IDNet: Instance-adaptive Dynamic Network with Adversarial Training for Intrusion Detection

Tianjing Wang[a], Qi Liu[a], Hang Shen[a,*] and Guangwei Bai[a]

[a]College of Computer and Information Engineering (College of Artificial Intelligence), Nanjing Tech University, 30 South Puzhu Road, Nanjing 211816, China

ARTICLE INFO

Keywords:
Dynamic networks
intrusion detection
adversarial training
curriculum learning
traffic instance

ABSTRACT

Cyberattacks continue to pose significant threats to interconnected systems. While deep neural networks offer high accuracy for intrusion detection, their high computational cost often leads to latency and missed detections on resource-constrained edge devices. To address this challenge, we present IDNet, an Instance-adaptive Dynamic Network enhanced with adversarial training to jointly improve detection accuracy and inference efficiency. IDNet incorporates a semantic-aware traffic instantiation module that converts heterogeneous network flows into unified grayscale matrices, enabling consistent processing of traffic data with different feature dimensionalities and extraction methodologies. Based on this unified representation, adversarial training is used to stabilize feature extraction under instance-adaptive dynamic inference. A generator is constrained by structural similarity to ensure that synthetic samples preserve label fidelity and traffic-specific characteristics. The discriminator consists of a residual detection network and a lightweight policy network; the latter generates routing vectors to dynamically activate relevant paths on the former based on traffic instance complexity. We introduce a self-optimized pre-training strategy that combines adversarial and curriculum learning to initialize the detection and policy networks. An augmentation-enhanced joint fine-tuning strategy is further applied to align detection behavior with learned block-discarding patterns, with improved generalization on underrepresented attack classes. Extensive experiments across multiple authoritative, heterogeneous datasets show that IDNet significantly reduces inference costs compared to full-scale residual networks and conventional dynamic neural networks, while maintaining competitive detection accuracy against known and unknown attacks. Evaluation on resource-limited platforms further demonstrates IDNet's practical efficiency, outperforming all baseline methods in inference speed.

1. Introduction

The rapid growth of the Internet and its interconnected ecosystems, including the Internet of Things (IoT), has transcended time and location constraints to create a global network linking humans, machines, and devices. However, this proliferation has also heightened the risk of cyberattacks, such as ransomware, data breaches, and hacking, which pose significant threats to critical information infrastructure, national security, and economic stability [1]. The fast-paced iteration of smart devices has further reduced attack costs and increased stealth, making modern attacks more sophisticated and elusive. A network intrusion detection system (NIDS) [2] is a real-time security technology that monitors network transmissions and issues alerts or initiates countermeasures upon detecting suspicious activity. Such systems demand high responsiveness and computational efficiency. However, NIDS are increasingly deployed in computing- and memory-constrained edge environments, making the deployment of complex deep models challenging. For instance, representative edge platforms, such as Raspberry Pi-class devices, typically provide only GFLOP-level computational capability, whereas server-grade accelerators, such as RTX 4090-class GPUs, operate at the TFLOP scale. Moreover, edge networks often experience bursty, high-density traffic arrivals, requiring a large number of flows to be processed within short time intervals. This mismatch poses a fundamental challenge to the timely and reliable detection of intrusions in interconnected IoT networks.

Conventional machine learning techniques (such as naive Bayes [3], random forests [4], and support vector machines [5]) often fail in large-scale network traffic with high dimensionality and complexity. Deep learning models with multilayer nonlinear architectures can extract abstract representations from raw traffic data, thereby significantly improving detection accuracy. Among these models, long short-term memory (LSTM)-based models effectively capture complex temporal dependencies. Enhanced variants like bidirectional LSTM (BiLSTM) [6], Res-TranBiLSTM [7], and multi-head attention-enhanced LSTM [8] further improve feature extraction across time steps. However, LSTM-based models face challenges with traffic variability, and performance gains plateau as model depth increases [9]. Additionally, conventional traffic representation methods relying on raw packet-level features struggle when packet lengths vary, as longer packets may obscure critical information from others, preventing the model from learning a complete and balanced representation of traffic behavior [10].

*This work was supported in part by the National Natural Science Foundation of China under Grants 61502230 and 61501224, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20201357, in part by the State Administration for Market Regulation Science and Technology Program under Grant 2025MK183, and in part by the Market Supervision Administration Science and Technology Fund of Jiangsu Province under Grant KJ2026027.

*Corresponding author
hshen@njtech.edu.cn (H. Shen)
ORCID(s): 0000-0002-8804-2787 (H. Shen)

As traditional one-dimensional representations remain limited, researchers have explored representing network traffic as two-dimensional matrices [10]. In the physical world, image instances convey rich visual information such as background, texture, and contours [11]. In cyberspace, traffic data can be abstracted into multi-dimensional features and temporal patterns, enabling a visual representation of network behavior. However, different attacks exhibit unique flow characteristics distributed across traffic instances, resulting in spatial patterns with varying levels of recognition difficulty. Existing image detection methods rely on static deep learning architectures to extract and classify spatial features [12, 13], which cannot adapt to diverse attack instances. Although resource-limited platforms may theoretically support commonly static models (e.g., ResNet-101 requires approximately 7.88 GFLOPs), practical deployment often suffers from system load and memory latency, leading to unstable real-time inference. To alleviate the bottleneck, some approaches apply model compression techniques such as pruning [14] or quantization [15]. Nevertheless, these methods often incur accuracy loss [16], limiting their ability to balance efficiency and detection reliability in constrained IDS scenarios.

1.1. Challenges and Related Works

Dynamic networks [17] offer a promising solution for balancing inference cost and detection accuracy. Unlike static models with fixed computation graphs, dynamic networks adapt their execution paths via selecting specific layers, blocks, or exits based on input complexity [18]. This enables more efficient resource usage and faster inference without sacrificing performance. However, the complex and diverse nature of traffic instances introduces several unique challenges for dynamic network-based intrusion detection:

1) Traffic-instance-adaptive inference. The goal is to minimize inference time while preserving accuracy through instance-adaptive computation. Prior research has primarily focused on vision tasks, such as object detection [19] and image classification [20]. Methods like BranchyNet [21] and ACT [22] use early-exit strategies, terminating inference once confidence thresholds are met. While effective for simple inputs, this approach struggles with covert or low-frequency attacks, where shallow features lack sufficient discriminative power. Alternatives, such as ConvNet-AIG [23], SkipNet [24], and MA-DyNN [25], utilize gating mechanisms to selectively execute network blocks. However, non-differentiable gates complicate optimization. ConvNet-AIG uses Gumbel-Softmax [26], while SkipNet combines supervised and reinforcement learning (RL) to improve stability. DDI [27] further enhances SkipNet by introducing regularization to reduce reliance on RL. Despite these advances, most approaches follow sequential decision-making, which can be inefficient on resource-constrained devices. BlockDrop [28] generates routing paths in one step, reducing latency. However, it depends on priors from pre-trained image models, which are unavailable in traffic data because of heterogeneous protocols and behavioral patterns.

2) Cooperative pre-training of policy and detection networks. A typical dynamic network consists of a detection and a policy network. The former provides fundamental detection functionality, while the latter can dynamically determine the inference depth [28] or width [29] of the detection network based on instance features. Many models for image instance detection (e.g., ResNet [30], GoogleNet [31], and VGG [32]) can be used to guide the pre-training of policy networks. However, these pre-trained models cannot be directly applied to traffic detection. More critically, no pre-trained models are available for traffic instances. Consequently, for traffic detection, the policy network cannot rely on a high-initialization detection network for pre-training. Given the extensive search space of policy networks, random initialization makes it difficult to optimize detection capability and dynamic inference simultaneously. This necessitates exploring a self-optimizing method to support pre-training without a high baseline detection network.

3) Enhancing generalization of dynamic inference. Most dynamic networks are trained on class-balanced datasets such as ImageNet[1] and CIFAR-100[2], enabling policy networks to learn routing strategies that adapt to diverse instances. Models like DMNN [33] and IAMNN [34] achieve dynamic path adaptation through gating mechanisms and adaptive computation time. In the context of intrusion detection, traffic data is typically highly imbalanced, with attack traffic representing only a small fraction. This imbalance causes dynamic networks to overfit to dominant benign patterns, leading to suboptimal routing behavior and reduced detection performance for minority attack classes. To alleviate this issue, generative adversarial network (GAN)-based data augmentation has been explored to balance training distributions [35]. However, conventional GANs struggle to model multi-class imbalances effectively and often lack semantic awareness of traffic-specific patterns. Recent advances, such as the BERT-enhanced conditional GAN (CGAN) [36], split-federated-BERT with adversarial training [37], improve data balance and feature richness by integrating pretrained contextual embeddings. TMG-GAN [38] further introduces a multi-generator framework with cosine similarity constraints to generate diverse attack samples, but its architectural complexity hinders deployment in resource-constrained environments. Auxiliary classifier GANs (AC-GANs) [39] generate class-conditional samples more efficiently, but rely on label perturbation without enforcing traffic-domain constraints. This limits the realism and diversity of the generated attacks, hindering the generalization capability of routing policies in dynamic networks.

1.2. Main Contributions

To address these challenges, we propose IDNet, an Instance-adaptive Dynamic Network framework designed for resource-efficient edge intrusion detection. IDNet incorporates a semantic-aware traffic instantiation for unified data representation. Based on this semantic mapping, the

[1] https://image-net.org/

[2] https://www.cs.toronto.edu/ kriz/cifar.html

framework leverages adversarial learning to jointly optimize feature extraction and dynamic inference routing, balancing detection accuracy with computational efficiency. The primary contributions are summarized as follows:

- An adversarial framework tailored for adaptive detection across diverse traffic instances. Its generator employs a structural similarity constraint, ensuring synthesized samples maintain label consistency and behavioral fidelity. The discriminator comprises a residual detection network and a lightweight policy network, which generates routing policy vectors to guide the detection network's inference path.
- A self-optimizing pre-training strategy based on curriculum learning (CL), jointly initializing the detection and policy networks. The curriculum is implemented across network depth and sample complexity through phased scheduling and adaptive sample weighting. This strategy mitigates policy initialization difficulties arising from an expansive search space and addresses the absence of transferable pre-trained models in intrusion detection.
- A joint fine-tuning approach enhanced by auxiliary data augmentation. Synthetic traffic instances enhance the detection network's ability to identify rare or underrepresented traffic types, thereby strengthening the policy network's robustness and generalization.

Comprehensive experiments on multiple datasets demonstrate IDNet's robustness, efficiency, and superior performance. To guide the experimental evaluation, this work aims to answer the following three research questions.

- **RQ1**: How does traffic instance diversity impact the behavior and effectiveness of dynamic inference (Answered in Subsection 5.1)?
- **RQ2**: To what extent can IDNet accelerate inference on resource-constrained environments (Answered in Subsection 5.2)?
- **RQ3**: Can IDNet robustly generalize to diverse traffic structures and unseen attack patterns (Answered in Subsection 5.3)?

The remainder of this paper is organized as follows. Section 2 introduces the IDNet architecture. The training methodology of IDNet is described in Section 3. Section 4 outlines the experimental setup and evaluation metrics. Comprehensive performance analyses are reported in Section 5. Finally, Section 6 concludes the paper and discusses potential directions for future work.

