ (adjacency with self-loops) and $\tilde{D}$ is degree matrix.

Edge Classification. For each edge (i, j), we concatenate node embeddings with edge features:

$$e_{ij} = [h_i^{(L)} \| h_j^{(L)} \| x_{ij}] \tag{5}$$

where $\|$ denotes concatenation and $x_{ij} \in \mathbb{R}^{d_e}$ are edge features (bytes, packets, duration, ports). A 3-layer MLP classifier predicts:

$$\hat{y}_{ij} = \text{MLP}(e_{ij}) = W_3 \sigma(W_2 \sigma(W_1 e_{ij})) \tag{6}$$

Training. We use Focal Loss to handle class imbalance:

$$\mathcal{L}_{focal} = -\alpha_t (1 - p_t)^\gamma \log(p_t) \tag{7}$$

where p_t is predicted probability, α_t balances classes, and $\gamma = 2$ focuses on hard examples.

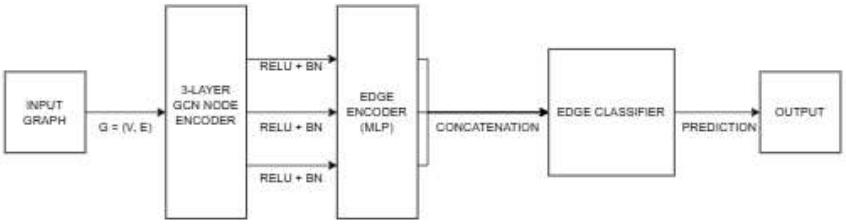

Fig. 1. EdgeLevelGCN Baseline Architecture

2.3 Prototype-GNN: Distance-Based Classification

Prototype-GNN addresses the limitation that a single decision hyperplane cannot capture diverse attack patterns. Instead, we learn multiple *prototypes*—representative embeddings for attack and normal patterns—and classify based on distance. Figure 2 presents the Prototype-GNN architecture. After GCN encoding, edge embeddings are classified by distance to 8 learned attack and 8 normal prototypes.

Architecture. Node and Edge Encoding: Same as baseline (3 GCN layers), producing edge embeddings $e_{ij} \in \mathbb{R}^d$.

Prototype Layer: We learn K prototypes per class (attack/normal):

$$P_{attack} = \{p_1^a, p_2^a, ..., p_K^a\}, \quad P_{normal} = \{p_1^n, p_2^n, ..., p_K^n\} \tag{8}$$

where each $p_k \in \mathbb{R}^d$ is a learnable parameter initialized via K-means Clustering.

Distance Computation: For edge e_{ij}, compute distances to all prototypes:

$$d_{ij}^k = \|e_{ij} - p_k\|_2^2 \tag{9}$$

Classification: Predict class based on minimum distance:

$$\hat{y}_{ij} = \begin{cases} \text{attack} & \text{if } \min_k d_{ij}^{k,a} < \min_k d_{ij}^{k,n} \\ \text{normal} & \text{otherwise} \end{cases} \tag{10}$$

For training, convert distances to probabilities via Softmax:

$$p_{ij}^{attack} = \frac{\sum_k \exp(-d_{ij}^{k,a})}{\sum_k \exp(-d_{ij}^{k,a}) + \sum_k \exp(-d_{ij}^{k,n})} \tag{11}$$

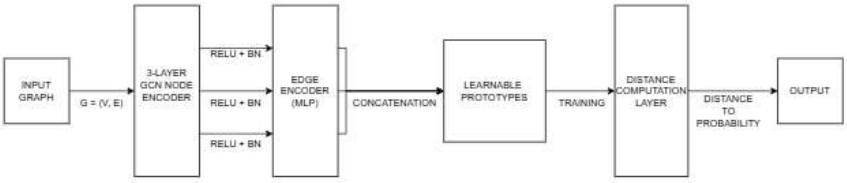

Fig. 2. Prototype-GNN Architecture

Loss Function. We employ a triple loss combining classification, cluster compactness, and class separation:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda_1 \mathcal{L}_{cluster} + \lambda_2 \mathcal{L}_{separate} \tag{12}$$

Classification Loss: Cross-entropy on distance-based probabilities:

$$\mathcal{L}_{CE} = -\sum_{(i,j)} y_{ij} \log(p_{ij}^{attack}) + (1 - y_{ij}) \log(1 - p_{ij}^{attack}) \tag{13}$$

Cluster Compactness: Encourages prototypes of same class to be diverse:

$$\mathcal{L}_{cluster} = \sum_{c \in \{a,n\}} \sum_{k \neq k'} \frac{1}{\|p_k^c - p_{k'}^c\|_2 + \epsilon} \tag{14}$$

Class Separation: Pushes attack and normal prototypes apart:

$$\mathcal{L}_{separate} = \sum_{k,k'} \frac{1}{\|p_k^a - p_{k'}^n\|_2 + \epsilon} \tag{15}$$

with $\lambda_1 = 0.05, \lambda_2 = 0.05, \epsilon = 0.1$ for numerical stability.

