Drift-Aware Conditional Diffusion with Compact Feature Representations for Malicious Traffic Detection

Zhihao Zheng, Wei Xia*, Shituo Ma, Xiang Luo, Gang Xiong, Gaopeng Gou
Institute of Information Engineering, Chinese Academy of Sciences
School of Cybersecurity, University of Chinese Academy of Sciences
Beijing, China
{zhengzhihao, xiawei, mashituo, luoxiang, xionggang, gougaopeng}@iie.ac.cn

***Abstract*—The rapid evolution of cyber-attack techniques poses significant challenges to malicious traffic identification, particularly under concept drift. Existing drift-aware approaches largely rely on autoencoder-based reconstruction, whose limited expressive capacity and generalization hinder the detection of subtle distributional shifts among diverse malicious traffic types. To address this limitation, we propose FlowDiff, a drift-aware framework that leverages class-conditional diffusion-based reconstruction in a compact feature space. FlowDiff first learns stable and discriminative compact representations through a Compact Feature Module (CFM). On top of these representations, a Conditional Diffusion Model (CDM) captures class-conditional feature distributions and enables drift detection via a loss-adaptive diffusion mechanism. In addition, a lightweight LightGBM classifier is employed for accurate in-distribution traffic classification. Extensive experiments on real-world malicious traffic datasets demonstrate that FlowDiff consistently outperforms state-of-the-art methods in closed-world classification, drift detection, and open-world recognition, with particularly significant improvements in detecting previously unseen malicious traffic.**

***Index Terms*—Concept drift, Malicious traffic identification, Compact feature representation, Diffusion model, LightGBM**

I. Introduction

Polymorphic malware, zero-day exploits, and sophisticated adversarial behaviors continuously reshape network traffic patterns, making traditional intrusion detection systems (IDSs) increasingly inadequate. Most existing solutions, including rule-based systems and static machine learning models, assume a stationary environment, which rarely holds in real-world networks and often leads to performance degradation and missed detections [1].

In practice, concept drift in intrusion detection mainly appears in two forms: distributional shifts within existing traffic classes and the emergence of previously unseen traffic types [2]. This work focuses on the latter, which poses a fundamental challenge for security systems operating under open-world conditions.

Existing drift detection approaches for network traffic analysis can be broadly divided into reconstruction-based methods [3], which identify unseen patterns through reconstruction errors, and prototype-based methods [1], [4], which detect deviations based on distances to representative class embeddings. Despite their differences, these methods often struggle to capture subtle distributional shifts across diverse malicious traffic types. Moreover, many frameworks jointly train feature extraction and classification modules, which entangles representation learning with classification objectives and may weaken discriminative capacity even under closed-world conditions. As a result, modern network security systems face two critical challenges:

- Accurately recognizing known attack types while remaining sensitive to previously unseen threats in open-world settings.
- Detecting when incoming traffic deviates from known patterns.

A key observation motivating this work is that diffusion-based generative models have been shown to provide substantially stronger reconstruction fidelity and distribution modeling capability than conventional autoencoder-based approaches, particularly in capturing fine-grained variations and complex multimodal data distributions [5], [6]. Such properties suggest that diffusion-based reconstruction can offer a more sensitive and expressive signal for detecting subtle distributional shifts, which is crucial for concept drift identification.

Based on this insight, we introduce *FlowDiff*, a drift-aware framework that adopts a decoupled architecture integrating compact feature representation learning with diffusion-based generative modeling.

Our main contributions are summarized as follows:

- We propose **FlowDiff**, a drift-aware malicious traffic detection framework combining compact feature learning, conditional diffusion modeling, and LightGBM-based classification.
- We design a **Compact Feature Module (CFM)** to learn stable and discriminative embeddings under evolving traffic distributions.
- We develop a **Conditional Diffusion Model (CDM)** with a **Loss-Adaptive Diffusion Detection (LADD)** mechanism to detect drifting traffic via reconstruction dynamics.

