JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 31 No. 3, pp. 21-35, March 2026
https://doi.org/10.9708/jksci.2026.31.03.021

Semantic Tabular-to-Image Conversion and Contrastive Learning for Lightweight Intrusion Detection

Jun Yeong Park*, Kunwoo Kang*, Hoin Lee*, Seungeun Lee*, Yu Rang Park**

*Graduate Student, Dept. of Digital Analytics, Yonsei University, Seoul, Korea
**Professor, Dept. of Biomedical Systems Informatics & Digital Analytics, Yonsei University, Seoul, Korea

[Abstract]

With the proliferation of IoT emphasizing the need for high-performance Intrusion Detection Systems (IDS) in edge environments, deep learning-based IDS research is actively pursued; However, existing approaches that directly utilize tabular data as input for deep learning models are limited in their ability to capture the complex inherent relationships of network traffic. To address this, we propose a 3-stage lightweight IDS framework that converts tabular data into images to leverage the CNN. (1) First, feature selection based on Shapley Additive exPlanations (SHAP) is performed to compress data by retaining only critical features. (2) The selected data is transformed into images using the LLM-categorized Vortex Feature Positioning (LVFP) technique, which reconstructs tabular data into CNN-optimized spatial patterns by assigning semantically categorized feature groups to RGB channels and rearranging them through vortex feature allocation. (3) Finally, we construct a lightweight CNN encoder and pre-train it on the converted images via contrastive learning to establish generalizable feature representations. As downstream tasks, evaluations on 6 IDS & IoT benchmark datasets demonstrate that the proposed model outperforms existing models while using a minimal number of parameters.

▸ **Key words**: Intrusion Detection System, Tabular-to-Image, Lightweight Model, LLM, Contrastive Learning

[요 약]

사물인터넷의 확산으로 엣지 환경에서의 고성능 침입 탐지 시스템(IDS)의 필요성이 강조됨에 따라, 딥러닝 기반의 IDS 연구가 활발히 진행되고 있다. 그러나 정형 데이터를 딥러닝 모델의 입력으로 직접 사용하는 기존 접근 방식은 네트워크 트래픽에 내재된 복잡한 관계를 포착하는 데 한계가 있다. 이에 본 연구는 정형 데이터를 이미지로 변환하여 CNN을 활용하는 3단계 경량 침입 탐지 사전학습 프레임워크를 제안한다. (1) 먼저, SHAP기반의 특징 선택을 통해 침입 탐지에 중요한 특징만을 유지함으로써 데이터를 압축한다. (2) 선별된 데이터는 LVFP 기법을 통해 이미지로 변환된다. 이 기법은 의미론적으로 범주화된 특징 그룹을 RGB 채널에 할당하고 소용돌이 특징 배치를 통해 재배열함으로써, 정형 데이터를 CNN에 최적화된 공간적 패턴으로 재구성한다. (3) 이후 경량 CNN 인코더를 구축하여 변환된 이미지에 대해 대조 학습으로 사전학습시킴으로써, 일반화 가능한 특징 표현을 구축한다. 다운스트림 작업으로서 6종의 IDS & IoT 벤치마크 데이터셋에 대해 평가한 결과, 제안한 모델은 최소한의 파라미터 수만으로도 기존 모델을 능가하는 성능을 보였다.

▸ **주제어**: 침입 탐지 시스템, 정형-이미지 변환, 거대 언어 모델, 대조 학습

• First Author: Jun Yeong Park, Kunwoo Kang, Corresponding Author: Yu Rang Park
*Jun Yeong Park (pjk000011@yonsei.ac.kr), Dept. of Digital Analytics, Yonsei University
*Kunwoo Kang (kunwoo2027@yonsei.ac.kr), Dept. of Digital Analytics, Yonsei University
*Hoin Lee (hoini0920@yonsei.ac.kr), Dept. of Digital Analytics, Yonsei University
*Seungeun Lee (seungeun430@yonsei.ac.kr), Dept. of Digital Analytics, Yonsei University
**Yu Rang Park (yurangpark@yuhs.ac), Dept. of Biomedical Systems Informatics & Digital Analytics, Yonsei University
• Received: 2025. 12. 18, Revised: 2026. 03. 03, Accepted: 2026. 03. 04.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

With the rapid proliferation of Internet of Things (IoT) devices, the scale and complexity of network traffic continue to grow, creating an environment in which numerous sensors and lightweight devices constantly exchange data. While this expansion improves service convenience, it also introduces a widening range of potential vulnerabilities that attackers can exploit. In practice, threats targeting IoT ecosystems—such as DDoS, Mirai botnet, and reconnaissance attacks—are steadily increasing[1], intensifying the need for security technologies capable of continuously analyzing network-wide behavior and detecting anomalous patterns[2].

In response to these security demands, intrusion detection systems (IDS) play a central role in monitoring IoT traffic and identifying abnormal patterns. Traditional IDS research relies on machine learning-based classification models that utilize network traffic data structured in a tabular format[3]. However, these conventional methods depend on manual feature engineering to extract meaningful patterns[4]. Furthermore, as the number of features increases, the tabular representation often suffers from information dilution, which further degrades model performance[5]. To address these limitations, Deep Learning (DL) models have been widely adopted in IDS, leveraging their deep architectures to automatically discover latent patterns and effectively model intricate features without explicit manual intervention[6].

Among deep learning architectures, Convolutional Neural Networks (CNN) effectively capture local dependencies and hierarchical patterns through convolutional operations, which has motivated the extensive adoption of CNN-based models in IDS research[7]. To make tabular data compatible with CNN, recent research have attempted to reshape traffic features into image formats[8]. However simply reshaping tabular traffic data into an image format results in arbitrary spatial adjacencies that fail to reflect semantic relationships, thereby limiting CNNs' ability to leverage their inductive biases for detecting attack behaviors. To address this, approaches such as DeepInsight[9], IGTD[10], and VFP[11] rearrange features into semantically informed 2D image representations that explicitly preserve inter-feature relationships, enabling CNNs to extract localized patterns more effectively. These methods demonstrate the potential of well-structured spatial representations in improving performance on tabular IDS data.

Despite these advancements, a critical limitation remains: these transformation techniques are designed for generic tabular data and thus struggle to represent the unique structural characteristics of IDS traffic. IDS datasets include numerous heterogeneous features with nonlinear dependencies across temporal, protocol-level, and behavioral dimensions. When such data are transformed using conventional methods, these complex relationships are often distorted or lost, removing subtle cues needed to distinguish attack behaviors and ultimately degrading detection performance[12].

Beyond structural challenges, the substantial label imbalance typical of IDS datasets, where normal and attack traffic occur at highly uneven frequencies and rare attack types are severely underrepresented. This imbalance limits supervised learning, making it difficult for models to learn robust representations or detect infrequent threats[10]. In this context, Self-Supervised Learning (SSL) mitigates this limitation by learning generalizable representations from large-scale unlabeled traffic data, followed by fine-tuning for downstream tasks[13-14].