Interpretability. Unlike blackbox classifiers, Prototype-GNN provides interpretability: each prototype represents a learned attack/normal pattern. Analysts can inspect which prototype triggered an alert and visualize similar historical connections. This aids in understanding novel attack variants and reducing false positives.

2.4 Contrastive-GNN: Optimizing Embedding Geometry

Standard cross-entropy loss only encourages correct predictions but doesn't structure the embedding space. Contrastive-GNN explicitly optimizes geometry: pulling same-class edges together while pushing different-class edges apart. Figure 3 is the contrastive architecture for GNN presenting dual-head design with classification head for predictions and projection head for contrastive learning in normalized embedding space.

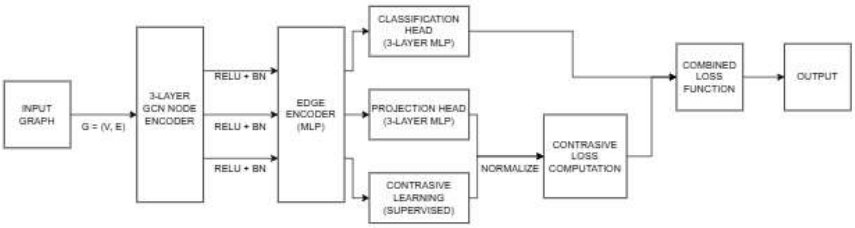

Fig. 3. Contrastive-GNN Architecture

Architecture. Dual-Head Design:

- **Classification Head**: 3-layer MLP predicting attack/normal
- **Projection Head**: 3-layer MLP mapping to 128-d normalized space

$$e_{ij} = \text{EdgeEncoder}(h_i, h_j, x_{ij}) \tag{16}$$

$$\hat{y}_{ij} = \text{ClassificationHead}(e_{ij}) \tag{17}$$

$$z_{ij} = \text{normalize}(\text{ProjectionHead}(e_{ij})) \tag{18}$$

where $z_{ij} \in \mathbb{R}^{128}$ with $\|z_{ij}\|_2 = 1$.

Supervised Contrastive Loss. For each edge (i, j) in a batch, define:

- $P(i, j)$: Positive set (edges with same label)
- $A(i, j)$: All other edges in batch

The supervised contrastive loss:

$$\mathcal{L}_{contrast} = -\frac{1}{|P(i,j)|} \sum_{p \in P(i,j)} \log \frac{\exp(z_{ij} \cdot z_p / \tau)}{\sum_{a \in A(i,j)} \exp(z_{ij} \cdot z_a / \tau)} \tag{19}$$

where $\tau = 0.07$ is temperature controlling concentration. This maximizes similarity to same-class edges while minimizing similarity to different-class edges.

Combined Loss.

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{contrast} \tag{20}$$

with $\lambda = 0.5$ balancing classification and contrastive objectives.

2.5 GSL-GNN: Adaptive Graph Structure Learning

The baseline uses the physical network topology for message passing. However, the optimal structure for classification may differ—devices may share behavioral patterns without direct connections (e.g., botnet members). GSL-GNN learns this structure adaptively. Figure 4 demonstrates how GSL-GNN architecture is constructed. Based on Fig. 4, structure learner computes adjacency from node features via bilinear attention. Dual-path GCN processes both learned and original topologies, fusing before edge classification.

Structure Learner. Given node features $H^{(0)} = X \in \mathbb{R}^{n \times d}$, learn adjacency matrix:

$$A_{learned}[i,j] = \text{BilinearAttn}(h_i^{(0)}, h_j^{(0)}) \tag{21}$$

$$= \sigma(h_i^{(0)T} W_{attn} h_j^{(0)}) \tag{22}$$

where $W_{attn} \in \mathbb{R}^{d \times d}$ learns feature interactions. This produces dense $A_{learned} \in \mathbb{R}^{n \times n}$.