2. IDNet Architecture

Consider an end-edge-cloud-collaborative intrusion detection scenario, as illustrated in Fig. 1 (a). A NIDS operates at the network edge, while a cloud platform is responsible for data storage and model training, including a generator (G) and a discriminator (D). The trained D is then deployed on edge nodes to perform dynamic inference on IoT traffic and generate risk assessments. Edge nodes continuously upload local traffic data to the cloud, enabling iterative updates of the global model parameters. The architecture of the proposed model is shown in Fig. 1 (b). The choice of a two-dimensional semantic layout combined with a residual detection network is motivated by the requirements of instance-adaptive dynamic inference. Successive residual blocks extract features at progressively higher semantic granularities, and their skip connections allow individual blocks to be bypassed without disrupting information flow, providing the structural basis for block-level dynamic routing. This hierarchical architecture enables the policy network to associate routing decisions with specific levels of semantic analysis, making the trade-off between computational cost and detection depth both controllable and interpretable.

Each network flow is converted via semantic mapping into a real instance x with dimensions 8×8 and a single channel. Random noise z and corresponding class labels y are input into G to synthesize instance samples $\tilde{x}_y = G(z, y)$. To ensure the generated samples are realistic, intra-class and inter-class similarity constraints are imposed on G outputs. Both real instances x and synthetic instances $\tilde{x}_y$ are then fed into D, where a lightweight policy network produces binary routing decisions, determining which layers in the residual detection network are skipped or executed. The D's authenticity loss for sample discrimination is expressed as

$$\mathcal{L}_v = \mathbb{E}\left[\log P(V = \text{real}|\text{x})\right] + \mathbb{E}\left[\log P(V = \text{fake}|\tilde{\text{x}})\right] \tag{1}$$

and that for sample classification loss is expressed as

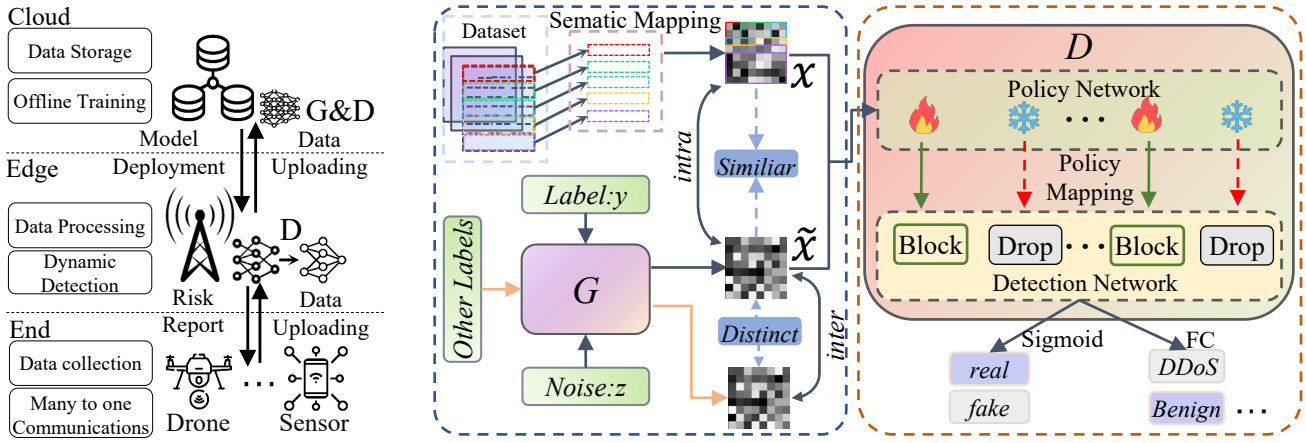
$$\mathcal{L}_y = \mathbb{E}\left[\log P(Y = y|x_y)\right] + \mathbb{E}\left[\log P(Y = y|\tilde{x}_y)\right]. \tag{2}$$

During training, G maximizes $\mathcal{L}_y - \mathcal{L}_v$ to generate realistic instances, while D maximizes $\mathcal{L}_y + \mathcal{L}_v$ to accurately differentiate real from synthetic instances and correctly classify them. This adversarial dynamic inference training enables IDNet to effectively distinguish diverse traffic instances while balancing inference efficiency and detection accuracy.

2.1. Semantic-aware traffic instantiation

Network intrusion detection datasets exhibit significant heterogeneity in feature representations due to the adoption of diverse feature extraction tools and statistical methodologies. NetFlow exporters and CICFlowMeter are two widely used tools that parse raw network packets into flow-level statistical features and output them as one-dimensional vectors. For instance, NF-UNSW-NB15-v2[3], which adheres to NetFlow standards, incorporates 43 protocol-oriented features,

[3]https://espace.library.uq.edu.au/view/UQ:ffbb0c1



(a) Working scene.

(b) IDNet architecture.

Figure 1: Overview of the working scene and IDNet architecture. The overall pipeline includes upstream traffic feature extraction, semantic-aware traffic instantiation, and instance-adaptive dynamic inference for resource-efficient intrusion detection, with adversarial sample augmentation used during training to improve robustness on underrepresented attacks.

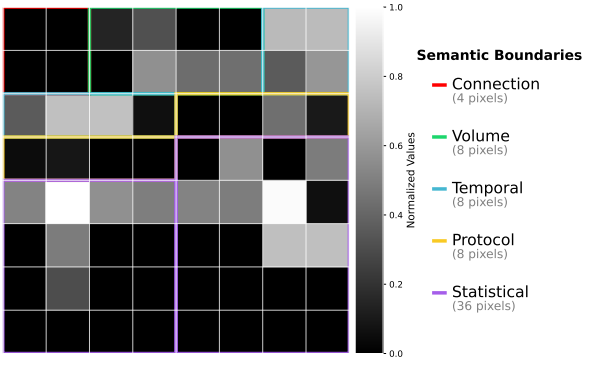
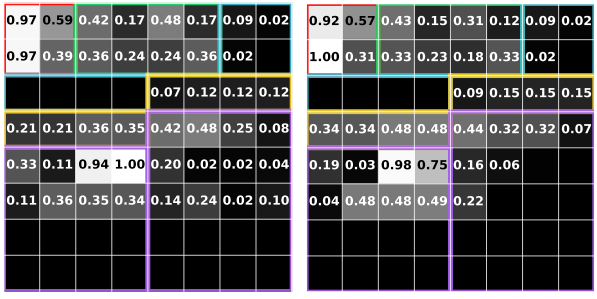
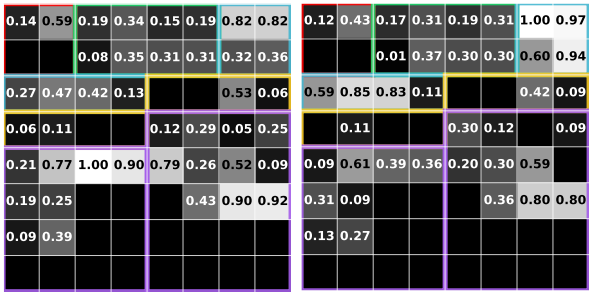


Figure 2: Semantic-aware traffic instantiation rule.



(a) Normal (left) and Reconnaissance (right) on NF-UNSW.



(b) Normal (left) and Reconnaissance (right) on CIC-UNSW.

Figure 3: Traffic instantiation examples.

whereas CIC-UNSW-NB15[4], leveraging the CICFlowMeter framework, encompasses 75 statistically-oriented features. Despite variations in dimensionality and nomenclature, these datasets fundamentally characterize identical behavioral dimensions: connection patterns, data transmission, temporal dynamics, protocol interactions, and statistical distributions, as illustrated in Fig. 2. Given the inherent semantic correlations and hierarchical relationships among network traffic features, traditional one-dimensional vectorized representations struggle to capture both intra-category dependencies and cross-semantic patterns. To address this, we propose a semantically aware traffic instantiation method that transforms traffic features into structured two-dimensional representations while preserving semantic locality. Unlike upstream feature extraction tools that produce only raw one-dimensional vectors, our method provides a unified spatial encoding through semantic grouping, deterministic spatial allocation, and adaptive feature aggregation.

[4]https://www.unb.ca/cic/datasets/cic-unsw-nb15.html

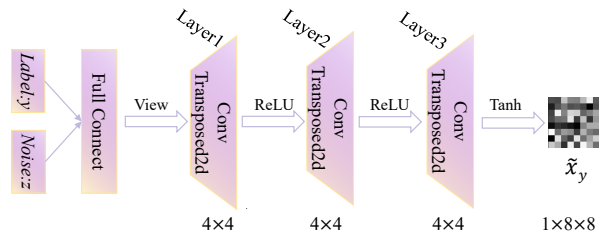
Consider the typical feature dimensionality across datasets (ranging from 40 to 80 features) and the need for computational efficiency. An 8×8 representation (64 pixels) emerges as an effective choice: it accommodates the common feature range without excessive sparsity or information loss through aggressive aggregation. This compact square format aligns well with standard convolutional kernels, enabling efficient feature extraction while maintaining sufficient resolution to preserve traffic characteristics. Additionally, we adopt an adaptive semantic aggregation strategy to reconcile the dimensional mismatch between variable-length feature vectors. For semantically dense categories,

multiple related features are aggregated via weighted averaging; for sparse ones, statistical imputation or duplication is applied to ensure complete coverage. Crucially, our mapping enforces semantic positional consistency, assigning functionally equivalent features from different datasets to the same pixel locations, regardless of their original names or formats. This unified layout preserves both interpretability and architectural consistency across datasets. For instance, "IN_BYTES"/"OUT_BYTES" from the NF series and "Total Length of Fwd/Bwd Packets" from the CIC series both describe volume semantics and are thus mapped to the same region within the image. As shown in Fig. 3, this semantic-aware mapping yields visually consistent spatial structures across structurally distinct datasets. The complete feature-to-pixel mapping specifications for each dataset family are detailed in Section 4.

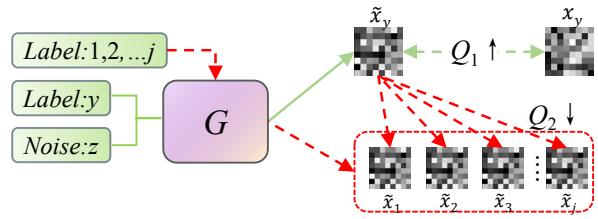
The spatial locality in our representation differs from that in natural images, where adjacent pixels are correlated because they capture a continuous physical scene. Network traffic features do not possess inherent spatial structure. Instead, the spatial locality is deliberately constructed through semantically aware grouping, where features that characterize the same behavioral dimension are mapped to contiguous pixel regions. A convolutional kernel sliding over such a region captures co-occurrence patterns among functionally related features. For instance, the combination of high byte volume, short flow duration, and specific TCP flag configurations within adjacent pixels may jointly indicate a volumetric attack. In one-dimensional representations, such semantically related features may be dispersed at distant positions, making it difficult for a single kernel to capture these combinatorial patterns.

2.2. Generator with structural similarity

Adversarial training refers to *G-D* optimization paradigm rather than adversarial evasion attacks against the detector. G synthesizes class-conditional traffic instances as continuous-valued grayscale matrices to augment underrepresented attack categories, while the adversarial process regularizes the detection network toward more generalizable feature representations. As shown in Fig. 4(a), G consists of a five-layer fully connected network. Initially, the embedding layer transforms a class label y and random noise z into a 100-dimensional vector, which is multiplied element-wise by a 100-dimensional normal random vector. This combined vector passes through a fully connected network consisting of one input layer, three hidden layers (each with 100 neurons), and an output layer (64 neurons). After ReLU activation, the resulting vector is further transformed by three transposed convolutional layers and a final Tanh activation, which produces a synthesized instance $\tilde{x}_y$. Although ACGAN employs labels for conditional generation [40], it only supervises generation via label classification accuracy, lacking explicit constraints on behavioral authenticity. In network traffic generation, this limitation may yield label-consistent but behaviorally unrealistic samples, thereby reducing both the quality of the generation and its practical utility.