Despite the rising importance of tabular-to-image conversion and SSL, the IDS domain lacks integrated approaches for enhanced representation. To bridge this gap, we propose a lightweight framework utilizing LLM-categorized Vortex Feature Positioning (LVFP). Specifically, an LLM

categorizes semantically related features into RGB channels, which are then arranged in a vortex structure to generate spatially optimized patterns for CNN. Subsequently, a lightweight CNN encoder is pre-trained via contrastive learning on large-scale unlabeled data, and the resulting pre-trained model is then fine-tuned for downstream tasks, leading to improved detection performance.

Our contributions are summarized as follows:

1. We introduce LLM-categorized Vortex Feature Positioning (LVFP), a novel tabular-to-image transformation pipeline that leverages LLMs to categorize features into RGB channels and applies vortex allocation to generate CNN-optimized spatial patterns.

2. We construct a lightweight CNN encoder initialized via contrastive Learning on the transformed images, enabling the model to learn robust and generalizable traffic representations from large-scale unlabeled data.

3. Although the proposed model is lightweight, it consistently outperforms existing deep learning models across 6 benchmark datasets.

II. Preliminaries

1. Related works

1.1 Intrusion Detection System

Early IDS research utilizing traditional classifiers such as SVM and Random Forest [15] heavily relied on manual feature engineering[16]. These methods often suffer from the curse of dimensionality, where expanding feature spaces lead to computational inefficiency and performance degradation, limiting generalization against unknown attacks.

To address these limitations, deep learning models including Autoencoders and LSTMs were introduced to capture non-linear and temporal dependencies[17-18]. Building upon these advances, recent IDS research has increasingly focused on tabular-specific architectures such as TabNet [19] and TabTransformer[20], which leverage attention mechanisms to model complex feature interactions[22]. Furthermore, general-purpose tabular foundation models have recently been explored in IDS contexts. TabPFN and TabICL demonstrated consistent improvements over traditional ensemble methods across multiple cybersecurity datasets (e.g., CICIDS2017, UNSW-NB15)[23], while TabPFN-based IDS for Industrial IoT achieved strong F1-scores even with limited training data[24]. However, despite their modeling capacity, these foundation models rely on generic priors and incur substantial inference costs, rendering them impractical for domain-specific, resource-constrained environments such as IoT edge devices.

This trade-off between model expressiveness and computational efficiency is further evidenced across widely adopted IDS benchmark datasets. On CICIDS2017, hybrid CNN-LSTM architectures achieved 99.8% accuracy but with significant computational overhead[37]. Similarly, on CICIoT2023, ensemble methods combining Autoencoders and CNNs reported detection accuracies exceeding 99%, yet their stacked designs introduce substantial parameter counts unsuitable for edge deployment[39]. For Edge-IIoTset, Vision Transformer-based approaches demonstrated near-perfect detection performance, but their heavy backbones render them infeasible for real-time scenarios[25]. In contrast, lightweight approaches such as MLP-based models achieved up to 99.9% accuracy on WUSTL-IIoT with reduced complexity[26], and autoencoder-based dimensionality reduction followed by random forest classifier yielded favorable detection accuracy on UNSW-NB15 with comparatively low inference costs[38]. These prior works highlight a persistent challenge: deep models offer high expressiveness but at prohibitive costs, while lightweight models often sacrifice robustness. Beyond supervised paradigms, Self-Supervised Learning (SSL) has been adopted to mitigate label dependency via

reconstruction or contrastive objectives [13-14]. Yet, most existing SSL approaches rely on raw tabular data, which fails to capture structural relationships or semantic invariance, ultimately restricting generalization across downstream tasks.

To address these limitations, our work proposes a tabular-to-image conversion coupled with lightweight contrastive pre-training, bridging the gap between representational richness and computational efficiency for edge-based IDS deployment.

1.2 Tabular-to-Image methods

DeepInsight[9] maps features into a 2D space using t-SNE, but this often distorts the intrinsic structure of features as such projection methods do not necessarily preserve domain-specific semantics. IGTD[10] optimizes pixel arrangement based on feature correlation matrices; however, its reliance on statistical relationships fails to capture the heterogeneous nature and semantic hierarchy unique to IDS data. VFP[11] calculates the cumulative Pearson Correlation Coefficient (PCC) for each feature and arranges them in a vortex pattern, positioning features with lower correlations near the center to maximize convolutional operations on independent attributes. Yet, VFP is limited by its single-channel approach, where the generated grayscale image is merely replicated to form RGB channels. This fails to fully exploit the multi-channel architecture of CNN for representing diverse feature attributes. Consequently, these methods remain insufficient for capturing the domain-aware feature grouping and complex correlation structures required for intrusion detection.

III. Proposed Method

In this section, we propose a comprehensive framework for network intrusion detection leveraging a lightweight pre-trained model. The process initiates with SHAP-based feature selection to ensure computational efficiency. Following this, as illustrated in Fig. 1, the workflow proceeds through 3 stages: 1) applying a tabular-to-image conversion to semantically rearrange 1D features into 2D spatial structures, 2) conducting lightweight contrastive pre-training optimized for resource-constrained IoT environments, and 3) fine-tuning the encoder and classifier on downstream tasks.

1. Feature Selection

Datasets for Network Intrusion Detection Systems (IDS) are inherently high-dimensional, comprising dozens of features. However, many of these features may be redundant or act as noise that hinders model generalization. To address this, we employ Explainable AI (XAI) techniques based on SHapley Additive exPlanations (SHAP) to perform feature selection, enabling the model to retain essential information while reducing dimensionality. Specifically, we utilize XGBoost, tree-based ensemble model, to initially train on the dataset. Subsequently, SHAP values are computed to quantify the contribution of each feature, enabling an evaluation of importance that accounts for complex inter-feature dependencies. We rank the features based on the mean absolute SHAP value across all data samples and eliminate the bottom 5% of features deemed to have negligible contribution to detection performance, thereby optimizing the input data for the model.

2. Tabular-to-Image Conversion

To implement tabular-to-image conversion optimized for CNN-based deep learning models, we propose the LLM-categorized Vortex Feature Positioning (LVFP) method. This comprehensive approach integrates 3 key phases: 1) LLM-guided Feature Categorization to maximize cross-channel interactions, 2) correlation-based vortex positioning to overcome the inherent lack of spatial information, and 3) spatial dimension standardization to ensure uniform input dimensions across heterogeneous