* Corresponding author

- We validate FlowDiff on real-world malicious traffic datasets, showing superior performance over state-of-the-art baselines in classification and drift detection.

II. Related Work

A. Closed-World Encrypted Traffic Classification

Encrypted traffic classification is a fundamental task in network security applications such as intrusion detection. Existing closed-world approaches assume a fixed and known set of traffic classes and can be broadly divided into supervised and unsupervised methods.

a) Supervised Methods: Supervised approaches learn discriminative representations from labeled encrypted traffic. Representative methods include FS-Net [7], which models traffic flows using recurrent encoder–decoder architectures, and ET-BERT [8], which leverages large-scale pre-training with transformer models. Lightweight designs have also been explored, such as CNN-based models with knowledge distillation [9] and self-attention enhanced recurrent architectures like BiRNN-SA [10]. While these methods achieve strong performance under closed-world assumptions, they rely heavily on labeled data and are unable to handle previously unseen traffic classes.

b) Unsupervised Methods: Unsupervised approaches aim to detect malicious traffic without relying on labeled data, typically by modeling benign traffic distributions. Prior work includes contrastive learning-based methods such as ContraMTD [11], multiscale autoencoder frameworks like MTCR-AE [12], reconstruction-based schemes such as unFlowS-Net [13], and graph autoencoder models including GTAE-IDS [14]. Although capable of detecting unknown attacks, these methods generally lack fine-grained classification capability and are sensitive to evolving traffic patterns.

B. Drift-Aware Encrypted Traffic Analysis

To address concept drift in encrypted traffic analysis, early studies primarily relied on confidence-based methods, which assume that drifting or unknown samples yield lower prediction confidence. For example, Zhang et al. [15] and Pathmaperuma et al. [16] exploited confidence score distributions to identify unknown traffic. However, such approaches are fundamentally limited by the overconfidence of softmax-based classifiers.

Reconstruction-based methods detect drift through elevated reconstruction errors. ZTI [3] identifies unseen flows using autoencoder reconstruction loss, while ZeroWall [17] models benign traffic semantics using encoder–decoder architectures. Prototype-based approaches further model class-level representations in latent space. SEEN [18] adopts metric learning for unknown detection, while CADE [1] and ICE-CP [4] detect drift based on distances to learned class prototypes. Despite their effectiveness, both reconstruction- and prototype-based methods often struggle to capture fine-grained distributional shifts.

C. Diffusion-Based Approaches in Encrypted Traffic Modeling

Diffusion models have recently been introduced into intrusion detection and network security research. Tang et al. [19] applied diffusion models for rare-class data generation, while Merzouk et al. [20] explored diffusion-based adversarial purification. Cai et al. [21] combined denoising diffusion probabilistic models with attention mechanisms to enhance intrusion detection performance.

Beyond intrusion detection, diffusion-based techniques have been applied to other traffic analysis tasks, including Tor traffic purification [22], mobile traffic prediction [23], and traffic generation via diffusion-based time-series modeling [24]. However, diffusion models have not yet been explored for concept drift detection in encrypted traffic analysis. This work addresses this gap by introducing diffusion-based reconstruction as a drift-sensitive detection mechanism.

III. Methodology

To effectively detect concept drift in network traffic while maintaining accurate in-distribution classification, FlowDiff is designed as a modular framework that integrates representation learning with generative modeling and classification. Based on this design principle, the overall architecture of FlowDiff is illustrated in Figure 1 and consists of three main modules:

- **Compact Feature Module (CFM).** The CFM extracts stable and discriminative features from traffic samples, forming a compact and robust representation space for downstream tasks.
- **Drift Detection Module.** Based on the compact features and the outputs of the diffusion model, this module detects samples that deviate from the training distribution.
- **In-Distribution Sample Classifier.** This module predicts labels for samples identified as in-distribution, leveraging the learned compact feature representations.