(a) Generator structure.



(b) Structural similarity constraint.

Figure 4: Generator under structural similarity constraint.

A structural similarity constraint is incorporated to enhance the interpretability and realism of generated samples at the distribution level, as detailed in Fig. 4(b). Consider generated and real traffic samples of class y as $\tilde{x}_y$ and x_y, with corresponding mean brightness (μ_f, μ_r), standard deviations (σ_f^2, σ_r^2), and covariance (σ_{fr}). The structural similarity is quantified as

$$\mathrm{S2}(\tilde{x}_y, x_y) = \frac{(2\mu_f\mu_r + a_1)(2\sigma_{fr} + a_2)}{(\mu_f^2 + \mu_r^2 + a_1)(\sigma_f^2 + \sigma_r^2 + a_2)} \tag{3}$$

where a_1 and a_2 stabilize the division, preventing zero denominators. S2 values range from $[-1, 1]$, with 1 indicating identical images and 0 signifying no similarity.

Denote y' as any class distinct from y, with $\tilde{x}_{y'} = G(y', z)$. The inter-class and intra-class similarity, Q_1 and Q_2, are quantified as

$$\begin{cases} Q_1 = \mathrm{S2}(\tilde{x}_y, x_y) \\ Q_2 = \dfrac{1}{C-1} \displaystyle\sum_{y' \in \{1,2,\ldots,C\} \setminus \{y\}} \mathrm{S2}(\tilde{x}_y, \tilde{x}_{y'}). \end{cases} \tag{4}$$

The goal is to maximize Q_1 while minimizing Q_2. Accordingly, the training objective for G, incorporating the losses from (1) and (2), is represented as

$$\mathcal{L}_G = \mathcal{L}_y - \mathcal{L}_v + Q_1 - Q_2. \tag{5}$$

A well-trained G generates synthetic attack instances to balance the dataset and enhance D's generalization to rare threats.

2.3. Discriminator with dynamic reasoning

D comprises ResNets with varying depths for constructing policy and detection networks. Given inputs x and $\tilde{x}$, the policy network determines binary routing probabilities

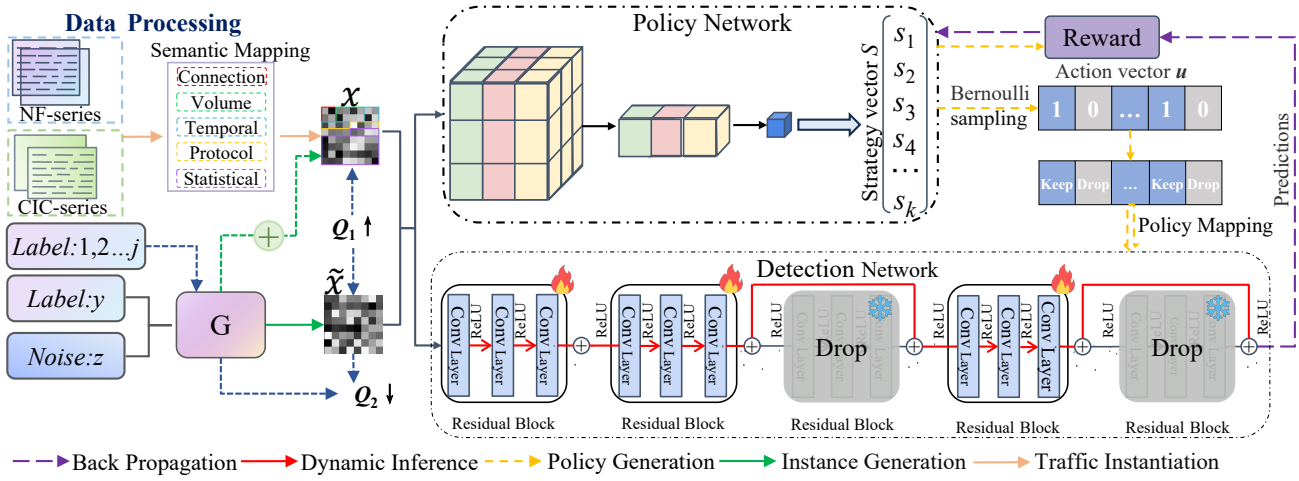


Figure 5: Joint training and fine-tuning workflow of IDNet.

for each residual block in the detection network. Predictions are then computed only through the active residual blocks. The detection accuracy and block usage rate inform a reward function that updates the policy network.

Fig. 5 illustrates the workflow. A traffic instance x is processed by the policy network, parameterized by $\mathcal{W}$, and activated by the Sigmoid function $\sigma(x) = 1/(1+e^{-x})$, yielding a probability vector

$$S = \phi_1(x; \mathcal{W}). \tag{6}$$

If the detection network contains K residual blocks, $s_k \in S$ indicates the drop probability of the kth block. Routing decisions $u \in \{0,1\}^K$ are made simultaneously via single-step prediction and follow a Bernoulli distribution

$$\pi_{\mathcal{W}}(u \mid x) = \prod_{k=1}^{K} s_k^{u_k}(1-s_k)^{1-u_k} \tag{7}$$

where $u_k = 1$ and $u_k = 0$ represent the decisions to drop and keep. Let C denote the total number of classes. For multi-class classification with true labels y_c and predicted probabilities $\hat{y}_c$, the cross-entropy loss is

$$\mathcal{L} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c). \tag{8}$$

Normalized prediction loss is given by

$$p = \frac{1}{1+\exp(-\mathcal{L})}. \tag{9}$$

The utilization rate of the residual blocks is calculated as $(\sum_{k=1}^{K} u_k)/K$. The reward function balances prediction accuracy and inference cost, designed as

$$\mathcal{R}(p,u) = \begin{cases} (1-p)\cdot\left[1-\left(\dfrac{\sum_{k=1}^{K} u_k}{K}\right)^2\right], & \text{if correct} \\ -p\cdot\gamma, & \text{otherwise} \end{cases} \tag{10}$$

where γ is the penalty for incorrect predictions. Unlike fixed-reward approaches, IDNet integrates normalized prediction loss to dynamically adjust incentives: lower loss values yield higher rewards, promoting minimal resource use, while higher losses impose larger penalties, ensuring robustness and flexibility.

Normally, the more residual blocks are selected, the higher the detection accuracy, but this comes at the expense of reduced inference speed. To achieve the optimal routing with minimal cost, we maximize the expected return of the policy network as

$$\mathcal{J} = \mathbb{E}_{u\sim\pi_{\mathcal{W}}}\left(\mathcal{R}(p,u)\right). \tag{11}$$

To optimize parameters, the policy gradient method [41] computes gradients based on a Bernoulli distribution rather than the traditional multinomial distribution commonly used in RL [42]. The gradient for policy network parameters $\mathcal{W}$ is calculated as

$$\begin{aligned} \Delta_{\mathcal{W}}\mathcal{J} &= \mathbb{E}\left[\mathcal{R}(p,u)\nabla_{\mathcal{W}}\log\pi_{\mathcal{W}}(u/x)\right] \\ &= \mathbb{E}\left[\mathcal{R}(p,u)\nabla_{\mathcal{W}}\log\prod_{k=1}^{K} s_k^{u_k}(1-s_k)^{1-u_k}\right] \\ &= \mathbb{E}\left[\mathcal{R}(p,u)\nabla_{\mathcal{W}}\sum_{k=1}^{K}\log[s_k u_k + (1-s_k)(1-u_k)]\right]. \end{aligned} \tag{12}$$

Due to the high variance of policy gradients, a self-critical baseline $\mathcal{R}(p,\hat{u})$, the most probable routing under the current policy, is introduced to reduce variance [43]. Routing decision $\hat{u}$ is defined by $u_k = 1$ if $0.5 < s_k < 1$, otherwise $u_k = 0$. With decision bias $\mathcal{A} = \mathcal{R}(p,u) - \mathcal{R}(p,\hat{u})$, the adjusted gradient expression becomes

$$\Delta_{\mathcal{W}}\mathcal{J} = \mathbb{E}\left[\mathcal{A}\cdot\nabla_{\mathcal{W}}\sum_{k=1}^{K}\log[s_k u_k + (1-s_k)(1-u_k)]\right], \tag{13}$$

providing a convergence guarantee while maintaining lower variance than standard policy gradients.

3. Joint IDNet Pre-Training and Fine-Tuning

During training, G and D (which includes the dynamic network) are updated in an alternating manner. When one component is trained, the other is frozen. This adversarial training cycle continues until convergence. The overall workflow is illustrated in Fig. 5, in which the key steps are as follows:

① **Traffic Instantiation:** Heterogeneous datasets (e.g., NF-series and CIC-series) are transformed into unified 8×8 semantic representations through the proposed semantic mapping approach, where features are systematically allocated to designated pixel regions based on their semantic categories.

② **Instance Generation (Augmentation):** G generates fake traffic instances to enhance the detection network's robustness via adversarial training. Once G is sufficiently trained, its parameters are fixed and used to augment training data by generating additional synthetic traffic instances (as shown by the green dashed).

③ **Policy Generation (Routing Decision):** The policy network generates binary routing vectors based on input instances. These vectors are sampled from a Bernoulli distribution to determine which blocks to activate.

④ **Dynamic Inference:** The detection network performs instance-adaptive inference by executing only the selected residual blocks, reducing computational cost.

⑤ **Back Propagation (Adversarial Training):** The detection network updates its parameters based on classification loss. Predictions and routing vectors are fed into a reward function and the resulting reward is backpropagated to optimize the policy network. These classification outcomes also guide further updates to D, facilitating improved traffic instance generation.

3.1. Detection-policy network pre-training

In complex, heterogeneous network environments, effective initialization of the detection and policy networks is crucial for learning robust, adaptive routing decisions. A detection network of K residual blocks yields 2^K potential routing paths, resulting in an exponentially ample policy search space. Traditional random search strategies often lack synergy with feature representation learning, leading to suboptimal accuracy and inefficient block usage [28]. Moreover, unlike computer vision tasks, where large-scale pretrained models are available, intrusion detection lacks transferable pretrained models. Consequently, untrained detection networks may fail to provide meaningful feedback to the policy network, limiting its generalization and decision-making capabilities.

To address these challenges, we develop a CL-based pre-training framework that introduces a staged scheduling mechanism for the collaborative initialization of detection and policy networks. CL progressively guides the model from simple to complex samples. While prior methods primarily focus on sample-wise curriculum progression, our approach also incorporates architectural progression from shallow to deep structures. Specifically, a scheduling variable h controls the number of residual blocks involved in optimization during training, with a growth rate governed by T. As the training epoch, e, increases, the scheduling strategy follows

$$h = \min\left\{1 + \left\lfloor \frac{e}{T} \right\rfloor, K\right\}. \tag{14}$$

At each stage, the policy network keeps the first $K - h$ blocks active and learns binary activation policies for the remaining h blocks. This enables the model to begin training with lower-complexity subnetworks and gradually adapt to the whole network.