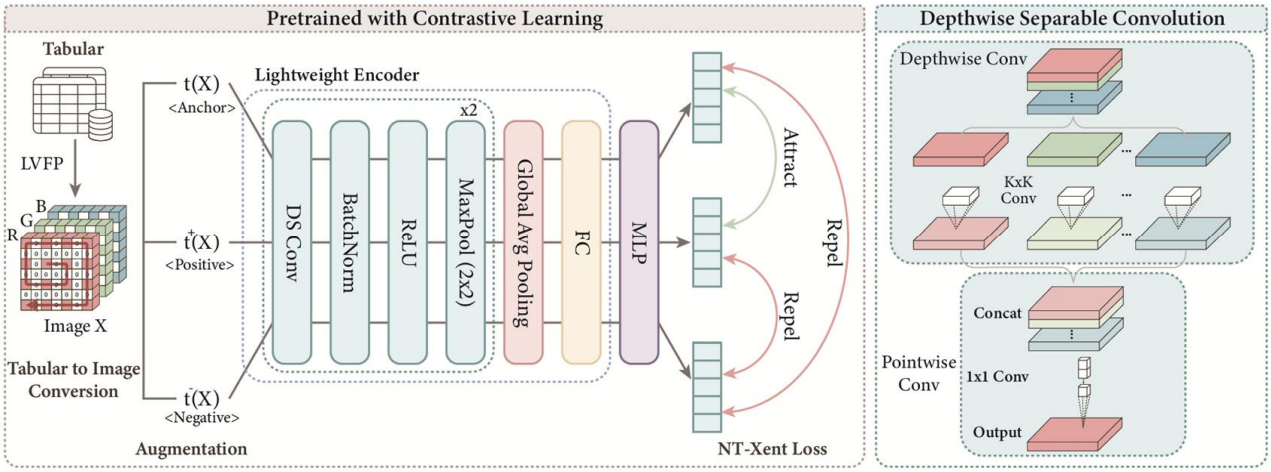

Fig. 1. Overall Framework

datasets.

2.1 LLM-guided Feature Categorization

In the first phase of LVFP, We partition the input vector $x \in \mathbb{R}^N$ consisting of N features into three independent subsets F_R, F_G, and F_B based on the characteristics of the network security domain. Each group is mapped to the Red, Green, and Blue channels of the image, respectively, according to the following criteria:

$$\text{Channel}(f_i) = \begin{cases} \text{R if } f_i \in \text{Physical Magnitude (Flow/Volume)} \\ \text{G if } f_i \in \text{Intrinsic Semantics (Protocol/Flag)} \\ \text{B if } f_i \in \text{Behavioral Dynamics (Pattern/Stats)} \end{cases} \quad (1)$$

Given the heterogeneity of IDS datasets, manually analyzing and allocating every feature to RGB channels is not only inefficient but also prone to inconsistency. To address this challenge, we employed an LLM, specifically Gemini 3.0 Pro, as a domain expert agent. In this process, we formulated the previously defined classification criteria for R, G, and B channels into structured instructions and provided them to the LLM. To ensure reproducibility, the LLM is configured with temperature=0.0 to eliminate stochastic variability, guaranteeing identical outputs for the same input across multiple runs. This automated approach minimizes human intervention, thereby ensuring the objectivity of the classification process.

Critically, this categorization is performed once per dataset as an offline preprocessing step, with the resulting mapping frozen as a static configuration file in the codebase. Consequently, the LLM incurs zero computational overhead and no recurring API costs during model training or inference, addressing practical deployment concerns. The complete prompt structure and full categorization results for all datasets are provided in Appendix A. This channel allocation strategy facilitates non-linear interactions among Physical Magnitude (R), Intrinsic Semantics (G), and Behavioral Dynamics (B) during convolutional operations, ultimately enhancing detection performance.

2.2 Vortex Positioning based on Correlation

Pixel arrangement within each channel follows the Vortex Feature Positioning (VFP)[11], utilizing the Pearson Correlation Coefficient (PCC) to quantify pairwise linear dependencies among features. The importance score S_i of each feature f_i is defined as the sum of the absolute correlation coefficients between f_i and all other features, as shown in Eq. (2):

$$S_i = \sum_{j \neq i} |\text{PCC}(f_i, f_j)| \quad (2)$$

Features with lower S_i (higher uniqueness) are positioned near the image center to align with the CNN's central receptive field, while highly correlated redundant features are placed at the periphery. Additionally, we apply a distancing technique that inserts zero-padding between

adjacent pixels as spatial buffers to minimize feature interference and preserve distinctiveness.

2.3 Spatial Dimension Standardization

To address resolution disparities caused by variable feature counts, we first align channels via zero-padding with centering, scaling them to the maximum dimension within a single image.

Subsequently, to standardize spatial dimensions across heterogeneous datasets used for pre-training, we adjust all inputs to the maximum global resolution observed using nearest neighbor interpolation. This unified dimension is maintained for downstream tasks, where all target datasets are interpolated to match the pre-trained encoder's input specification.

3. Pretrained with Contrastive Learning

To derive robust feature representations from large-scale network traffic, we employ a Contrastive Learning-based SSL framework. Unlike generative approaches, this method learns meaningful latent structures by distinguishing between similar and dissimilar samples. The encoder integrates Depthwise Separable Convolution(DS Conv) to minimize computational cost. For training, we utilize the NT-Xent Loss[27] to maximize the similarity between augmented views, thereby capturing inherent invariance and discriminative patterns of network traffic, as shown in Eq. (3),

$$\ell_{i,j} = -\log \frac{\exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(sim(z_i, z_k)/\tau)} \quad (3)$$

where N is the batch size, z denotes the feature vector, τ is the temperature, and $sim(\cdot,\cdot)$ represents cosine similarity.

IV. Experiments

1. Experimental Setup

1.1 Datasets

We evaluate our model's performance on 6 recent benchmark datasets. For pre-training, the CICIDS2017[28] and CICIoT2023[29] datasets are utilized, as they exhibit low class imbalance, encompass a wide range of modern attack scenarios, and provide flow-based statistical information. For downstream tasks, we employ all 6 datasets—Edge-IIoTset[30], RT-IoT[31], WUSTL-IIoT[32], UNSW-NB15[33], as well as the 2 datasets used for pre-training (CICIDS2017 and CICIoT2023).

1.2 Baselines

To evaluate performance against standard architectures, we employ LightGBM[34] as a representative tree-based baseline, alongside deep learning models such as CNN[35] and LSTM[36]. The deep learning models are trained for 20 epochs. TabNet[19] and TabTransformer[20] are included as specialized models for tabular data that utilize attention mechanisms to model feature interactions, trained for 50 epochs with early stopping applied.

1.3 Implementation Details

We utilize Min-Max normalization and Label Encoding to process numerical and categorical features, respectively, which are subsequently transformed into spatially synchronized 11×11 RGB images. The lightweight encoder employs two Depthwise-Separable Convolution blocks followed by a two-layer MLP projection head to generate 32-dimensional embeddings.

For pre-training, we minimize the NT-Xent loss with a temperature of 0.07 for 50 epochs, utilizing the Adam optimizer with a learning rate of 1e-3 and a batch size of 1,024. During this phase, we apply data augmentations including Random Resized Crop, Horizontal Flip, Color Jitter, and Gaussian Blur.

For downstream tasks, we fine-tune the encoder and the linear classifier jointly for 20 epochs, utilizing the Adam optimizer with a learning rate of 5e-4 and a batch size of 512. To address memory constraints, we sampled 60,000 instances per class

Table 1. Comparison with existing models on 6 datasets for binary classification, measured by (accuracy, weighted f1-score). The best performance is shown in **bold** and the second-best is underlined.