The detailed design and operation of each module are described in the following subsections.

A. Problem Formulation

Let $D_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ denote the training set, where $x_i \in \mathbb{R}^D$ represents a network flow and $y_i \in \{1, \ldots, C\}$ is the corresponding known malicious class. During testing, samples may originate either from the same C known classes (closed-world setting) or from additional previously unseen classes (open-world setting).

The objective is to learn a model that (i) accurately classifies samples from known classes, and (ii) detects samples whose distributions deviate from the training data. Detected samples are treated as drifting traffic and assigned to a unified unknown class.

B. CFM

This module learns compact and discriminative representations for traffic samples, enabling clear separation of different classes in the feature space. These representations serve as the basis for subsequent drift detection and in-distribution

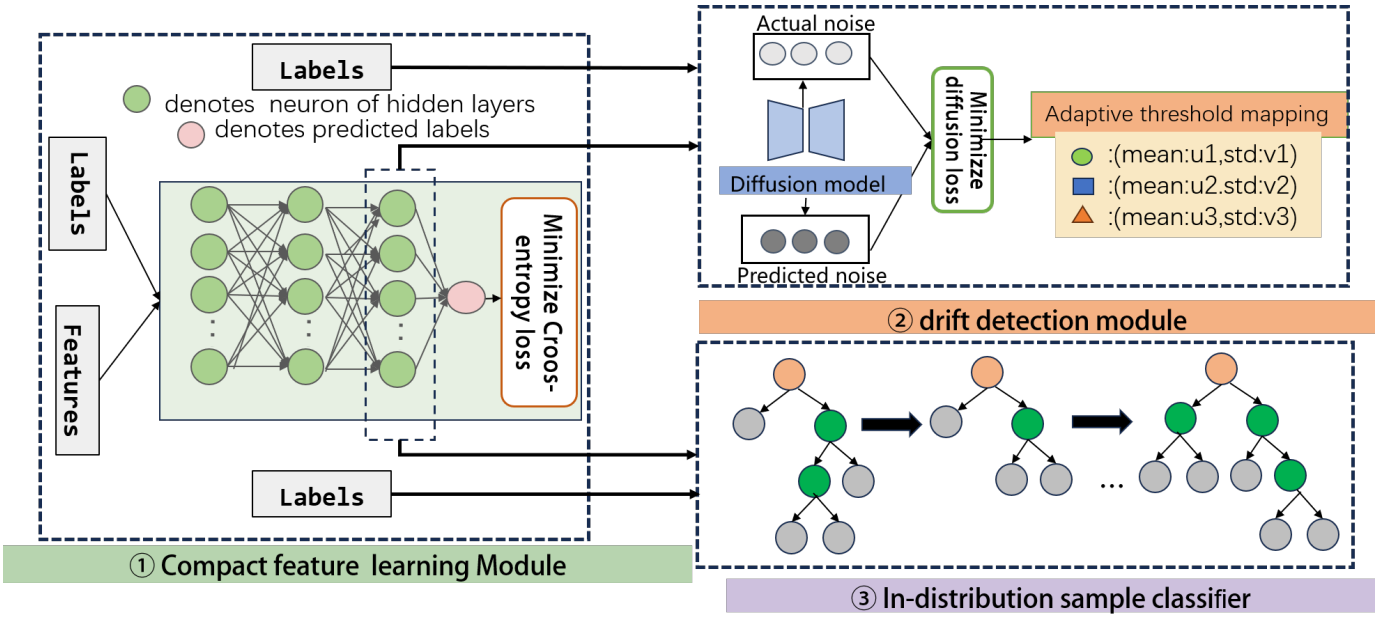

Fig. 1: Overall framework of FlowDiff.

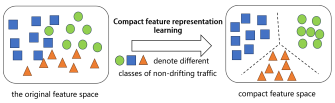

Fig. 2: Effect of the CFM.