To regulate the progression of training samples, we introduce two curriculum regularization principles: (i) sample diversity should increase over time; and (ii) the training weight for each sample should be non-decreasing. Let $\lambda = e/M \in [0, 1]$ represent normalized training progress, where e is the current epoch and M is the total number of epochs. For a sample z, let $W_\lambda(z)$ be its training weight at step λ, with $W_1(z) = 1$. Based on (14), the weight function is defined as

$$W_\lambda(z) = (1 - \lambda)\exp\left(-\frac{|a(z) - (K - h)|}{K}\right) + \lambda \tag{15}$$

where $a(z)$ is the number of residual blocks selected by the policy network for z. The deviation, $|a(z) - (K - h)|$, reflects how well the model's perceived complexity of a sample aligns with the current stage. This adaptive weighting encourages the model to focus on samples that match its representational capacity, stabilizing training and providing more effective gradient signals. This design also enhances training effectiveness by guiding the model to learn from samples of appropriate complexity, thereby promoting stable and progressive capability development.

Let $P(z)$ denote the prediction distribution for sample z. The training distribution at curriculum step λ is then

$$Q_\lambda(z) = \frac{W_\lambda(z)P(z)}{\int W_\lambda(z')P(z')} \tag{16}$$

which ensures that $\int Q_\lambda(z) = 1$ and $Q_1(z) = P(z)$. The curriculum design allows the model to prioritize simple samples (requiring few active blocks) and progressively transition to learning from complex ones. This facilitates a smooth capability expansion and effective policy learning.

The entropy loss of the decision for the residual block is represented as $H = -\sum_k^n s_k \log(s_k)$. Based on (13), we introduce entropy regularization as

$$I = \frac{1}{N}(\Delta_{\mathcal{W}}\mathcal{J} - \alpha H) \tag{17}$$

where α denotes the weight to control the relative contribution of the entropy regularization and N represents the

number of samples in a batch. The mean value ensures that I is not affected by the batch size. The purpose of (17) is to enhance policy exploration and encourage the generation of a diverse action probability distribution.

The policy network's training requires feedback from the detection network. Due to the lack of a pre-trained ResNet dedicated to intrusion detection, using only policy loss as the training loss ignores detection accuracy, leading to routing with low detection precision. Accordingly, an adversarial learning loss is introduced into D's optimization objective to balance inference cost and accuracy during pre-training. The objective is formulated as maximizing

$$\mathcal{L}_D = \mathcal{L}_y + \mathcal{L}_v + I. \tag{18}$$

Fake and real instances are input into D. According to (18), the policy and detection networks in D are updated by Adam [44]. Subsequently, D is frozen for updates. According to (5), NAdam [45] is used to update G. This alternating process is repeated M_1 times until G and D converge.

3.2. Detection-policy network fine-tuning

The policy network in D is first self-optimized through CL-based pre-training to learn diverse routing strategies. Although this phase aims to preserve accuracy, discarding residual blocks during inference may introduce inconsistencies between training and deployment. Therefore, joint fine-tuning must align the detection network with the learned routing behaviors. During this phase, the policy network transitions from CL-guided progressive block control to unrestricted RL, allowing both networks to adapt simultaneously to dynamic routing strategies. Under the adversarial training, the competition between D and G resumes for M_2 iterations. Throughout training, G periodically saves its parameters to generate synthetic samples for augmenting underrepresented attack classes. Training proceeds until convergence. The optimization objective for D during joint fine-tuning is represented as

$$\mathcal{L}_D = \mathcal{L}_y + \mathcal{L}_v + \Delta_{\mathcal{W}}\mathcal{J} \tag{19}$$

which integrates classification loss, authenticity loss, and policy-gradient rewards to promote accurate classification.

Algorithm 1 outlines the execution process of pre-training and joint fine-tuning. Initially, the weights of G and D are set to random values. The numbers of epochs for the pre-training and joint fine-tuning phases are set to M_1 and M_2. The penalty factor is set to γ and the probability bounding factor to α. During training, random noise z and labels y are introduced to generate image instances $\tilde{x}_y$ through G. The policy network generates a policy. CL and adversarial training are incorporated in the pre-training phase to self-optimize the policy and detection networks. Subsequently, the reward is evaluated by combining class and real-fake probabilities, and the policy gradient is calculated. Finally, the detection and policy networks are jointly fine-tuned to enhance detection accuracy and reduce inference costs.

Pre-training and fine-tuning involve gradient updates for G and D, with D comprising the policy and detection networks. The total time complexity can be approximated as $O(\mu+\delta+\varepsilon)$, where μ, δ, and ε denote the computational volume of G, the policy network, and the full detection network. During pre-training, blocks are activated based on a scheduling mechanism, resulting in a time complexity of $O(m \cdot M_1 \cdot (\mu + \delta + \varepsilon \cdot (K - h)/K))$, where m denotes the batch size. In contrast, the joint fine-tuning phase utilizes routing paths dynamically generated by the policy network, resulting in a time complexity of $O(m \cdot M_2 \cdot (\mu + \delta + \varepsilon \cdot \sum_{k=1}^{K} u_k/K))$.

Algorithm 1: Joint Pre-training and Fine-tuning Algorithm

Input: Original traffic instance set $\mathcal{X}_{\text{real}}$, labels y;
Output: Policy s, class probability p_y and real-fake probability p_v;

Initialize the weights of G and D, which includes the policy network ϕ_1 and the detection network ϕ_2, to random values;
Set epochs for pre-training and joint fine-tuning to M_1 and M_2; and set penalty factor γ and probability bounding factor α;
for $t \leftarrow 1$ *to* M_1 **do**
  $\tilde{x}_y \leftarrow G(z, y)$;
  Calculate Q_1 and Q_2 based on (3) and (4);
  $s \leftarrow \phi_1(x; \mathcal{W})$, $x \in \mathcal{X}_{\text{real}} \cup \mathcal{X}_{\text{fake}}$;
  Determine scheduling strategy h based on (14);
  if $h < K$ **then**
    set $s[1 : K - h] \leftarrow 1$;
    Calculate $W_\lambda(z)$ based on (15);
  $u \sim \text{Bernoulli}(s)$;
  $\{p_y, p_v\} \leftarrow \phi_2(x, u)$;
  Freeze D, update G using NAdam based on (5);
  $R \leftarrow \mathcal{R}(p, u)$ in which p is detection loss;
  Calculate policy gradient $\mathcal{J}$ based on R;
  Freeze G, update ϕ_1 and ϕ_2 in D using Adam;
for $t \leftarrow 1$ *to* M_2 **do**
  Jointly fine-tune the detection and policy networks based on (19);

4. Experimental Preparation

We evaluate the proposed method on four publicly available intrusion detection datasets: three from the NF series and one from the CIC series, covering a wide range of network environments and attack scenarios. Notably, CIC-UNSW-NB15 (CIC-UNSW) serves as the CICFlowMeter-based counterpart to NF-UNSW-NB15-v2 (NF-UNSW), sharing identical traffic categories but differing in feature extraction methodology, thus enabling a direct evaluation of the proposed semantic mapping's structural transferability. Among the NF datasets, NF-BoT-IoT-v2 (NF-BoT) is heavily imbalanced with over 99% malicious traffic; NF-ToN-IoT-v2 (NF-ToN) offers a more balanced mix of benign and multiple attack types; and NF-UNSW is predominantly

Table 1
Semantic-aware mapping specification for CIC-series and NF-series features

Semantic	CIC-series			NF-series		
	Representative Features	#Feat./Pix.	Agg.	Representative Features	#Feat./Pix.	Agg.
Connection	Subflow fwd/bwd packets & bytes	4 / 4	Direct	IP addresses & ports	4 / 4	Direct
Volume	Packet counts & byte totals	11 / 8	Mean	In/outbound bytes and packets	8 / 8	Direct
Temporal	Flow duration & IAT statistics	15 / 8	Mean	Flow duration & directional timing	3 / 8	Direct
Protocol	TCP flags, header length, init window	16 / 8	Mean	TCP flags, TTL, window size	9 / 8	Mean
Statistical	Length variance, flow rate, active/idle	29 / 36	Direct	Throughput & packet size	21 / 36	Direct

benign, with 96% of its traffic being non-malicious. Together, these datasets provide a comprehensive benchmark for assessing the robustness of the proposed NIDS. For all experiments, 70% of each dataset is used for training and the remaining 30% for testing.

Tables 1 provide the concrete feature-to-pixel mapping specifications for CIC-series and NF-series datasets, respectively. Each table lists the five semantic groups, representative features assigned to each group, the ratio of features to allocated pixels (#Feat./Pix.), and the aggregation strategy (Agg.). When the number of features exceeds the allocated pixels, weighted mean aggregation is applied; otherwise, direct mapping is used. Despite the significant difference in feature dimensionality (75 vs. 43 features), both dataset families are mapped to the same spatial layout, with semantically equivalent features occupying identical pixel regions.

4.1. Baseline methods

Selection criteria focused on maintaining comparable model capacities and similar inference paradigms. The baselines include both established methods from the literature and custom-designed variants with varying parameter scales. This setup allows us to disentangle performance gains from architectural innovations versus those from increased model complexity, thereby highlighting the specific advantages of our approach. Eight representative baseline methods were selected for comprehensive comparison:

- **BlockDrop** [28]: A dynamic inference method originally developed for image classification to minimize computational cost. Following the official implementation, we adopt it to the traffic instance format without additional task-specific fine-tuning.
- **Full ResNet** [30]: A static model variant that disables the policy network, executing inference across the entire detection network without dynamic routing.
- **LSTM** [46]: A lightweight sequential model composed of six LSTM layers with 39, 20, 60, 80, and 90 neurons in the hidden layers. The output layer is dataset-specific, matching the number of classes in each dataset (5 for NF-BoT, and 10 for both NF-ToN and NF-UNSW). The model contains 21K parameters.
- **LSTM-210k**: A medium-scale LSTM model with eight layers, configured with neuron counts of 39, 256, 256, 128, 128, 128, 128, and a dataset-specific output layer. Total parameters are approximately 210K.
- **LSTM-240k**: A larger LSTM model comprising ten layers with neuron counts of 39, 256, 256, 256, 128, 128, 128, 128, 128, and a dataset-specific output layer. Total parameters are approximately 240K.
- **KNN** [47]: A classical non-parametric intrusion detection method. The number of neighbors is set to 3.
- **IDL-IDS** [48]: Raw network traffic is transformed into images and applies standard deep learning classifiers (CNN, LSTM, or RNN) for detection.
- **IEDL-IDS** [48]: A state-of-the-art image-based IDS that extends IDL-IDS by introducing a CNN-based encoder to extract latent features from traffic images before classification.

4.2. Evaluation metrics

The following five metrics are employed to quantitatively analyze and evaluate intrusion detection methods. Accuracy intuitively reflects the proportion of correctly detected instances, while precision and recall emphasize the proportion of correctly classified instances. The F1-score is a more comprehensive evaluation metric. FLOPs are used to assess the computational complexity of intrusion detection methods. Convolutional and fully connected layers make the major contributions to inference computation. Assume the number of input and output channels in a convolutional layer is denoted as N_1 and N_2, the kernel size as K_s, and the height and width of the output feature map as H_2 and W_2. For the fully connected layer, the numbers of input and output features are denoted as F_1 and F_2. The FLOPs can be expressed as

$$\begin{aligned} \text{FLOPs} &= \text{FLOPs}_{\text{conv}} + \text{FLOPs}_{\text{fc}} \\ &= 2 \cdot N_1 \cdot N_2 \cdot K_s^2 \cdot H_2 \cdot W_2 + 2 \cdot F_1 \cdot F_2. \end{aligned} \tag{20}$$

4.3. Implementation Details

Experiments were conducted on a workstation with a 14th Gen Intel Core i9-14900K CPU, 128 GB DDR5 RAM, a 2 TB NVMe SSD, and dual NVIDIA RTX 4090 GPUs (24 GB each). The model was implemented using PyTorch and trained with the Adam optimizer. During pre-training, we used a batch size of 2048 and a learning rate of 1×10^{-4}.