Dataset	LightGBM[34]	CNN[35]	LSTM[36]	TabNet[19]	TabTrans[20]	**Ours**
CICIoT2023	(**96.8**, **96.8**)	(91.2, 90.9)	(91.7, 91.7)	(90.9, 91.0)	(87.5, 86.9)	(92.1, 92.1)
CICIDS2017	(**99.9**, **99.9**)	(97.1, 97.1)	(97.3, 97.3)	(96.9, 96.9)	(91.8, 91.4)	(97.4, 97.4)
Edge-IIoT	(**100**, **100**)	(**100**, **100**)	(**100**, **100**)	(**100**, **100**)	(**100**, **100**)	(**100**, **100**)
UNSW-NB15	(87.4, 87.1)	(86.4, 86.1)	(82.0, 81.5)	(81.8, 80.9)	(81.8, 80.9)	(**89.2**, **89.1**)
WUSTL-IIoT	(**99.9**, **99.9**)	(**99.9**, **99.9**)	(99.7, 99.7)	(84.3, 84.1)	(99.1, 99.1)	(**99.9**, **99.9**)
RT-IoT	(99.6, 99.5)	(99.2, 99.1)	(99.3, 99.3)	(96.7, 96.3)	(99.2, 99.2)	(**99.7**, **99.6**)
Average	(**97.3**, **97.2**)	(95.6, 95.5)	(95.0, 94.9)	(91.8, 91.5)	(93.2, 92.9)	(96.4, 96.4)

for all datasets. Datasets without predefined splits were divided with a 7:3 ratio, and all experiments were conducted on an NVIDIA RTX 3090 GPU.

2. Experimental results

2.1 Analysis of the lightweight model

Table 2 presents the computational efficiency metrics. While our proposed framework exhibits higher FLOPs due to spatial feature extraction, it achieves the lowest number of parameters at 9,662 and the smallest model size at 38.3KB. Notably, our model is approximately 9 times more compact than LightGBM (343.7KB) and orders of magnitude smaller than TabTransformer (5,684.6KB), highlighting its superior storage efficiency for resource-constrained edge devices.

Table 3 details latency and peak memory usage on a CPU. The total end-to-end latency is 2.60ms, comprising 2.13ms for LVFP preprocessing and 0.47ms for inference. This throughput is significantly faster than complex deep learning baselines such as TabNet (11.56ms) and TabTransformer (15.89ms), ensuring suitability for real-time aggregation intervals. Furthermore, our model demonstrates the highest memory efficiency, requiring only 946.96MB of peak memory. This is comparable to LightGBM (947.11MB) but approximately 45% lower than deep learning baselines like CNN and LSTM, which exceed 1,685MB.

Table 2. Comparison of computational efficiency

Model	FLOPs	Params.	Model Size(KB)
LightGBM	-	-	343.7
CNN	341,476	13,020	49.7
LSTM	76,560	36,578	120.5
TabNet	60,500	71,467	279.2
TabTrans	260,648	1,455,002	5,684.6
Ours	368,402	9,662	38.3

Table 3. Comparison of Latency and Peak Memory Usage on CPU

Model	Preprocessing Latency (ms)	CPU Inference Latency (ms)	Peak Memory (MB)
LightGBM	0.41	0.66	947.11
CNN	1.46	0.16	1685.4
LSTM	0.87	0.25	1686.98
TabNet	7.96	3.60	2005.44
TabTrans	8.82	7.07	1871.06
Ours	2.13	0.47	946.96

2.2 Feature Categorization to RGB Channels

Table 3 presents the number of features eliminated during feature selection, followed by the distribution of features allocated to the R, G, and B channels, and the total count of utilized features across the 6 datasets upon applying the LVFP method. This semantic partitioning enables the encoder to process semantically coherent information via distinct functional channels, facilitating structured feature interaction and reflecting the unique feature composition of each dataset.

Table 4. Distribution of features

Dataset	drop	R	G	B	tot
CICIoT2023	3	12	19	12	43
CICIDS2017	4	23	14	36	73
Edge-IIoTset	3	7	32	4	43
UNSW-NB15	3	11	8	19	38
WUSTL-IIoT	3	17	6	15	38
RT-IoT	5	26	15	36	77

Table 5. Comparison with existing models on 6 datasets for multi-class classification, measured by (accuracy, weighted f1-score). The best performance is shown in **bold** and the second-best is underlined.

Dataset	LightGBM[34]	CNN[35]	LSTM[36]	TabNet[19]	TabTrans[20]	**Ours**
CICIoT2023	(**92.4**, **92.9**)	(75.8, 74.8)	(76.8, 75.8)	(74.6, 73.8)	(73.7, 73.1)	(76.2, 75.3)
CICIDS2017	(**99.6**, **99.6**)	(96.6, 96.5)	(96.5, 96.4)	(96.7, 96.6)	(98.5, 98.4)	(96.8, 96.7)
Edge-IIoT	(89.6, 90.7)	(94.9, 94.5)	(95.0, 94.8)	(93.6, 93.1)	(86.9, 84.3)	(**95.1**, **94.9**)
UNSW-NB15	(71.8, **75.5**)	(**74.5**, 74.7)	(68.2, 69.2)	(73.0, 73.9)	(67.3, 69.3)	(73.9, 74.8)
WUSTL-IIoT	(**99.9**, **99.9**)	(99.8, 99.8)	(99.7, 99.7)	(59.1, 55.1)	(98.7, 98.7)	(**99.9**, **99.9**)
RT-IoT	(**99.8**, **99.8**)	(98.1, 98.1)	(99.0, 99.0)	(90.0, 87.5)	(99.1, 99.1)	(99.5, 99.5)
Average	(**92.2**, **93.1**)	(90.0, 89.7)	(89.2, 89.2)	(81.2, 80.0)	(87.4, 87.2)	(90.2, 90.2)

2.3 Comparison with baseline models

As presented in Tables 1 and 5, our model consistently outperforms the deep learning-based methods in binary and multi-class scenarios, respectively. It achieves average gains of 0.8% in both accuracy and weighted f1-score for binary classification, and 0.2% and 0.1% improvements for multi-class tasks. Notably, while LightGBM performs well on datasets with consistent distributions, our model outperforms it by 1.8% on UNSW-NB15, a dataset known for high discrepancies between training and testing sets. This demonstrates superior generalization capabilities against unknown attack patterns (detailed in sec 2.5). Although the absolute gains are modest, they are achieved with fewer parameters, demonstrating that the LVFP-based pre-training balances detection performance and parameter efficiency for resource-constrained IoT environments.