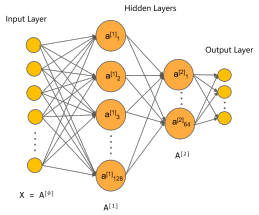

Fig. 3: Architecture of the CFM.

classification. An illustration of the effect of the CFM is shown in Figure 2.

The model is trained using the standard cross-entropy loss, and its architecture is shown in Figure 3.

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log \hat{y}_{i,c}, \tag{1}$$

After training, the output of the last hidden layer of the MLP is taken as the compact feature representation of each traffic sample. This approach not only enhances inter-class separability but also provides high-quality feature inputs for subsequent drifting sample detection and classification while maintaining a lightweight model structure. Furthermore, the module is computationally efficient, making it suitable for training and deployment on large-scale network traffic datasets.

C. Drift Detection Module

In this section, we explain our approach to detecting OOD malicious network flows from the following three aspects: 1) the overall architecture of the CDM; 2) the backbone network of the CDM ; 3) the drift detection algorithm.

1) Overall Architecture of the CDM: The CDM [6] models class-conditional distributions of compact feature representations. Let $z_i \in \mathbb{R}^d$ be the compact feature vector from a pretrained MLP.

a) Forward diffusion: A forward noising process is applied to z_i using a learnable SNR schedule:

$$\gamma_t = \gamma_{\min} + (\gamma_{\max} - \gamma_{\min})\, t, \qquad t \in [0,1], \tag{2}$$

$$\alpha_t = \text{sigmoid}(-\gamma_t), \tag{3}$$

$$z_i^t = \sqrt{\alpha_t}\, z_i + \sqrt{1-\alpha_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{4}$$

where $\gamma_{\min}$ and $\gamma_{\max}$ are learnable.

b) Noise prediction: Given the noisy feature z_i^t and class embedding h_y, the backbone predicts the noise $\hat{\epsilon}_\theta$. The diffusion error, $\epsilon_i - \hat{\epsilon}_\theta$, is used to detect out-of-distribution samples via class-specific thresholds.

$$\mathcal{L}_{\text{diff}} = \frac{1}{N}\sum_{i=1}^{N} \left\| \epsilon_i - \hat{\epsilon}_\theta(z_i^t, h_y, \gamma_t) \right\|^2. \tag{5}$$

c) Conditional encoding: We define a learnable embedding function that maps discrete class labels $y_i \in \{0, \ldots, C-1\}$ to continuous vectors $h_y \in \mathbb{R}^{128}$:

$$h_y = \text{LabelEncoder}(y_i), \quad \text{LabelEncoder} : \mathbb{Z}_C \to \mathbb{R}^{128}.$$

This function converts each class label into a 128-dimensional embedding used by the diffusion model.

d) Noise prediction: Given the noisy feature z_i^t, class embedding h_y, and SNR parameter γ_t, the backbone network predicts the noise component:

$$\hat{\epsilon}_\theta = \text{Backbone}(z_i^t, h_y, \gamma_t), \tag{6}$$

where the backbone serves as a learnable function mapping the concatenated inputs to a noise prediction. The model is trained by minimizing the diffusion loss:

$$\mathcal{L}_{\text{diff}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\epsilon_i - \hat{\epsilon}_\theta(z_i^t, h_y, \gamma_t)\right\|^2, \tag{7}$$

where N is the number of training samples. The detailed architecture of the backbone network is described in Section III-C2.