In the fine-tuning phase, the batch size was reduced to 1024, and the learning rate was set to 1×10^{-5} to improve stability. The penalty factor γ was fixed at 4, and the probability bounding factor α in Eq. (17) was set to 0.8.

From the implementation perspective, the framework consists of semantic-aware traffic instantiation and instance-adaptive dynamic inference. Raw traffic features extracted by different frameworks are first normalized and reorganized into one-dimensional vectors, and then mapped to 8×8 grayscale matrices according to the proposed semantic-aware rule. This step provides a unified semantic-spatial encoding for heterogeneous traffic data, enabling convolutional filters to capture local co-occurrence patterns among functionally related attributes. Based on this representation, dynamic routing is realized through a lightweight policy network and a residual detection network. For each input instance, the policy network outputs a routing vector that determines whether each residual block is executed or skipped, enabling different traffic instances to be processed with varying computational depths. This design is well-suited to IDS deployment, where benign traffic should be processed efficiently, while suspicious or stealthy traffic should trigger deeper inference. During training, semantic representation learning, adversarial augmentation, and dynamic routing are jointly optimized. During deployment, only D is retained for online inference, while G is used only for offline minority-class augmentation, keeping runtime overhead low for resource-constrained IDS scenarios.

5. Experimental results

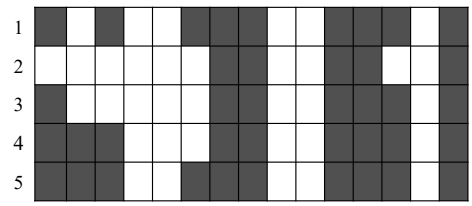
Real-world network traffic exhibits substantial variation in detection difficulty, while edge devices impose strict computational budgets. The following experiments evaluate whether IDNet can exploit instance-level traffic complexity to achieve efficient and accurate intrusion detection, addressing the three research questions (**RQ1**–**RQ3**) raised in Section 1.2. Subsection 5.1 examines whether different attack types require different inference depths, thereby motivating dynamic routing (**RQ1**). Subsection 5.2 measures the resulting computational savings and deployment efficiency under constrained resources (**RQ2**). Subsection 5.3 analyzes whether reduced computation affects detection performance, covering class-imbalanced scenarios, cross-tool feature compatibility, unseen attack types, and per-class behavior on rare attacks (**RQ3**). Subsections 5.4 and 5.5 further evaluate whether adversarial augmentation generates reliable synthetic traffic and improves the detection of under-represented attack classes.

5.1. Reasoning Sensitivity to Instance Discrepancy

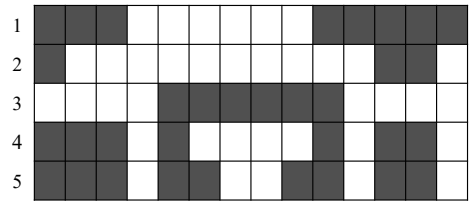
Given that traffic instances exhibit significant differences in brightness, contrast, and structure, we employ $S2(\cdot,\cdot)$ to quantify the similarity between attack instances and benign instances. Table 2 provides statistics on precision, data proportion, similarity, and routing length for the five attack types illustrated in Fig. 2. Theft and DDoS exhibit similar precision; however, Theft exhibits the highest similarity and lowest data proportion, resulting in the longest routing. In contrast, DDoS shows the lowest similarity to Benign and the highest data proportion, resulting in the shortest routing. These results demonstrate an inverse relationship between similarity and routing length, with data proportion as a contributing factor.

Table 2
Routing path correlation

Attack type	Precision	Similarity	Data proportion	Avg. Route length
Benign	98.59%	N/A	0.36%	9
Theft	99.99%	0.0504	0.01%	10
Reconnaissance	94.31%	0.0386	6.94%	7
DoS	98.24%	0.0365	44.15%	6
DDoS	99.40%	0.0306	48.54%	4



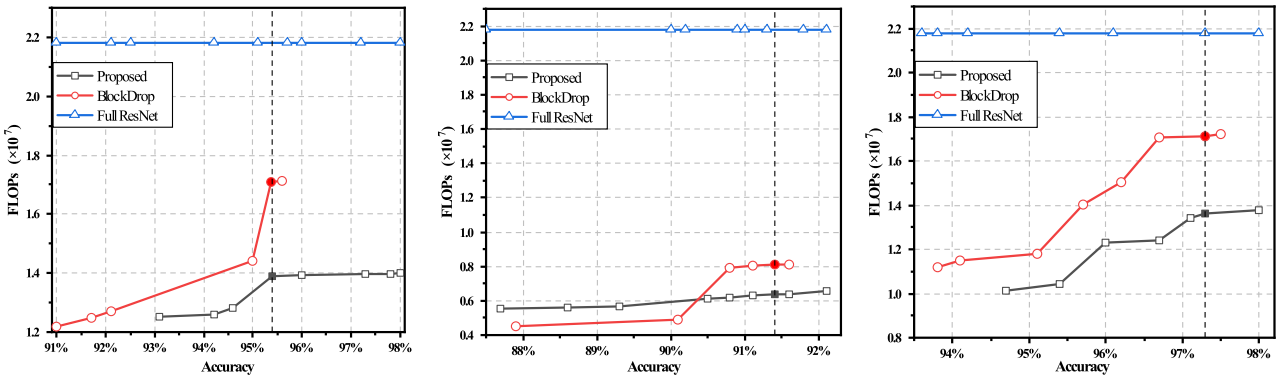
(a) IDNet.



(b) BlockDrop.

Figure 6: Visualization of routing paths for different traffic instances (The first to fifth lines represent the routing distribution of Benign, DDoS, DoS, Reconnaissance, and Theft).

Further analysis reveals differences in routing path distributions across traffic instances. Traffic instances of various types in NF-BoT are input into the policy network, with routing vectors generated by the proposed method and BlockDrop visualized in Fig. 6. The horizontal axis represents residual modules, and the vertical axis represents traffic types; gray (blank) indicates whether the residual blocks participate in the routing. It is observed that the routing strategies for attack types exhibit high similarity, with some similar attack features stored in adjacent filters. Additionally, in a detection network comprising 15 residual blocks, BlockDrop utilizes an average of 7.65 residual blocks, whereas the proposed method uses an average of 6.17. This signifies that the proposed method reduces inference cost by approximately 58.87% compared to the whole network and by about 19.35% compared to BlockDrop.



(a) Accuracy vs. FLOPs on NF-BoT. (b) Accuracy vs. FLOPs on NF-ToN. (c) Accuracy vs. FLOPs on NF-UNSW.

Figure 7: Detection accuracy and FLOPs across different datasets.

Table 3
Comparison of image-based intrusion detection on NF-ToN

Model	Classifier	Accuracy	FLOPs ($\times 10^6$)	Params (K)
IDL-IDS [48]	LSTM	89.79%	17.65	23.18
	CNN	87.20%	13.49	22.67
	RNN	92.48%	21.67	13.87
IEDL-IDS [48]	LSTM	95.51%	38.92	165.41
	CNN	96.42%	36.61	164.99
	RNN	95.99%	36.52	162.66
IDNet	ResNet	**98.02%**	**8.80**	240.00

Table 4
Comparison of LSTM-related intrusion detection on NF-BoT

Model	Accuracy	Params
LSTM	81.55%	21k
LSTM-210k	89.97%	210k
LSTM-240K	89.87%	240k
IDNet	97.98%	240k

5.2. Inference acceleration analysis

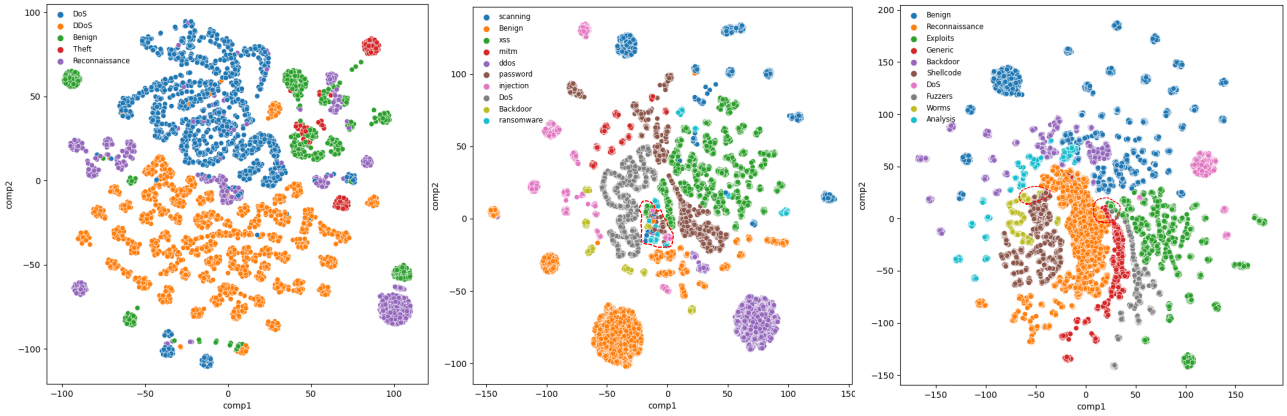
As illustrated in Fig. 7, the proposed solution reduces the computational load for multi-class classification by 18.65% to 21.36% across the three datasets compared to BlockDrop and by a more substantial 36.19% to 70.81% compared to a Full ResNet. These results indicate that the dynamic network flexibly determines dynamic network routing for different traffic instances, resulting in a significant reduction in FLOPs compared to Full ResNet. The FLOPs generated by G in the proposed method are almost negligible. By leveraging adversarial training, the proposed method improves detection accuracy while reducing the number of floating-point operations during inference.

To position IDNet within the broader landscape of image-based intrusion detection methods, we further compare it with IDL-IDS and IEDL-IDS [48], two recent IDS approaches that also transform network traffic into visual representations for deep learning-based classification but rely on static architectures with fixed computational paths. Table 3 reports the comparative results on the NF-ToN dataset. IDL-IDS achieves detection accuracies ranging from 87.20% to 92.48% with computational costs between 13.49×10^6 and 21.67×10^6 FLOPs. IEDL-IDS further improves accuracy to 95.51%–96.42% by employing an encoder-based architecture, but this gain comes at the expense of substantially higher computational complexity, with 36.52×10^6 to 38.92×10^6 FLOPs, and an increased parameter count of 162.66K to 165.41K. In contrast, IDNet achieves 98.02% accuracy while requiring only 8.80×10^6 FLOPs, representing a reduction of approximately 76% compared with IEDL-IDS. These results indicate that instance-adaptive dynamic inference can surpass the detection performance of static encoder-based architectures while maintaining significantly lower computational overhead. Such efficiency is crucial for edge deployment, where IDNet's reduced FLOPs facilitate real-time intrusion detection under constrained computational budgets, demonstrating that dynamic inference is a viable and effective paradigm for resource-constrained network security applications. Beyond image-based approaches, Table 4 compares the accuracy of IDNet- and LSTM-related approaches with varying parameter counts on NF-BoT. Because LSTMs lack convolutional and fully connected layers, their inference cost cannot be measured by FLOPs. Increasing LSTM layers and neurons to match IDNet's parameters does not yield satisfactory detection performance.