2.4 Confusion Matrix

Figs. 2 and 3 present the confusion matrices of our model on CICIDS2017 and CICIoT2023, respectively. Fig. 2 shows that the model successfully identifies near-perfect PortScan detection and perfectly classifies the rare Heartbleed attack despite its scarcity, confirming that our approach remains robust for attacks with distinct structural features regardless of sample size. However, it misclassifies 426 DoS instances as Benign due to feature overlap with normal traffic patterns. Additionally, WebAttack exhibits notable confusion with BruteForce, with 184 instances misclassified, likely due to insufficient training

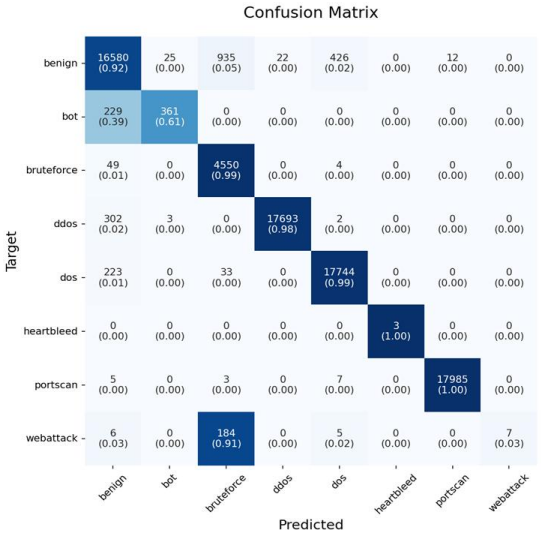

Fig. 2. Confusion Matrix for CICIDS2017

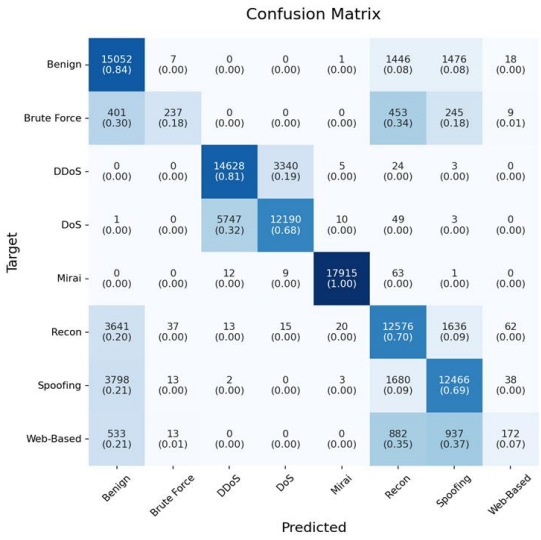

Fig. 3. Confusion Matrix for CICIOT2023

samples and overlapping HTTP-level request patterns. Fig. 3 demonstrates near-perfect detection of Mirai attacks, validating strong performance on volumetric IoT threats, while revealing confusion between DoS and DDoS, with

32% of DoS samples misclassified as DDoS. Nonetheless, most minority attack classes fail to be correctly identified, suggesting that class imbalance and attack signature similarity remain challenging for distinction in the learned feature space.

2.5 Generalization to Unknown Attack Types

To evaluate practical robustness against novel attack patterns, we compare zero-shot detection capability with LightGBM by withholding specific attack classes from training
and evaluating on these unseen types. As shown in Figure 4, while both models achieve comparable accuracy on known attacks, our model consistently outperforms LightGBM on unknown attacks across four datasets. Notably, on CICIoT2023, our model achieves 95.4% accuracy on unseen attack types compared to LightGBM's 88.1%, demonstrating that contrastive pre-training enables generalization beyond the training distribution. Complete results are provided in Appendix B.

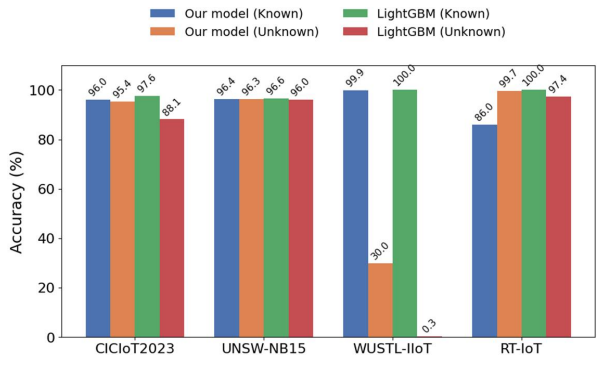

Fig. 4. Intrusion detection performance on known and unknown attacks

Table 6. Ablation Study on Tabular-to-Image Conversion

Method	Binary		Multi-class	
	acc	f1	acc	f1
w/o RGB	90.6	90.0	84.0	82.0
w/o VFP	96.3	96.2	90.1	90.1
w/o Correlation	92.3	92.4	80.6	79.2
w/o Vortex	91.5	91.4	79.0	76.0
w/o nn interpolate	90.3	90.3	81.5	78.5
Ours	**96.4**	**96.4**	**90.2**	**90.2**

3. Ablation Study

This experiment analyzes the impact of each component within the LVFP strategy. As shown in Table 5, aggregating features into a single grayscale map without RGB categorization significantly degrades performance to 90.6% binary accuracy, and replacing VFP with simple row-wise reshaping reduces accuracy to 96.3%. Further ablations on VFP's internal mechanisms show that removing correlation-based ranking reduces binary accuracy performance to 92.3%, excluding the vortex arrangement leads to a decrease to 91.5%, and eliminating nearest-neighbor interpolation further degrades accuracy to 90.3%. These results confirm that LVFP contribute to effective feature representation for CNN-based intrusion detection. Feature categorization and VFP are critical for capturing meaningful relationships in IDS data.

V. Applicability and Limitations

The proposed framework is particularly applicable to heterogeneous IoT networks where diverse traffic protocols coexist, as LLM-guided categorization enables seamless integration of datasets with varying feature schemas without manual re-engineering. Moreover, with the growing prevalence of encrypted traffic (HTTPS/TLS), payload-based intrusion detection faces inherent limitations. Our model overcomes this by relying solely on flow-level statistical features—such as packet rates, inter-arrival times, byte distributions, and protocol behaviors—which remain observable under encryption, enabling effective intrusion detection without decryption.

While our model demonstrates superior robustness against unknown attack patterns compared to LightGBM, its overall detection performance still falls slightly short of LightGBM. Furthermore, the model incurs additional latency due to the tabular-to-image conversion and requires further validation on real-world noisy traffic and edge hardware.

VI. Conclusion and Future Work

In this paper, we propose a novel lightweight pre-training framework for network intrusion detection optimized for resource-constrained IoT environments. By integrating SHAP-based feature selection with the LLM-categorized Vortex Feature Positioning (LVFP) strategy, we implement tabular-to-image conversion, enabling robust contrastive pre-training. Comprehensive experiments across 6 benchmark datasets demonstrate that our model outperforms baselines in both binary and multi-class scenarios. Notably, these performance gains are achieved with a significantly reduced parameter footprint, which validates the practicality of our approach for edge-based deployment.

Future work will address these challenges by implementing the model on real-world edge devices, applying data augmentation techniques such as SMOTE or adversarial sample generation to improve minority class detection, and evaluating robustness against zero-day attacks and adversarial evasion in operational environments.