2) Backbone Network of the CDM: The backbone network is a fully connected U-Net-style architecture designed for one-dimensional feature vectors. It comprises an input layer, an encoder, a decoder, and an output layer.

a) Input layer: The network input consists of the noisy feature vector $z_i^t \in \mathbb{R}^{128}$, the SNR parameter γ_t, and the class embedding $h_y \in \mathbb{R}^{128}$. The scalar γ_t is encoded using a sinusoidal embedding and projected to 128 dimensions through a two-layer fully connected network with GELU activation. The class embedding h_y is projected in the same manner. These representations are concatenated to form the network input:

$$\mathbf{x}_{\text{in}} = [\, z_i^t,\ \mathbf{e}_\gamma^{\text{p}},\ h_y^{\text{p}}\,] \in \mathbb{R}^{384}, \tag{8}$$

where $\mathbf{e}_\gamma^{\text{p}}$ and h_y^{p} denote the projected embeddings.

b) Encoder and decoder: The encoder consists of five fully connected blocks with hidden dimensions decreasing as $4096 \to 2048 \to 1024 \to 512 \to 256$, while the decoder mirrors this structure with dimensions increasing as $256 \to 512 \to 1024 \to 2048 \to 4096$. Each block applies a linear transformation followed by layer normalization and a GELU activation. Skip connections are implemented via element-wise addition between corresponding encoder and decoder layers.

c) Output layer: A final linear projection maps the decoder output to the original feature dimension, producing the predicted noise vector $\hat{\epsilon}_\theta \in \mathbb{R}^{B\times 128}$, where B denotes the batch size.

3) Drift Detection Algorithm: We adopt a class-wise, loss-based strategy for out-of-distribution (OOD) detection using the trained CFM and CDM. During training, samples are mapped to the compact feature space and a diffusion reconstruction loss is computed for each class to estimate class-specific loss statistics.

For each class c, the mean μ_c and standard deviation σ_c of diffusion losses are calculated, and a detection threshold is defined as $\tau_c = \mu_c + k\sigma_c$, where k controls the detection sensitivity.

At inference time, a test sample is first projected into the compact feature space and assigned a predicted class $\hat{y}$ by the CFM. The corresponding diffusion loss is then computed using the CDM. Samples whose loss exceeds the class-specific threshold are identified as OOD; otherwise, they are regarded as in-distribution. The complete procedure is summarized in Algorithm 1.

Algorithm 1: Loss-adaptive Diffusion Detection

Input: Trained CDM and CFM, Training set $\{(x_i, y_i)\}$, Test sample x_{test}
Output: OOD decision for x_{test}
Training:
for *each class* c **do**
  for *each* (x_i, y_i) *in* c **do**
    $z_i \leftarrow \text{CFM}(x_i),\ \mathcal{L}_i \leftarrow \text{CDM}(z_i, y_i, t)$
  end
  $\mu_c \leftarrow \text{mean}(\{\mathcal{L}_i\}),\ \sigma_c \leftarrow \text{std}(\{\mathcal{L}_i\}),\ \tau_c \leftarrow \mu_c + k \cdot \sigma_c$
end
Testing:
$z_{\text{test}} \leftarrow \text{CFM}(x_{\text{test}}),\ \hat{y} \leftarrow$ predict label using CFM, $\mathcal{L}_{\text{test}} \leftarrow \text{CDM}(z_{\text{test}}, \hat{y}, t)$ **if** $\mathcal{L}_{\text{test}} > \tau_{\hat{y}}$ **then**
  classify as OOD
else
  classify as in-distribution
end

D. In-Distribution Sample Classifier

The compact feature representations learned by the CFM are used to train a LightGBM classifier for in-distribution network traffic recognition. LightGBM is selected for its efficiency and strong performance on structured feature representations.

By leveraging gradient boosting decision trees, the classifier effectively models nonlinear feature interactions while maintaining low computational overhead. As a result, it serves as a reliable in-distribution classifier within our framework.

IV. Experiment

A. Experimental Setup

We conduct comparative experiments on the CIC-IDS2018 dataset, which is provided by the Canadian Institute for Cybersecurity. The original dataset contains 14 categories of malicious traffic,with the statistics summarized in Table I. To ensure sufficient data for each category, we discard four malicious traffic categories whose sample counts are less than 2000. As a result, our experiments are conducted on 10 malicious traffic categories.