To quantitatively analyze inference efficiency under resource-constrained conditions, we used Docker to set up a controlled execution environment with explicitly limited resources (4 CPU cores capped at 1.5 GHz, 4 GB RAM, and 1 Gbps network bandwidth). This configuration provides a fair platform for comparing computational overhead across all methods under equivalent resource constraints. IDNet, BlockDrop, and Full ResNet were deployed in this environment to assess their inference latency when processing network traffic. During inference, all algorithms achieved near-maximum CPU utilization. As shown in Table 5, IDNet consistently achieves lower inference latency and a smaller memory footprint than all baselines, demonstrating superior

Table 5
Inference latency and memory footprint across models on resource-constrained conditions

Algorithm	NF-BoT			NF-ToN			NF-UNSW		
	Latency (ms/sample)	Relative Speedup	RAM Memory	Latency (ms/sample)	Relative Speedup	RAM Memory	Latency (ms/sample)	Relative Speedup	RAM Memory
Full ResNet	31.75	N/A	788 MB	31.75	N/A	933 MB	31.75	N/A	1662 MB
BlockDrop	26.96	15.1% ↑	718 MB	27.03	14.9% ↑	723 MB	15.02	52.7% ↑	1184 MB
IDNet	**23.23**	**26.8% ↑**	**675 MB**	**22.20**	**30.1% ↑**	**688MB**	**12.47**	**60.7% ↑**	**964 MB**



(a) Feature distribution on NF-BoT. (b) Feature distribution on NF-ToN. (c) Feature distribution on NF-UNSW.

Figure 8: Visualization of high-dimensional feature representations.

resource efficiency under constrained computational budgets. Building on the latency and resource usage results, we estimate per-sample energy consumption assuming a peak power draw of approximately 7 W[5] at full CPU load. Under this assumption, IDNet requires about 0.0873 J/sample on NF-UNSW (12.47 ms/sample), yielding a throughput of roughly 80 samples/second. A 25 Wh energy source could thus support approximately 3.6 hours of continuous operation, processing over one million traffic instances.

5.3. Detection performance analysis

5.3.1. Extracted feature distribution visualization

The high-dimensional traffic feature distributions across three datasets were visualized to illustrate how features vary across traffic types. For each dataset, 10,000 samples were randomly selected using a fixed random seed (seed = 42) to facilitate qualitative visualization. The experiment employed uniform manifold approximation and projection (UMAP) [49] to project the high-dimensional features into a two-dimensional space (i.e., components 1 and 2). Figs. 8(a)-(c) show the visualized distributions of traffic instance features. Attack instances are separated from benign instances. One noteworthy observation is that in NF-ToN and NF-UNSW, the high-dimensional features of a few low-frequency and stealthy attack instances are intermixed (marked by red circles). Consequently, the classification effect for these instances is weakened.

[5]https://www.pidramble.com/wiki/benchmarks/power-consumption

5.3.2. Multi-class and binary classification

Table 6 presents the accuracy, precision, recall, and F1-score of different methods on NF-ToN, NF-BoT, and NF-UNSW. As shown in Table 6, both the proposed method and Full ResNet outperform other methods in both binary and multi-class classification metrics. Unlike BlockDrop, the proposed solution improves multi-class accuracy by 2.28%, 0.53%, and 0.51% on these three datasets, respectively. On NF-UNSW, which has the lowest proportion of attack traffic, the weighted F1-score (W-F1) of the proposed solution surpasses BlockDrop, LSTM, and KNN by 0.78%, 5.43%, and 17.73%, respectively. The proposed model exhibits comparable precision and W-F1 to Full ResNet. Traffic instances are high-dimensional data, and the distance between data points in high-dimensional space becomes large, leading to degraded KNN detection performance. LSTM is prone to overfitting when the training data is imbalanced or noisy, leading to reduced generalization on test data. Although BlockDrop adopts a dynamic network architecture, integrating auxiliary modules can further boost performance. Considering Table 6 and Fig. 7, the IDNet enhances the attack identification capability. Additionally, Tables 12, 13, and 14 show the precision, recall, and F1-score for benign and attack instance recognition across the three datasets. Both the proposed solution and Full ResNet achieve nearly 100% F1-score for attack instance detection. On NF-UNSW, where the proportion of benign instances exceeds 96%, the proposed solution and BlockDrop perform well, closely matching Full ResNet. On NF-BoT and NF-UNSW, the F1-score for attack

Table 6
Binary and multi-classification accuracy.

Algorithm	Binary classification						Multi-classification					
	NF-BoT		NF-ToN		NF-UNSW		NF-BoT		NF-ToN		NF-UNSW	
	Accuracy	W-F1	Accuracy	W-F1	Accuracy	W-F1	Accuracy	W-F1	Accuracy	W-F1	Accuracy	W-F1
KNN	87.50%	88.68%	89.29%	89.12%	89.60%	89.76%	82.37%	83.71%	71.25%	71.16%	73.25%	74.45%
LSTM	95.13%	94.96%	89.16%	88.79%	95.54%	95.25%	81.55%	79.66%	69.22%	61.73%	87.39%	86.75%
BlockDrop	99.59%	99.60%	99.43%	99.43%	99.47%	99.46%	95.7%	95.74%	97.49%	97.51%	91.6%	91.4%
Full ResNet	99.82%	99.8%	99.89%	99.9%	99.69%	99.55%	98.02%	98.02%	98.02%	98.04%	92.14%	92.76%
IDNet	99.80%	99.76%	99.87%	99.87%	99.50%	99.50%	97.98%	97.99%	98.02%	98.02%	92.11%	92.18%

Table 7
Detection accuracy comparison on CIC-UNSW

Method	Binary Classification		Multi-Classification	
	Accuracy	W-F1	Accuracy	W-F1
BlockDrop	98.80%	98.60%	91.70%	91.30%
Full ResNet	99.72%	99.71%	93.2%	93.50%
IDNet	99.61%	99.7%	92.9%	93.10%

Table 8
Ablation of mapping strategies on CIC-UNSW

Strategy	Binary Classification		Multi-Classification	
	Accuracy	W-F1	Accuracy	W-F1
w/ Liner	99.40%	98.70%	92.20%	92.40%
w/ Random	98.60%	98.40%	91.80%	92.00%
w/ Proposed	**99.61%**	**99.7%**	**92.90%**	**93.10%**

instance recognition by LSTM and KNN ranges from 70% to 85%, indicating suboptimal performance.

5.3.3. Semantic mapping compatibility and ablation

To evaluate the proposed semantic-aware mapping across different feature extraction methodologies, we apply the same mapping rule to project the 75-dimensional features of CIC-UNSW into the unified representation, and independently train IDNet on CIC-UNSW without modifying the model architecture or input dimensions. As shown in Table 7, IDNet achieves competitive detection performance, indicating that the proposed mapping can represent both feature extraction paradigms within a unified spatial layout when training and testing are performed within the same dataset setting. The ablation results in Table 8 further demonstrate the importance of semantic-aware mapping. Replacing it with a linear or random mapping results in significant performance degradation, indicating that preserving semantic locality contributes to effective feature extraction.

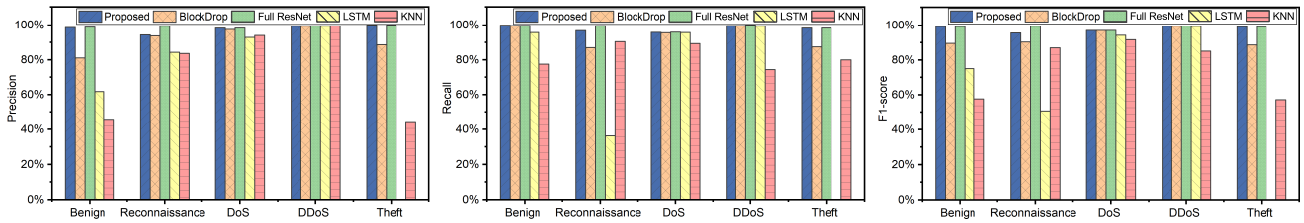
We additionally examine cross-paradigm transferability, not to validate the proposed method, but to characterize the boundary between structural unification and distributional alignment. Models trained on NF-series datasets are directly applied to CIC-UNSW without any domain adaptation. As reported in Table 9, all three settings exhibit high false-positive rates and limited accuracy. This outcome reveals that semantic-aware mapping addresses the structural heterogeneity of feature representations but does not eliminate distributional differences in value ranges, statistical granularity, and extraction methodology between NetFlow and CICFlowMeter. Structural unification is therefore a necessary but not sufficient condition for cross-paradigm transfer. The unified representation established by the proposed mapping nevertheless provides a foundation upon which lightweight domain adaptation techniques can be incorporated, which we plan to explore in future work.

Table 9
Exploratory analysis of cross-paradigm transferability without domain adaptation (NF → CIC)

Method	Accuracy	AUC	FPR
NF-UNSW → CIC-UNSW	39.75%	0.2946	0.9857
NF-BoT → CIC-UNSW	80.73%	0.6215	0.9471
NF-ToN → CIC-UNSW	73.28%	0.5242	0.9675

5.3.4. Unknown attack detection within the NF series

To evaluate the model's ability to detect previously unseen attack types, we design a series of cross-dataset zero-shot experiments within the NF dataset family. All three NF datasets share the same NetFlow-based feature extraction methodology and the same 43-dimensional feature structure, which preserve consistent statistical properties and reduce distributional discrepancies across datasets. Although the datasets contain fine-grained attack-category labels, our zero-shot evaluation focuses on binary malicious traffic detection, i.e., distinguishing benign from malicious traffic. This design reflects practical deployment requirements, in which security systems must first determine whether traffic is malicious before further analysis, especially when encountering unknown attack variants. Specifically, we construct three cross-dataset evaluation settings with complete separation between training and testing attack types, as detailed in Table 10. Overlapping attack categories between dataset pairs are removed from the testing set to ensure that there are no shared classes between training and testing. As reported in Table 11, IDNet consistently outperforms



(a) Precision for each class (b) Recall for each class (c) F1-score for each class

Figure 9: Multi-class comparison on NF-BoT.