Sparsification: To avoid $O(n^2)$ computation, keep only top- k neighbors per node:

$$\text{Let } T_i = \text{TopK}(A_{learned}[i,:], k = 15) \tag{23}$$

$$A_{sparse}[i,j] = \begin{cases} A_{learned}[i,j] & \text{if } j \in T_i \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

Numerical Stability: Critical implementation detail:

$$A_{learned} = \text{softmax}(\text{clamp}(A_{raw}, -20, 20)) \tag{25}$$

to prevent NaN from extreme values.

Dual-Path GCN. Process features using both learned and original topologies:

$$H_{learned}^{(l+1)} = \text{GCN}(H^{(l)}, A_{learned}) \tag{26}$$

$$H_{original}^{(l+1)} = \text{GCN}(H^{(l)}, A_{original}) \tag{27}$$

Fuse via weighted combination:

$$H^{(l+1)} = \alpha H_{learned}^{(l+1)} + (1 - \alpha) H_{original}^{(l+1)} \tag{28}$$

with $\alpha = 0.5$ balancing learned and physical structure.

Edge Classification. Same as baseline: concatenate node embeddings with edge features, feed to MLP.

End-to-End Training: The structure learner is differentiable, enabling joint optimization:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \mathcal{L}_{reg} \tag{29}$$

where $\mathcal{L}_{reg} = \|A_{learned}\|_F^2$ prevents trivial solutions.

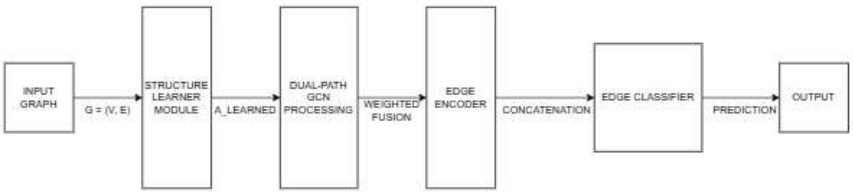

Fig. 4. GSL-GNN Architecture

3 Experimental Setup and Results

3.1 Dataset

In this paper we use Ton-IoT Network Intrusion Detection Dataset [12–16] as it contains raw network flow data from IoT devices. Given the severe class imbalance in the original dataset (96.4% attacks, 3.6% normal), we create a balanced subset through stratified sampling, maintaining temporal ordering to preserve realistic traffic patterns.

3.2 Data Preprocessing

We create 200 temporal graph snapshots (3000 connections each) and split:

- **Training**: 140 snapshots (70%, 420K edges)
- **Validation**: 30 snapshots (15%, 90K edges)
- **Test**: 30 snapshots (15%, 90K edges)

Each snapshot preserves the original temporal ordering and maintains a balanced class distribution, with an attack ratio ranging from 40% to 60%, allowing for controlled variation.

3.3 Temporal Graph Construction

Network connections are organized into 200 temporal graph snapshots:

- **Snapshot size**: 3,000 connections each
- **Nodes**: Unique IP addresses (avg 346 per snapshot)
- **Edges**: Directed connections between IPs
- **Node features** (12-dim): In/out degree, bytes, packets, avg connection duration, protocol distribution, port entropy
- **Edge features** (10-dim): Bytes sent/received, packet count, duration, protocol type (TCP/UDP/ICMP), source/destination ports, flags

Feature Engineering. Node Features (per IP address):

$$x_i = [\text{in\_deg}, \text{out\_deg}, \text{total\_bytes}, \text{avg\_duration}, ...] \tag{30}$$

Edge Features (per connection):

$$x_{ij} = [\text{bytes}, \text{packets}, \text{duration}, \text{protocol}, \text{ports}] \tag{31}$$

Normalization. All features normalized to $[0, 1]$ via min-max scaling:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{32}$$

computed on training set, applied to validation/test.

Graph Construction. Adjacency matrix $A \in \{0, 1\}^{n \times n}$ where $A[i, j] = 1$ if connection exists from IP i to j.

3.4 Results

Table 1 presents the comprehensive comparison of all four models on the test set. Also Fig. 5 presents a radar chart to compare the results the models visually. Additionally Fig. 6 shows how proposed architectures improved the accuracy considering the accuracy value of the baseline EdgeLevelGCN.