1) Data Preprocessing: Traffic features are extracted using *CICFlowMeter*. To prevent potential bias, timestamp and IP address related fields are removed. Categorical features are appropriately encoded, and numerical features are normalized.

2) Baseline Methods: We compare the proposed FlowDiff with several representative approaches:

- **DIAI [16]:** A confidence-based method for drift detection that uses a deep neural network to identify in-distribution samples. Samples with prediction confidence below a threshold are considered potential drift instances.
- **OpenMax [25]:** An open-set recognition method that estimates the probability of a sample belonging to an unknown class using activation vectors and Extreme Value Theory (EVT). For network traffic, OpenMax is adapted by replacing the image-based feature extractor with a shallow fully connected neural network processing flow-level statistical features, enabling the identification of unseen or drifting traffic patterns.
- **ZTI [3]:** A reconstruction-based method combining classification and drift detection. A CNN first classifies network traffic, and an autoencoder measures reconstruc-

TABLE I: Statistics of CIC-IDS2018 traffic categories.

Class	Number	Class	Number	Class	Number
Benign	13,484,708	DDOS attack-HOIC	686,012	DDoS attacks-LOIC-HTTP	576,191
DoS attacks-Hulk	461,912	Bot	286,191	FTP-BruteForce	193,360
SSH-Bruteforce	187,589	Infilteration	161,934	DoS attacks-SlowHTTPTest	139,890
DoS attacks-GoldenEye	41,508	DoS attacks-Slowloris	10,990	DDOS attack-LOIC-UDP	1,730
Brute Force-Web	611	Brute Force-XSS	230	SQL Injection	87

tion errors, with unusually high errors indicating drifting samples.

- **CADE [1]:** A prototype-based method that projects traffic samples into a latent space via a contrastive autoencoder. Distances to class centroids determine whether a sample belongs to known classes or represents drift.
- **ICE-CP [4]:** A representation-learning approach that integrates a variational autoencoder with a Class-Perception detector. It identifies drifting samples based on latent representations and distances to class centroids, effectively distinguishing in-distribution and drifting traffic.

3) Experimental Design: We evaluate the models in three experimental settings: *closed-world classification*, *drift detection*, and *open-world recognition*. Following the design of ICE-CP [4], we configure 1, 2, and 3 OOD classes. For each OOD configuration in each experiment, three selections of OOD categories are performed, with the remaining categories used as in-distribution classes. For every in-distribution class, 10,000 samples are used for training and 1,000 for testing; each OOD class contributes 200 test samples to simulate the realistic imbalance where drifting samples are rare. To ensure the robustness and statistical reliability of results, each model is trained five times on identical sampled datasets.

4) Evaluation Metrics: To evaluate the performance of all methods, we consider two types of tasks in our framework: multi-class traffic classification (*closed-world* and *open-world*) and *drift detection*. Correspondingly, different evaluation metrics are adopted for each task.

For multi-class classification, we report macro-averaged Precision, Recall, and F1 Score. These metrics treat all classes equally and are widely used to assess overall classification performance under class imbalance.

For drift detection, which is formulated as a binary decision problem between in-distribution and drifting samples, we employ Normalized Accuracy (NA), False Detection Rate (FD), and Precision on Drifted Samples (PRED). These metrics capture complementary aspects of drift detection performance.

a) Normalized Accuracy (NA): Normalized Accuracy evaluates detection performance on both in-distribution and drifting samples. Assuming C training classes and drifted samples labeled as class $C+1$, NA is defined as

$$NA = \lambda_r AKS + (1-\lambda_r) AUS, \tag{9}$$

where

$$AKS = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C}(TP_i + FN_i)}, \quad AUS = \frac{TP_{C+1}}{TP_{C+1} + FN_{C+1}}. \tag{10}$$

Here, AKS and AUS denote accuracy on in-distribution and drifting samples, respectively, and $\lambda_r = 0.5$ balances their contributions.