Table 10
Zero-shot cross-dataset evaluation settings

Setting	Training Attacks	Testing Attacks
NF-ToN → NF-UNSW	Scanning, Injection, MITM, Password, Ransomware, XSS	Exploits, Generic, Shellcode, Worms, Fuzzers, Recon., Analysis
NF-BoT → NF-UNSW	DDoS, DoS, Recon., Theft	Exploits, Generic, Shellcode, Worms, Fuzzers, Analysis
NF-BoT → NF-ToN	DDoS, DoS, Recon., Theft	Injection, MITM, Password, Ransomware, Scanning, XSS, Backdoor

Table 11
Zero-shot attack detection accuracy comparison

Method	NF-ToN →NF-UNSW	NF-BoT →NF-UNSW	NF-BoT →NF-ToN
KNN	28.83%	33.87%	51.72%
LSTM	25.62%	11.41%	55.12%
ACGAN	30.02%	36.00%	56.98%
IDNet	**93.11%**	**72.58%**	**89.87%**

Table 12
Binary classification results on NF-BoT

Model	Class	Precision	Recall	F1-Score
IDNet	Benign	99.30%	98.84%	99.07%
	Attack	99.83%	99.90%	99.86%
BlockDrop	Benign	97.92%	98.75%	98.43%
	Attack	99.84%	99.69%	99.77%
Full ResNet	Benign	99.32%	99.45%	99.37%
	Attack	99.89%	99.91%	99.88%
LSTM	Benign	86.85%	73.39%	79.55%
	Attack	96.15%	98.35%	97.24%
KNN	Benign	50.96%	82.43%	62.98%
	Attack	97.14%	88.25%	92.48%

all baseline methods in the three zero-shot settings. These results highlight IDNet's strong generalization ability in detecting unseen attack categories within the NF series, supporting its applicability in scenarios where new attack variants continuously emerge.

Table 13
Binary classification comparison on NF-ToN

Model	Class	Precision	Recall	F1-Score
IDNet	Benign	99.12%	99.89%	99.50%
	Attack	99.98%	99.87%	99.93%
BlockDrop	Benign	98.83%	99.35%	99.50%
	Attack	99.65%	99.37%	99.10%
Full ResNet	Benign	99.76%	99.83%	99.84%
	Attack	99.99%	99.88%	99.95%
LSTM	Benign	91.68%	95.17%	93.39%
	Attack	76.27%	64.27%	69.76%
KNN	Benign	91.80%	94.30%	93.03%
	Attack	80.43%	73.54%	76.83%

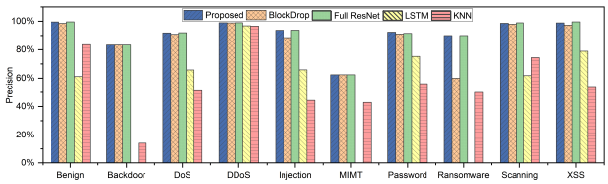
Table 14
Binary classification comparison on NF-UNSW

Model	Class	Precision	Recall	F1-Score
IDNet	Benign	99.31%	99.96%	99.64%
	Attack	99.91%	98.52%	99.71%
BlockDrop	Benign	99.28%	99.57%	99.15%
	Attack	99.88%	98.44%	99.61%
Full ResNet	Benign	99.38%	99.95%	99.75%
	Attack	99.95%	99.30%	99.74%
LSTM	Benign	95.25%	99.73%	97.43%
	Attack	97.89%	71.84%	82.86%
KNN	Benign	95.20%	89.34%	92.18%
	Attack	79.45%	90.15%	84.46%

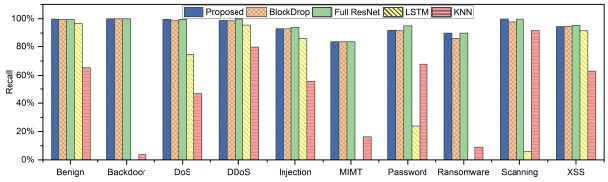
5.3.5. Impact of attack types

Figs. 9-11 illustrate the multi-class performance of different methods across three datasets. Both the proposed method and Full ResNet outperform other models across all metrics. Notably, the NF-BoT has the highest proportion of attack traffic, followed by NF-ToN, with the lowest on NF-UNSW. As the proportion of attacks decreases, the difficulty of accurate identification increases. The ability to accurately identify less prevalent attack types is the most convincing.

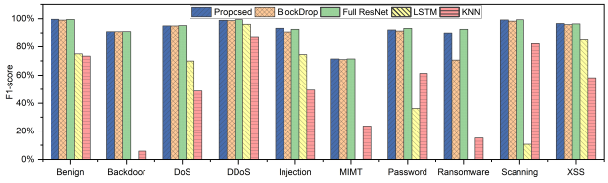
As shown in Fig. 9, in NF-BoT, the detection performance of the five methods for the more prevalent DoS and DDoS categories is satisfactory. However, LSTM fails to



(a) Precision for each class

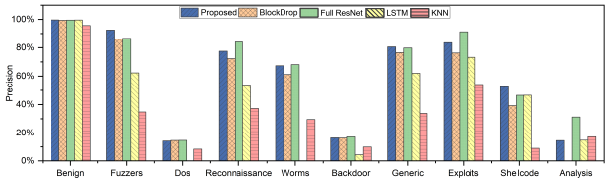


(b) Recall for each class

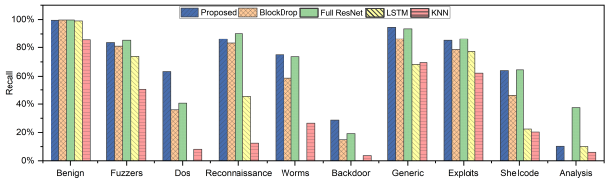


(c) F1-score for each class

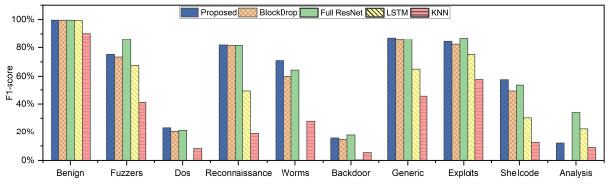
Figure 10: Multi-class performance comparison on NF-ToN.



(a) Precision for each class



(b) Recall for each class



(c) F1-score for each class

Figure 11: Multi-class performance comparison on NF-UNSW.

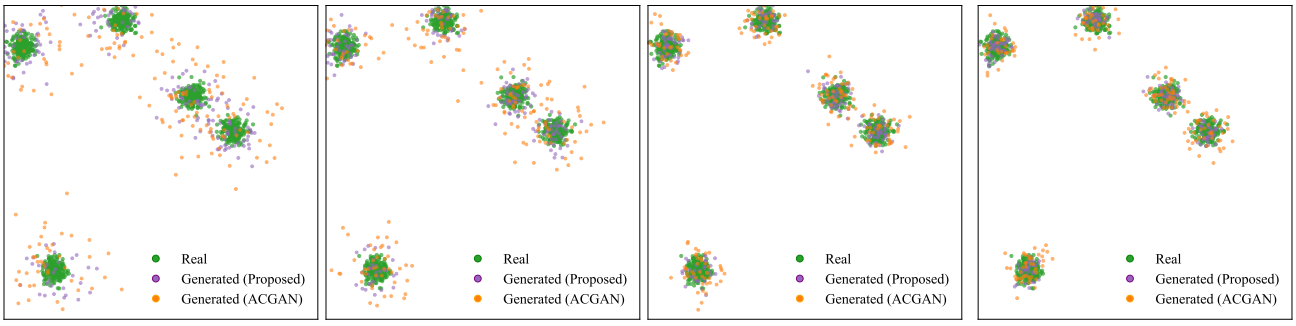
identify Theft, which constitutes only 0.01% of the dataset. In contrast, the proposed method achieves a precision, recall, and F1 score of 99.99%, 98.25%, and 99.12%, respectively, comparable to Full ResNet and significantly better than BlockDrop and KNN. The results demonstrate that the proposed method has a significant advantage in identifying attack instances that benchmark models struggle to detect.

As depicted in Fig. 10, on the NF-ToN, both IDNet and Full ResNet exhibit excellent classification performance. Specifically, for the least prevalent Ransomware, the proposed method and Full ResNet demonstrate outstanding performance in F1-score, precision, and recall. In contrast, LSTM fails to recognize this type of attack, and the detection results of BlockDrop and KNN are also unsatisfactory. The proposed method can significantly improve the detection performance for some less prevalent attack instances. Due to ResNet's powerful feature extraction capability, the misclassification rates of IDNet and Full ResNet are significantly lower than those of BlockDrop, LSTM, and KNN.

Compared to NF-BoT and NF-ToN, the class imbalance in NF-UNSW is more pronounced, with the total proportion of all attack types not exceeding 4%. Nevertheless, Figs. 11(a)-(c) show that the precision, F1-score, and recall of the proposed method and Full ResNet still outperform the other methods, particularly for the least prevalent Shellcode and Worms. LSTM fails to detect Worms and DoS, while BlockDrop fails to detect Analysis. Compared to BlockDrop, the proposed method improves the precision, recall, and F1-score for Worms by 6.13%, 16.18%, and 10.97%, respectively. Similarly, for Analysis, the proposed method improves the three metrics by 14.55%, 10.23%, and 12.02%, respectively. The proposed method has made progress in detecting the two least prevalent attack instances, and future efforts will continue to explore improving the detection of DoS, Analysis, and Backdoor attacks.

5.4. Quality analysis of generated traffic instances

To assess the effectiveness of the proposed generation framework in modeling network traffic instances, we conducted a distribution visualization experiment using NF-BoT. Since NF-BoT is highly imbalanced with over 99% malicious traffic, we adopted a class-balanced sampling strategy to ensure that the evaluation reflects G's ability to model each class individually rather than being dominated by the majority class. Specifically, 1000 real instances were randomly sampled from each traffic class, and 200 corresponding synthetic instances per class were generated at various training stages (500, 1000, 1500, and 2000 epochs). Using t-SNE [50], we projected the traffic instances into a two-dimensional space, where green points denote real traffic instances. As shown in Fig. 12, the original ACGAN (orange points) initially produced widely dispersed traffic instances that deviated significantly from real traffic distributions, gradually aligning with real traffic clusters as training progressed. In contrast, IDNet (purple points) demonstrated faster convergence and achieved near-complete alignment with real traffic distributions by epoch 2000. Moreover, Fig. 13 presents a boxplot of cluster distances. Although both methods eventually converge toward the real distribution, the proposed approach consistently maintains more compact cluster structures with fewer outliers. These results confirm that the proposed method not only accelerates convergence but also enhances G's ability to capture the true traffic distribution while ensuring clear inter-class separability.



(a) 500 epochs. (b) 1000 epochs. (c) 1500 epochs. (d) 2000 epochs.

Figure 12: Distribution of real and generated traffic instances on NF-BoT.

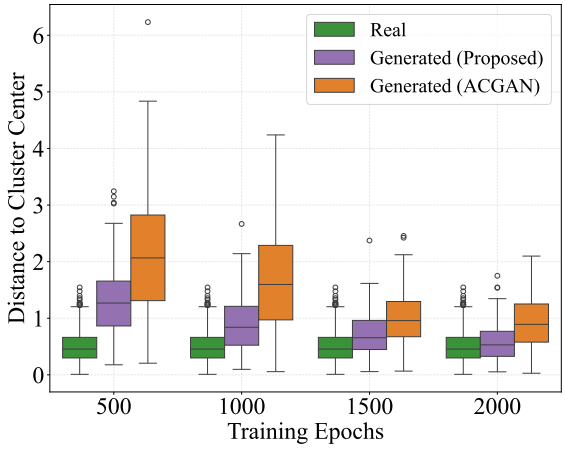


Figure 13: Cluster distance distribution across training epochs.