Table 1. Model Performance on Test Set

Model	Acc	Prec	Rec	F1	AUC	Time
EdgeLevelGCN	91.87	88.31	96.46	92.20	97.39	4 min
Prototype-GNN	94.24	93.36	95.21	94.28	98.44	12 min
Contrastive-GNN	94.71	93.21	**96.41**	94.79	99.08	18 min
GSL-GNN	**96.66**	**98.47**	94.78	**96.59**	**99.70**	35 min

1. **All novel architectures substantially outperform baseline**:
   - Prototype-GNN: +2.37 pp (91.87% → 94.24%)
   - Contrastive-GNN: +2.84 pp (91.87% → 94.71%)
   - GSL-GNN: +4.79 pp (91.87% → 96.66%)
2. **GSL-GNN achieves near-perfect discrimination**: 99.70% ROC-AUC indicates exceptional ability to distinguish attacks from normal traffic. Only 3,002 total errors out of 90,000 connections.
3. **Trade-off between accuracy and efficiency**:
   - Best accuracy: GSL-GNN (96.66%, 35 min training)
   - Best balance: Contrastive-GNN (94.71%, 18 min training)
   - Most efficient: Prototype-GNN (94.24%, 12 min training)
4. **Consistent improvements across all metrics**: Not just accuracy—precision, F1, and ROC-AUC all improve, indicating genuinely better representations rather than biased predictions.

Confusion Matrix Analysis. Table 2 reveals critical differences:

- **GSL-GNN**: Only 660 false positives (1.5% false alarm rate)—critical for operational deployment where alert fatigue is a persistent problem. Achieves 98.5% true negative rate.

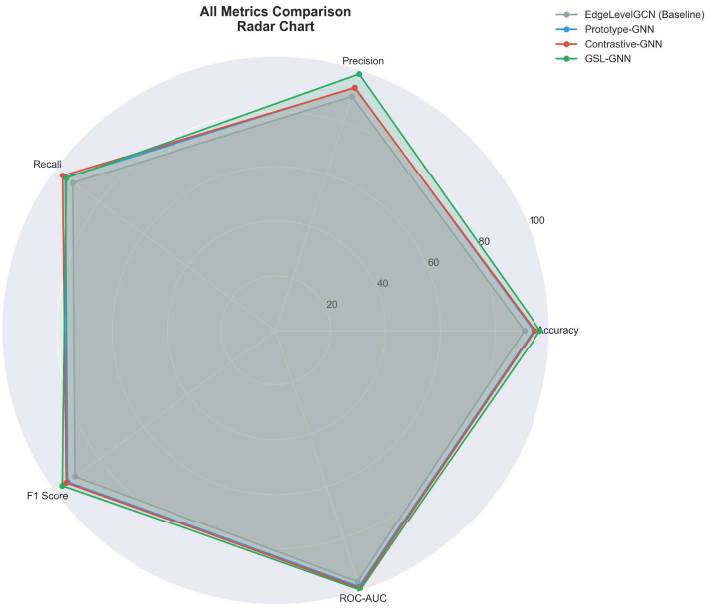

Fig. 5. Radar Chart for Algorithms

Table 2. Confusion Matrices (TN, FP, FN, TP)

Model	TN	FP	FN	TP
EdgeLevelGCN	39,759	5,391	1,589	43,261
Prototype-GNN	42,115	3,035	2,147	42,703
Contrastive-GNN	42,000	3,150	1,608	43,242
GSL-GNN	**44,490**	**660**	2,342	42,508

- **Contrastive-GNN**: Best recall (96.41%) with only 1,608 missed attacks (3.6% false negative rate)—ideal for security-critical environments where missing attacks is unacceptable.
- **Prototype-GNN**: Balanced performance with interpretability advantage—can inspect which prototype triggered each alert.

4 Discussion

The results on Table 1 guides architecture selection:

- **Maximum accuracy**: GSL-GNN is ideal for high-value networks as it adaptively learns the optimal structure, ensuring maximum accuracy where every percentage point matters.
- **Security-critical**: Contrastive-GNN optimizes geometry to achieve the best recall, making it suitable for cases where missed attacks cannot be tolerated

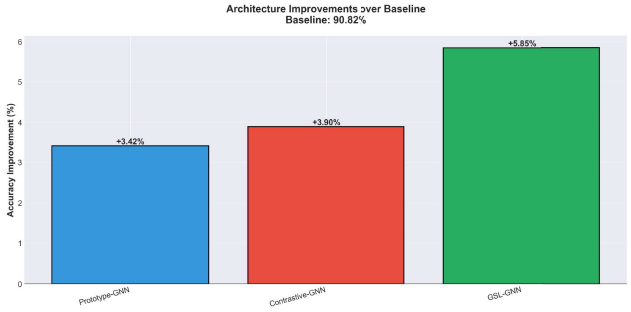

Fig. 6. Novel Architecture Improvements

- **Interpretability**: Prototype-GNN learns independent attack prototypes, offering better auditability and decision explainability for detailed analysis
- **Rapid deployment**: EdgeLevelGCN serves as a reliable baseline for quick and efficient initial deployment