b) False Detection Rate (FD): FD measures the proportion of drifting samples incorrectly classified as in-distribution:

$$FD = \frac{FN_{C+1}}{FN_{C+1} + TP_{C+1}}. \tag{11}$$

c) Precision on Drifted Samples (PRED): PRED quantifies the proportion of correctly detected drifting samples among all samples identified as drifting:

$$PRED = \frac{TP_{C+1}}{TP_{C+1} + FP_{C+1}}. \tag{12}$$

In summary, macro-level metrics evaluate overall classification capability, while NA, FD, and PRED focus on the robustness and effectiveness of drift detection. Table II summarizes the adopted metrics and their corresponding experimental scenarios.

5) Parameter Settings: For our proposed method, we adopt the AdamW optimizer with a learning rate of 0.0002 and a batch size of 128.

B. Closed-world Performance Comparison

Table III reports the macro-precision, macro-recall, and macro-F1 scores of all methods under different numbers of in-distribution classes. Results are averaged over multiple runs and reported as mean $\pm$ standard deviation.

To evaluate closed-world classification performance, we conduct experiments on the original training dataset, where 10% of the data are randomly selected for testing and the remainder for training. Under the closed-world assumption (Section III-A), all test samples belong to known classes observed during training.

As shown in Table III, FlowDiff consistently achieves the best performance across all class settings, demonstrating strong discriminative capability in closed-world scenarios. This advantage stems from its compact feature learning strategy, which reduces intra-class variability and enhances inter-class separability, combined with a LightGBM classifier that effectively captures nonlinear feature interactions in low-dimensional representations.

Most baseline methods exhibit competitive closed-world performance, while ICE-CP performs noticeably worse. This degradation is likely due to its joint training of the classifier and autoencoder, which can compromise in-distribution discrimination. In contrast, methods with decoupled representation learning and classification, such as FlowDiff, maintain more stable decision boundaries.

TABLE II: Evaluation Metrics and Experimental Scenarios

Metric	**Full Name**	**Experiment Type**	**Purpose**
Macro Precision	Macro-level Precision	Closed/Open World	Recognition capability
Macro Recall	Macro-level Recall	Closed/Open World	Coverage capability
Macro F1	Macro-level F1 Score	Closed/Open World	Overall performance
NA	Normalized Accuracy	Drift Detection	Overall identification
FD	False Detection Rate	Drift Detection	Drift recall capability
PRED	Precision on Drifted Samples	Drift Detection	Drift detection precision

TABLE III: Closed-world performance of different methods

Method	9 in-distribution classes			8 in-distribution classes			7 in-distribution classes		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
DIAI	0.94 ± 0.007	0.93 ± 0.002	0.93 ± 0.003	0.94 ± 0.008	0.92 ± 0.003	0.92 ± 0.004	0.93 ± 0.009	0.91 ± 0.002	0.91 ± 0.003
OpenMax	0.95 ± 0.007	0.92 ± 0.002	0.91 ± 0.003	0.94 ± 0.009	0.91 ± 0.004	0.90 ± 0.006	0.94 ± 0.008	0.90 ± 0.002	0.89 ± 0.005
ZTI	0.94 ± 0.012	0.92 ± 0.010	0.92 ± 0.011	0.93 ± 0.016	0.91 ± 0.013	0.90 ± 0.014	0.93 ± 0.011	0.91 ± 0.007	0.90 ± 0.010
CADE	0.95 ± 0.008	0.92 ± 0.001	0.91 ± 0.002	0.94 ± 0.004	0.91 ± 0.001	0.90 ± 0.002	0.93 ± 0.009	0.90 ± 0.002	0.89 ± 0.004
ICE-CP	0.84 ± 0.028	0.89 ± 0.000	0.85 ± 0.001	0.81 ± 0.001	0.87 ± 0.000	0.83 ± 0.001	0.79 ± 0.001	0.86 ± 0.000	0.81 ± 0.001
FlowDiff	**0.96 ± 0.002**	**0.94 ± 0.001**	**0.94 ± 0.001**	**0.95 ± 0.000**	**0.94 ± 0.001**	**0.93 ± 0.001**	**0.94 ± 0.003**	**0.93 ± 0.002**	**0.93 ± 0.002**

C. Drift Detection Performance Comparison

We next evaluate the ability of different methods to detect drifted samples. Table IV summarizes the detection results under varying numbers of OOD classes.