Appendices

A. Detailed Feature Categorization

A.1 LLM-guided Feature Categorization Prompt

```
SYSTEM:
You are an IDS/IoT Feature Engineering Expert.
We are building a large-scale Foundation Model that requires MAXIMUM INFORMATION.
Your goal is to assign EVERY SINGLE FEATURE from the dataset into R, G, B domains.

CORE INSTRUCTIONS:
NO DISCARD POLICY (Absolute Rule):
You must NOT discard any features.
Even if a feature seems like noise, or redundancy, it MUST be included in the output.

The output feature count must exactly match the input feature count (minus strictly overlapping one-hots if the raw column is preserved but err on the side of keeping them if unsure).

RGB Classification Logic (Priority):

R - Physical Magnitude:
Includes quantitative features representing the physical scale of the traffic, such as bytes, packets, duration, load, and size-related metrics.

G - Intrinsic Semantics:
Includes features that define the identity and structural semantics of the connection, such as protocols, services, flags, IPs, ports, and other structural attributes. Note: Raw categorical columns (e.g., service) should remain in this group without transformation.

B - Behavioral Dynamics:
Includes features that capture behavioral patterns and temporal variability, such as rates, counts, jitter, variance, errors, and time-interval-based metrics.
USER: I will provide the full feature list from XIIoTID. Generate the Python dictionary mapping ALL features to R, G, or B.

Output Format:

{DATASET_NAME}: FeatureGroup = {
    "R": [ ... ], # Include Physical Volume + Randomly assigned leftovers
    "G": [ ... ], # Include Identity/Structure + Randomly assigned leftovers
    "B": [ ... ]  # Include Dynamics/Behavior + Randomly assigned leftovers
}

Feature List:
{feature_names}
```

A.2 Reproducibility Configuration

To ensure deterministic and reproducible categorization across all datasets, the following configuration was strictly enforced. We utilize Gemini 3.0 Pro, the latest stable version available at the time of experimentation. The temperature parameter is set to 0.0 to completely eliminate stochastic token sampling, ensuring that the model produces identical outputs for identical inputs across multiple runs. JSON mode is strictly enforced to guarantee structured output validation and prevent any extraneous text generation.

Following LLM invocation, a rigorous post-processing protocol is applied to ensure integrity and consistency. 1) The raw LLM output is parsed into a Python dictionary object using json.loads(). 2) feature count validation is performed to confirm that len(R) + len(G) + len(B) == total_features, ensuring no features were inadvertently dropped or duplicated. 3) duplicate detection is conducted across all three channels to enforce mutual exclusivity, verifying that no feature appeared in multiple color channels simultaneously. Finally, this configuration is frozen within the model codebase and used consistently across all training and evaluation experiments,

eliminating any possibility of runtime variation in feature categorization.

A.3 LLM-guided Feature Categorization Results

Table 7. LLM-guided Feature Categorization Results

Dataset	Channel	Features
CICIDS 2017	R	min_seg_size_forward, Flow_Duration, Fwd_Packet_Length_Max, Bwd_Packet_Length_Min, Fwd_Header_Length, Bwd_Packet_Length_Mean, act_data_pkt_fwd, Average_Packet_Size, Fwd_Packet_Length_Mean, Max_Packet_Length, Total_Length_of_Bwd_Packets, Total_Length_of_Fwd_Packets, Bwd_Header_Length, Min_Packet_Length, Bwd_Packet_Length_Max, Packet_Length_Mean, Total_Fwd_Packets, Total_Backward_Packets, Fwd_Packet_Length_Min, Subflow_Bwd_Bytes, Avg_Fwd_Segment_Size, Avg_Bwd_Segment_Size, Fwd_Header_Length.1
	G	Init_Win_bytes_forward, Init_Win_bytes_backward, PSH_Flag_Count, ACK_Flag_Count, URG_Flag_Count, FIN_Flag_Count, Fwd_PSH_Flags, Bwd_URG_Flags, Bwd_PSH_Flags, SYN_Flag_Count, ECE_Flag_Count, CWE_Flag_Count, RST_Flag_Count, Fwd_URG_Flags
	B	Bwd_Packets/s, Flow_IAT_Min, Fwd_IAT_Min, Flow_Bytes/s, Flow_IAT_Mean, Fwd_IAT_Std, Bwd_Packet_Length_Std, Fwd_IAT_Mean, Flow_IAT_Std, Fwd_Packets/s, Flow_IAT_Max, Fwd_IAT_Total, Flow_Packets/s, Fwd_IAT_Max, Packet_Length_Std, Bwd_IAT_Std, Idle_Min, Down/Up_Ratio, Bwd_IAT_Max, Bwd_IAT_Min, Fwd_Packet_Length_Std, Bwd_IAT_Total, Idle_Mean, Bwd_IAT_Mean, Active_Min, Active_Mean, Idle_Std, Idle_Max, Active_Max, Active_Std, Packet_Length_Variance, Fwd_Avg_Bytes/Bulk, Bwd_Avg_Bulk_Rate, Bwd_Avg_Packets/Bulk, Bwd_Avg_Bytes/Bulk, Fwd_Avg_Bulk_Rate
CICIOT 2023	R	number, header_length, weight, flow_duration, magnitue, min, tot_size, duration, max, avg, tot_sum, radius
	G	protocol_type, https, tcp, ack_flag_number, udp, syn_flag_number, http, psh_flag_number, ssh, rst_flag_number, icmp, fin_flag_number, ipv, dns, arp, cwr_flag_number, ece_flag_number, irc, dhcp
	B	iat, rst_count, urg_count, variance, rate, syn_count, covariance, fin_count, ack_count, std, srate, drate
EdgeIoT	R	dns.qry.name.len, tcp.len, http.content_length, arp.hw.size, mqtt.len, mqtt.proto_len, mqtt.topic_len
	G	http.request.method, http.referer, tcp.ack, tcp.flags, tcp.seq, tcp.ack_raw, http.request.version, udp.stream, icmp.checksum, tcp.checksum, tcp.connection.rst, arp.opcode, icmp.seq_le, mqtt.topic, mqtt.protoname, mqtt.conack.flags, tcp.connection.syn, tcp.connection.fin, tcp.connection.synack, dns.qry.name, tcp.flags.ack, http.response, dns.qry.qu, icmp.unused, http.tls_port, dns.qry.type, mqtt.hdrflags, mqtt.conflag.cleansess, mqtt.conflags, mqtt.msgtype, mqtt.msg_decoded_as, mqtt.ver
	B	dns.retransmission, udp.time_delta, dns.retransmit_request, dns.retransmit_request_in
RTIoT	R	fwd_pkts_payload.min, flow_pkts_payload.avg, fwd_header_size_min, fwd_pkts_payload.max, fwd_pkts_payload.tot, fwd_header_size_tot, fwd_subflow_bytes, flow_pkts_payload.min, bwd_pkts_payload.max, fwd_pkts_payload.avg, flow_pkts_payload.max, flow_pkts_payload.tot, bwd_header_size_tot, bwd_pkts_payload.avg, bwd_subflow_bytes, bwd_pkts_payload.min, bwd_pkts_tot, bwd_pkts_payload.tot, fwd_header_size_max, flow_duration, bwd_header_size_min, fwd_pkts_tot, fwd_data_pkts_tot, fwd_subflow_pkts, bwd_data_pkts_tot, bwd_subflow_pkts