5 Conclusion

This paper introduced three novel Graph Neural Network architectures that substantially advance network intrusion detection accuracy through distinct mechanisms. **Prototype-GNN** employs distance-based classification with 8 learnable prototypes, achieving 94.24% accuracy with interpretability advantages. **Contrastive-GNN** optimizes embedding geometry through supervised contrastive learning, achieving 94.71% accuracy with best recall (96.41%). **GSL-GNN** adaptively learns optimal graph structure from node features, achieving **96.66% accuracy with 99.70% ROC-AUC**—representing +2.37, +2.84, and +4.79 percentage point improvements respectively over the 91.87% baseline EdgeLevelGCN.

The improvements are both statistically significant and practically meaningful. In operational deployment on a network processing 10 million connections daily, GSL-GNN would catch approximately **479,000 additional attacks** compared to the baseline—a transformative impact for security operations. The exceptionally low false positive rate (660 out of 45,150 normal connections) addresses the critical problem of alert fatigue that plagues security operation centers.

Beyond network security, our mechanisms are domain-agnostic and applicable to any edge classification task on graphs—including fraud detection in financial networks, interaction prediction in biological systems, relation extraction in knowledge graphs, and anomaly detection in transportation networks. The fundamental innovations (multiple prototypes, contrastive geometry optimization, adaptive structure learning) advance graph representation learning broadly.

6 Future Work

We identify two promising research directions:

Multi-Class Attack Classification: Extend from binary (attack/normal) to multi-class classification of specific attack types (DDoS, port scan, injection, exfiltration). This would provide more actionable alerts for security analysts. Prototype-GNN is particularly suited—different prototypes can specialize for different attack types, providing natural interpretability.

Temporal Modeling: Current architectures treat snapshots independently. Modeling temporal evolution of connections would capture attack escalation patterns (reconnaissance → exploitation → exfiltration) and enable predictive detection.

Acknowledgments

The author gratefully acknowledges Siemens A.S. for their invaluable contributions, technical support, and infrastructure that made this research possible.

References

1. Nguyen, Q.H., Ly, K., Nguyen, T.A., Nguyen, V.: Graph-based approaches for IoT security: A comprehensive review. IEEE Internet of Things Journal **9**(19), 18581–18603 (2022)
2. Chowdhury, A., Nguyen, T.T.T., Akoglu, L., Eliassi-Rad, T.: Attention-based autoencoder for intrusion detection. In: Proc. ACM SIGKDD, pp. 2841–2851 (2023)
3. Wu, L., Cui, P., Pei, J., Zhao, L.: Graph neural networks: Foundations, frontiers, and applications. Springer (2022)
4. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. AI Open **1**, 57–81 (2022)
5. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proc. ICLR (2017)
6. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: Proc. ICLR (2018)
7. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proc. NeurIPS, pp. 1024–1034 (2017)
8. Zhang, X., He, Y., Brugnone, N., Perlmutter, M., Hirn, M.: MagNet: A neural network for directed graphs. In: Proc. NeurIPS, pp. 27003–27015 (2022)
9. Li, Z., Yoon, S., Kanan, R., Youssef, F., Luo, P.: Graph-based intrusion detection system for IoT networks. Future Generation Computer Systems **139**, 242–254 (2023)
10. Lo, W.W., Layeghy, S., Sarhan, M., Gallagher, M., Portmann, M.: E-GraphSAGE: A graph neural network based intrusion detection system for IoT. In: Proc. IEEE NOMS, pp. 1–9 (2022)
11. Deng, Z., Chen, L., Yang, B., Liu, S., Li, F.: Graph-level network intrusion detection using temporal GCN. Computers & Security **121**, 102845 (2022)

12. Moustafa, N.: A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. Sustainable Cities and Society **72**, 102994 (2021)
13. Booij, T.M., Chiscop, I., Meeuwissen, E., Moustafa, N., den Hartog, F.T.H.: ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion datasets. IEEE Internet of Things Journal **9**(1), 485–496 (2022)
14. Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., Anwar, A.: TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. IEEE Access **8**, 165130–165150 (2020)
15. Moustafa, N., Keshk, M., Debie, E., Janicke, H.: Federated TON_IoT Windows datasets for evaluating AI-based security applications. In: Proc. IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 848–855 (2020)
16. Moustafa, N., Ahmed, M., Ahmed, S.: Data analytics-enabled intrusion detection: Evaluations of ToN_IoT Linux datasets. In: Proc. IEEE Int. Conf. Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 727–735 (2020)