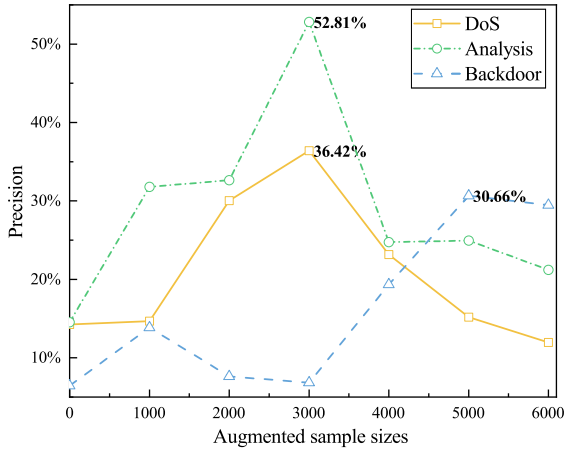


Figure 14: Precision under different augmented sample sizes.

5.5. Data augmentation effectiveness analysis

To evaluate the effectiveness of our synthetic traffic instances, we conduct data augmentation experiments on a downstream attack detection model. We systematically augment the NF-UNSW dataset by adding 1000-6000 synthetic samples for DoS, Analysis, and Backdoor attack types to address class imbalance and assess improvements in detection performance. As illustrated in Fig.14, the detection accuracy initially improves with increasing numbers of synthetic attack samples before eventually decreasing. For DoS attacks (yellow line) and analysis attacks (green line), precision peaks at 3000 synthetic samples, resulting in improvements of 22.16% and 38.26% compared to the original dataset. Backdoor (blue line) attack detection reaches maximum precision with 5000 additional samples, yielding a 24.21% improvement. Adding sufficient attack samples can balance the original dataset's class distribution. However, excessive sample generation may lead G to learn biased features, thus diminishing the model's detection performance. Additionally, the integration of random noise during sample generation simulates real-world uncertainty [51], enhancing IDNet's generalization capability to unseen attack instances.

6. Conclusion

This paper proposes IDNet, an instance-adaptive intrusion detection framework based on adversarially optimized dynamic networks. To handle heterogeneous traffic structures, we introduce a semantic-aware mapping mechanism that transforms diverse flow features into unified grayscale matrices, enabling consistent spatial encoding across datasets. Built on this foundation, IDNet combines adversarial self-optimized pre-training with augmentation-assisted fine-tuning to jointly improve detection accuracy and inference efficiency. Experiments across multiple benchmark datasets show that IDNet significantly reduces computational cost while maintaining high accuracy, with routing behavior that clearly corresponds to traffic instance complexity. The proposed mapping further allows the model to be independently trained on both NetFlow- and CICFlowMeter-based datasets without dimensional modification, and zero-shot experiments confirm strong detection performance on unseen attack types.

Future work will focus on cross-domain intrusion detection and routing interpretability. Building on the unified structural representation established by the proposed mapping, we plan to incorporate lightweight domain adaptation techniques to improve robustness and transferability across different feature extraction paradigms.

References

[1] Zinuo Yin, Hongchang Chen, Hailong Ma, Tao Hu, and Luxin Bai. CAEAID: An incremental contrast learning-based intrusion detection framework for IoT networks. *Computer Networks*, 262:111161, 2025.

[2] Renjie Xu, Guangwei Wu, Weiping Wang, Xing Gao, An He, and Zhengpeng Zhang. Applying self-supervised learning to network intrusion detection for network flows with graph neural network. *Computer Networks*, 248:110495, 2024.

[3] Jie Gu and Shan Lu. An effective intrusion detection approach using SVM with naïve Bayes feature embedding. *Computers & Security*, 103:102158, 2021.

[4] Paulo Angelo Alves Resende and André Costa Drummond. A survey of random forest based methods for intrusion detection systems. *ACM Computing Surveys*, 51(3):1–36, 2018.

[5] Mikel K Ngueajio, Gloria Washington, Danda B Rawat, and Yolande Ngueabou. Intrusion detection systems using support vector machines on the KDDCUP'99 and NSL-KDD datasets: A comprehensive survey. In *Proc. of SAI Intelligent Systems Conference*, pages 609–629, 2022.

[6] Yakubu Imrana, Yanping Xiang, Liaqat Ali, and Zaharawu Abdul-Rauf. A bidirectional LSTM deep learning approach for intrusion detection. *Expert Systems with Applications*, 185:115524, 2021.

[7] Shiyu Wang, Wenxiang Xu, and Yiwen Liu. Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. *Computer Networks*, 235:109982, 2023.

[8] Hang Shen, Yanke Yao, Tianjing Wang, and Guangwei Bai. MobiFormer: Split-federated transfer learning for drone RAN slicing with multi-head attention. *IEEE Transactions on Mobile Computing*, 25(4):4920–4935, 2026.

[9] Xiaoliang Dai, Hongxu Yin, and Niraj K Jha. Grow and prune compact, fast, and accurate LSTMs. *IEEE Transactions on Computers*, 69(3):441–452, 2019.

[10] Ruijie Zhao, Mingwei Zhan, Xianwen Deng, Fangqi Li, Yanhao Wang, Yijun Wang, Guan Gui, and Zhi Xue. A novel self-supervised framework based on masked autoencoder for traffic classification. *IEEE/ACM Transactions on Networking*, 32(3):2012–2025, 2024.

[11] Yu Liu, Hang Shen, Tianjing Wang, and Guangwei Bai. Vehicle counting in drone images: An adaptive method with spatial attention and multiscale receptive fields. *ETRI Journal*, 47(1):7–19, 2025.

[12] Danish Vasan, Mamoun Alazab, Sobia Wassan, Hamad Naeem, Babak Safaei, and Qin Zheng. IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138, 2020.

[13] Gueltoum Bendiab, Stavros Shiaeles, Abdulrahman Alruban, and Nicholas Kolokotronis. IoT malware network traffic classification using visual representation and deep learning. In *Proc. of IEEE Conference on Network Softwarization (NetSoft)*, pages 444–449, 2020.

[14] Vahidreza Niazmand and Qiang Ye. Joint task offloading, DNN pruning, and computing resource allocation for fault detection with dynamic constraints in industrial IoT. *IEEE Transactions on Cognitive Communications and Networking*, 11(5):3486–3501, 2025.

[15] Rahul Umesh Mhapsekar, Lizy Abraham, Steven Davy, and Indrakshi Dey. Application adaptive light-weight deep learning (AppAdapt-LWDL) framework for enabling edge intelligence in dairy processing. *IEEE Transactions on Mobile Computing*, 24(2):1105–1119, 2025.

[16] Zhuoqing Chang, Shubo Liu, Xingxing Xiong, Zhaohui Cai, and Guoqing Tu. A survey of recent advances in edge-computing-powered artificial intelligence of things. *IEEE Internet of Things Journal*, 8(18):13849–13875, 2021.

[17] Jifeng Guo, C. L. Philip Chen, Zhulin Liu, and Xixin Yang. Dynamic neural network structure: A review for its theories and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):4246–4266, 2025.

[18] Haseena Rahmath P, Vishal Srivastava, Kuldeep Chaurasia, Roberto G Pacheco, and Rodrigo S Couto. Early-exit deep neural network-a comprehensive survey. *ACM Computing Surveys*, 57(3):1–37, 2024.

[19] Zhihao Lin, Yongtao Wang, Jinhe Zhang, and Xiaojie Chu. Dynamicdet: A unified dynamic architecture for object detection. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6282–6291, 2023.

[20] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021.

[21] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. BranchyNet: Fast inference via early exiting from deep neural networks. In *Proc. of International Conference on Pattern Recognition (ICPR)*, pages 2464–2469, 2016.

[22] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1039–1048, 2017.

[23] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[24] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. SkipNet: Learning dynamic routing in convolutional networks. In *Proc. of European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[25] Hang Shen, Qi Liu, Yu Liu, Tianjing Wang, and Guangwei Bai. MA-DyNN: Modal-adaptive dynamic neural network for crowd-counting on consumer drones. *IEEE Transactions on Consumer Electronics*, 71(4):9594–9605, 2025.

[26] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[27] Yue Wang, Jianghao Shen, Ting-Kuei Hu, Pengfei Xu, Tan Nguyen, Richard Baraniuk, Zhangyang Wang, and Yingyan Lin. Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):623–633, 2020.

[28] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. BlockDrop: Dynamic inference paths in residual networks. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8817–8826, 2018.

[29] Zhourong Chen, Yang Li, Samy Bengio, and Si Si. You look twice: Gaternet for dynamic filter selection in CNNs. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9172–9180, 2019.

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] Yingcheng Su, Yichao Wu, Ken Chen, Ding Liang, and Xiaolin Hu. Dynamic multi-path neural network. In *Proc. of International Conference on Pattern Recognition (ICPR)*, pages 4137–4144, 2021.

[34] Sam Leroux, Pieter Simoens, Bart Dhoedt, P Molchanov, T Breuel, and J Kautz. IamNN: iterative and adaptive mobile neural network for efficient image classification. In *International Conference on Learning Representations (ICLR)*, pages 1–4, 2018.

[35] Zhengfa Li, Chuanhe Huang, and Wanyu Qiu. An intrusion detection method combining variational auto-encoder and generative adversarial networks. *Computer Networks*, 253:110724, 2024.

[36] Fang Li, Hang Shen, Jieai Mai, Tianjing Wang, Yuanfei Dai, and Xiaodong Miao. Pre-trained language model-enhanced conditional

generative adversarial networks for intrusion detection. *Peer-to-Peer Networking and Applications*, 17(1):227–245, 2024.

[37] Hang Shen, Qi Liu, Fang Li, Tianjing Wang, Yuanfei Dai, and Guangwei Bai. Split-federated BERT with adversarial training for edge intrusion detection. *IEEE Internet of Things Journal*, 2026. DOI: 10.1109/JIOT.2026.3679000.

[38] Hongwei Ding, Yu Sun, Nana Huang, Zhidong Shen, and Xiaohui Cui. TMG-GAN: Generative adversarial networks-based imbalanced learning for network intrusion detection. *IEEE Transactions on Information Forensics and Security*, 19:1156–1167, 2024.

[39] Hongwei Ding, Leiyang Chen, Liang Dong, Zhongwang Fu, and Xiaohui Cui. Imbalanced data classification: A KNN and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems*, 131:240–254, 2022.

[40] Tianjing Wang, Qi Liu, Hang Shen, Xiaokang Luo, and Guangwei Bai. Graph neural network-enhanced auxiliary classifier generative adversarial network framework for robust intrusion detection. *ETRI Journal*, 2025. DOI: 10.4218/etrij.2025-0152.

[41] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[42] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[43] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7008–7024, 2017.

[44] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[45] Timothy Dozat. Incorporating Nesterov Momentum into Adam. In *Proc. of International Conference on Learning Representations (ICLR)*, pages 1–4, 2016.

[46] Hakan Aydın, Zeynep Orman, and Muhammed Ali Aydın. A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment. *Computers & Security*, 118:102725, 2022.

[47] Ilhan Firat Kilincer, Fatih Ertam, and Abdulkadir Sengur. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. *Computer Networks*, 188:107840, 2021.

[48] Shiyun Wang, Qiang Ye, and Yujie Tang. IEDL-IDS: an image-enhanced encoder-based deep learning scheme for intrusion detection systems. *ACM Transactions on Privacy and Security*, 29(1):1–35, 2026.

[49] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[50] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[51] Haonan Yan, Xiaoguang Li, Wenjing Zhang, Rui Wang, Hui Li, Xingwen Zhao, Fenghua Li, and Xiaodong Lin. Automatic evasion of machine learning-based network intrusion detection systems. *IEEE Transactions on Dependable and Secure Computing*, 21(1):153–167, 2024.