	G	id.resp_p, id.orig_p, fwd_last_window_size, service, fwd_URG_flag_count, proto, flow_FIN_flag_count, flow_SYN_flag_count, fwd_init_window_size, flow_ACK_flag_count, bwd_init_window_size, fwd_PSH_flag_count, flow_RST_flag_count, bwd_PSH_flag_count, flow_CWR_flag_count
	B	bwd_pkts_per_sec, flow_iat.min, flow_pkts_payload.std, flow_pkts_per_sec, payload_bytes_per_second, active.min, fwd_pkts_per_sec, fwd_iat.min, flow_iat.std, fwd_iat.avg, bwd_iat.min, fwd_iat.tot, bwd_pkts_payload.std, flow_iat.avg, fwd_iat.std, fwd_iat.max, bwd_iat.avg, active.max, active.avg, flow_iat.max, active.tot, fwd_pkts_payload.std, down_up_ratio, bwd_iat.max, bwd_iat.std, flow_iat.tot, idle.min, bwd_iat.tot, idle.avg, idle.max, idle.tot, active.std, idle.std, bwd_bulk_packets, fwd_bulk_rate, fwd_bulk_packets
UNSW-NB15	R	sbytes, smean, dbytes, dmean, dload, sload, trans_depth, dur, response_body_len, dpkts, spkts
	G	sttl, proto, dtcpb, stcpb, dttl, state, is_ftp_login, swin
	B	ct_dst_sport_ltm, ct_srv_dst, ct_srv_src, ct_dst_src_ltm, sloss, ct_src_ltm, ct_src_dport_ltm, dloss, synack, ct_dst_ltm, djit, ct_state_ttl, sinpkt, tcprtt, sjit, ackdat, dinpkt, rate, ct_flw_http_mthd
WUSTL IIOT	R	DstPkts, SrcBytes, DstLoad, DAppBytes, TotAppByte, TotBytes, SrcPkts, Load, Dur, SrcLoad, TotPkts, SAppBytes, Sum, Mean, DstBytes, Max, Min
	G	sTtl, sIpId, Proto, dTtl, dIpId, sDSb
	B	IdleTime, DIntPkt, SIntPkt, pLoss, SrcLoss, DstRate, TcpRtt, SynAck, Loss, Rate, SrcRate, SrcJitter, SrcJitAct, DstLoss, DstJitter

B. Details on Generalization to Unknown Attack Types

B.1 Additional Results on CICIDS2017 & EdgeIIoT

Table 8 details the accuracy metrics for CICIDS2017 and Edge-IIoT. On CICIDS2017, our model achieves 0.6% higher accuracy on unknown attacks compared to LightGBM, while maintaining comparable performance on known attacks. This result further supports the observation that our model generalizes better to previously unseen attack types under zero-shot settings.

B.2 Split of Known and Unknown Attack Types

To evaluate the zero-shot detection capability described in Section 2.5, we split the attack categories of each dataset into disjoint Known (seen during training) and Unknown (unseen during training) sets. This split allows us to assess the model's ability to generalize to novel threat patterns that were not part of the training distribution. The detailed breakdown of attack categories for all datasets is provided in Table 9.

Table 8. Zero-shot intrusion detection accuracy on CICIDS2017 and Edge-IIoT datasets (%)

Dataset	Model	Known	Unknown
CICIDS2017	Ours	99.5	62.4
	LightGBM	100	61.8
Edge-IIoT	Ours	100	100
	LightGBM	100	100

Table. 9 Split of Known and Unknown Attack Categories for Zero-Shot Evaluation

Dataset	Attack Type	Attack Categories
CICIDS2017	Known	portscan, dos, bruteforce
	Unknown	ddos, bot, heartbleed, webattack
CICIOT2023	Known	spoofing, bruteforce, ddos
	Unknown	dos, recon, web-based, mirai
UNSW-NB15	Known	dos, generic, exploits, fuzzers
	Unknown	analysis, reconnaissance, worms, backdoor, shellcode
RT-IoT	Known	ddos_slowloris, arp_poisoning, metasploit_brute_force_ssh, nmap_os_detection
	Unknown	nmap_xmas_tree_scan, nmap_fin_scan, nmap_tcp_scan, nmap_udp_scan, dos_syn_hping
WUSTL-IIoT	Known	dos, backdoor
	Unknown	comminj, reconn

Edge-IoT	Known	port_scanning, ddos_icmp, ddos_tcp, mitm, password, xss
	Unknown	ddos_udp, fingerprinting, ransomware, sql_injection, uploading, vulnerability_scanner, backdoor

ACKNOWLEDGEMENT

This research was supported by the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (No. P0023675, HRD Program for Industrial Innovation).

REFERENCES

[1] Hyung-Woo Lee, "A method for collecting traces of intrusion incidents based on IoT botnet malware," Journal of Internet of Things and Convergence, vol. 11, no. 1, pp. 1-8, 2025. DOI: 10.20465/KIOTS.2021.7.3.001

[2] Tinshu Sasi et al., "A comprehensive survey on IoT attacks: Taxonomy, detection mechanisms and challenges," Journal of Information and Intelligence, vol. 2, no. 6, pp. 455-513, 2024. DOI: 10.1016/j.jiixd.2023.12.001

[3] Brunel Rolack Kikissagbe and Meddi Adda, "Machine learning-based intrusion detection methods in IoT systems: A comprehensive review," Electronics, vol. 13, no. 18, Art. no. 3601, 2024. DOI: 10.3390/electronics13183601

[4] M. Marwah and M. Arlitt, "Deep Learning for Network Traffic Data," in Proc. of the 28th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (KDD 2022), Washington, DC, USA, pp. 4804-4805, 2022. DOI: 10.1145/3534678.3542618

[5] Mohanad Sarhan et al., "Feature extraction for machine learning-based intrusion detection in IoT networks," Digital Communications and Networks, vol. 10, no. 1, pp. 205-216, 2024. DOI: 10.1016/j.dcan.2022.08.012

[6] Chinnasamy, Ramya, et al. "Deep learning-driven methods for network-based intrusion detection systems: A systematic review." ICT Express 2025. DOI: 10.1016/j.icte.2025.01.005

[7] Mohammadpour, Leila, et al. "A survey of CNN-based network intrusion detection." Applied Sciences 12.16:8162 2022. DOI: 10.3390/app12168162

[8] Li, Zeyu, and Wenbin Yao. "A two stage lightweight approach for intrusion detection in Internet of Things." Expert Systems with Applications 257: 124965 2024. DOI: 10.1016/j.eswa.2024.124965

[9] Alok Sharma et al., "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," Scientific Reports, vol. 9, no. 1, Art. no. 11399, 2019. DOI: 10.1038/s41598-019-47765-6

[10] Yitan Zhu et al., "Converting tabular data into images for deep learning with convolutional neural networks," Scientific Reports, vol. 11, no. 1, Art. no. 11325, 2021. DOI: 10.1038/s41598-021-93376-5

[11] Jong-Ik Park et al., "Vortex Feature Positioning: Bridging tabular IIoT data and image-based deep learning," Internet of Things, vol. 31, Art. no. 101533, 2025. DOI: 10.1016/j.iot.2025.101533

[12] Tommaso Zoppi, Stefano Gazzini, and Andrea Ceccarelli, "Anomaly-based error and intrusion detection in tabular data: No DNN outperforms tree-based classifiers," Future Generation Computer Systems, vol. 160, pp. 951-965, 2024. DOI: 10.1016/j.future.2024.06.051

[13] Elvin Li et al., "SAFE: Self-Supervised Anomaly Detection Framework for Intrusion Detection," arXiv preprint, 2025. DOI: 10.48550/arXiv.2502.07119

[14] Giuseppina Andresini et al., "GAN augmentation to deal with imbalance in imaging-based intrusion detection," Future Generation Computer Systems, vol. 123, pp. 108-127, 2021. DOI: 10.1016/j.future.2021.04.017

[15] T. Tulasi Bhavani, M. Kameswara Rao, and A. Manohar Reddy, "Network intrusion detection system using random forest and decision tree machine learning techniques," International Conference on Sustainable Technologies for Computational Intelligence (ICTSCI 2019), pp. 108-116, Singapore, 2019. DOI: 10.1007/978-981-15-0029-9_50

[16] Iksu Shin et al., "A practical feature extraction study to improve accuracy and responsiveness of machine learning-based IDS security event classification," Journal of the Korea Institute of Information Security and Cryptology, vol. 28, no. 2, pp. 385-395, 2018. DOI: 10.13089/JKIISC.2018.28.2.385

[17] Richard Kimanzi et al., "Deep learning algorithms used in intrusion detection systems: A review," arXiv preprint, 2024. DOI: 10.48550/arXiv.2402.17020

[18] Mohamed Mahmoud et al., "Ae-LSTM: Autoencoder with LSTM-based intrusion detection in IoT," Proceedings of the International Telecommunications Conference (ITC-Egypt 2022), Cairo, Egypt, 2022. DOI: 10.1109/ITC-Egypt55520.2022.9855688

[19] Sercan Ö. Arik and Tomas Pfister, "TabNet: Attentive interpretable tabular learning," AAAI Conference on Artificial Intelligence, vol. 35, no. 8, 2021. DOI: 10.1609/aaai.v35i8.16826

[20] Xin Huang et al., "TabTransformer: Tabular data modeling using contextual embeddings," arXiv preprint, 2020. DOI: 10.48550/arXiv.2012.06678

[21] Tasnimul Hasan et al., "Enhanced intrusion detection in IIoT networks: A lightweight approach with autoencoder-based feature learning," arXiv preprint, 2025. DOI: 10.48550/arXiv.2501.15266

[22] Abdullah I. A. Alzahrani et al., "Anomaly detection in fog computing architectures using custom tab transformer for Internet of Things," Electronics, vol. 11, no. 23, Art. no. 4017, 2022. DOI: 10.3390/electronics11234017

[23] Garcia Molina, Pablo, et al. “Foundation Models for Cybersecurity: A Comprehensive Multi-Modal Evaluation of TabPFN and TabICL for Tabular Intrusion Detection.” Electronics 14.19, 2025. DOI: 10.3390/electronics14193792

[24] Ruiz-Villafranca, Sergio, et al. “A TabPFN-based intrusion detection system for the industrial internet of things.” The Journal of Supercomputing 80.14, 20080-20117, 2024. DOI: 10.1007/s11227-024-06166-x

[25] Zhou H., Zou H., Li W., Li D., and Kuang Y., “HiViT-IDS: An Efficient Network Intrusion Detection Method Based on Vision Transformer,” Sensors, vol. 25, no. 6, p. 1752, 2025. DOI: 10.3390/s25061752

[26] A. Orman, “Cyberattack detection systems in industrial Internet of Things (IIoT) networks in big data environments,” Applied Sciences, vol. 15, p. 3121, 2025. DOI: 10.3390/app15063121

[27] Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." International conference on machine learning. PmLR, pp. 1597-1607, 2020. DOI: 10.5555/3524938.3525087

[28] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," International Conference on Information Systems Security and Privacy (ICISSP 2018), pp. 108–116, Funchal, Portugal, 2018. DOI: 10.5220/0006639801080116

[29] Euclides Carlos Pinto Neto et al., "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," Sensors, vol. 23, no. 13, Art. no. 5941, 2023. DOI: 10.3390/s23135941

[30] Mohamed Amine Ferrag et al., "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," IEEE Access, vol. 10, pp. 40281–40306, 2022. DOI: 10.1109/ACCESS.2022.3165809

[31] J. B. Capital, "Real-Time Internet of Things (RT-IoT2022)," Dataset, 2022.

[32] M. Zolanvari et al., "WUSTL-IIoT-2021: A dataset for IIoT cybersecurity research," Washington University in St. Louis, USA, 2021.

[33] Nour Moustafa and Jill Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," Military Communications and Information Systems Conference (MilCIS 2015), Canberra, Australia, 2015. DOI: 10.1109/MilCIS.2015.7348942

[34] Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in neural information processing systems 30, 2017. DOI: 10.5555/3294996.3295074

[35] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11, 2278-2324, 1998. DOI: 10.1109/5.726791

[36] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8: 1735-1780. 1997. DOI: 10.1162/neco.1997.9.8.1735

[37] Halbouni, Asmaa, et al. "CNN-LSTM: hybrid deep neural network for network intrusion detection system." IEEE Access 10, 99837-99849, 2022. DOI: 10.1109/ACCESS.2022.3206425

[38] Abdel-Rahman, Mohamed A., et al. "Feature importance guided autoencoder for dimensionality reduction in intrusion detection systems." Scientific Reports 16.1: 5013, 2026. DOI: 10.1038/s41598-026-36695-9

[39] B. Susilo, A. Muis, and R. F. Sari, “Intelligent intrusion detection system against various attacks based on a hybrid deep learning algorithm,” Sensors, vol. 25, no. 2, p. 580, Jan. 2025. DOI: 10.3390/s25020580

Authors

Jun Yeong Park received the B.S. degree in Urban Engineering from Hongik University, korea, in 2024. He is currently pursuing the M.S. degree in Digital Analytics at Yonsei University, Seoul, Korea.

His research interests focus on anomaly detection, MLLM, Continual Learning.

Kunwoo Kang received the B.A degree in Economics from Yonsei University, Korea, in 2025. He is currently pursuing the M.S. degree in Digital Analytics at Yonsei University, Seoul, Korea.

His research interests focus on time-series modeling.

Hoin Lee received the B.S. degree in Applied Statsitcs from The Pennsylvania State University, USA, in 2024. He is currently pursuing the M.S. degree in Digital Analytics at Yonsei University, Seoul, Korea.

His research interests focus on multi-agent systems and LLM-based agents for collaborative statistical data analysis.

Seungeun Lee received the B.A. degree in Library and Information Science from Chung-Ang University, Korea, in 2024. She is currently pursuing the M.S. degree in Digital Analytics at Yonsei University, Seoul, Korea.

Her research interests focus on Recommender Systems.

Yu Rang Park received the M.S. and Ph.D. degrees in Molecular and Genomic Medicine from Seoul National University, Korea, in 2006 and 2012. She is now a professor at the Department of Biomedical Systems

Informatics & Digital Analytics at Yonsei University, Seoul, Korea. Her research interests focus on Medical informatics, Standardization, Distributed computing.