As shown in Table IV, FlowDiff consistently outperforms all baselines across all OOD settings. It achieves the highest NA and PRED scores while maintaining the lowest FD. For instance, with one OOD class, FlowDiff attains an NA of 0.95 and a PRED of 0.95, with an FD of only 0.04. This performance remains stable as the number of OOD classes increases, demonstrating strong robustness to concept drift.

The superior drift detection performance of FlowDiff can be attributed to its integrated design. The Compact Feature Module produces highly discriminative embeddings, while the conditional diffusion model captures class-conditional feature distributions and amplifies reconstruction discrepancies for drifted samples. The loss-adaptive thresholding strategy further enables precise separation between in-distribution and drifting traffic.

Baseline methods degrade noticeably as drift complexity increases, whereas FlowDiff maintains robust and superior performance across all OOD settings.

D. Open-world Recognition Performance Comparison

We evaluate overall recognition performance under the open-world setting, where OOD samples may appear during inference. In this setting, all OOD samples are treated as a single novel class, and the model is expected to both separate in-distribution and OOD samples and correctly classify in-distribution traffic. Details of the open-world assumption are provided in Section III-A.

As shown in Table V, FlowDiff achieves the best performance across all open-world configurations. Its F1-score remains stable as the number of OOD classes increases, demonstrating strong robustness under challenging open-world conditions. In contrast, baseline methods exhibit lower performance and degrade as OOD classes increase, whereas FlowDiff maintains consistently strong open-world recognition.

E. Ablation Study

1) Experimental setup: As introduced in Section III, the *CFM*, *In-Distribution Sample Classifier* and *CDM* are the key components of *FlowDiff* designed to handle concept drift. To evaluate the effectiveness of each component, we conduct four ablation experiments by removing them individually. The comparison models used are as follows: (1) the full model of FlowDiff (Full), (2) FlowDiff without the CFM (w/o R), (3) FlowDiff without the In-Distribution Sample Classifier (w/o C) and (4) FlowDiff without the CDM (w/o D). In this variant, the conditional diffusion module is removed, and drift detection is directly performed using the feature representations learned by the CFM. Specifically, a distance-based detection strategy is adopted, where the drift samples are identified according to the Euclidean distance between the feature vector of a sample and the centroid vector of its predicted class. For clearer comparison and demonstration of each components contribution, we only report the metrics from the closed-world classification and drift-sample detection experiments. The open-world experiment, which reflects the overall recognition capability of the model, is not included in this ablation. Moreover, we do not further group the results by the number of drift categories.

2) Evaluation results: Table VI presents the closed-world classification and drift detection metrics under different ablation settings. The table reports the average performance and standard deviation across multiple runs for each method, highlighting how the removal of each component affects the respective metrics.

TABLE IV: Drift detection performance of different methods

Method	1 OOD class			2 OOD classes			3 OOD classes		
	NA ↑	FD ↓	PRED ↑	NA ↑	FD ↓	PRED ↑	NA ↑	FD ↓	PRED ↑
DIAI	0.50 ± 0.042	0.93 ± 0.085	0.34 ± 0.316	0.48 ± 0.045	0.95 ± 0.091	0.29 ± 0.311	0.49 ± 0.054	0.91 ± 0.106	0.32 ± 0.380
OpenMax	0.63 ± 0.098	0.60 ± 0.197	0.39 ± 0.343	0.70 ± 0.080	0.46 ± 0.157	0.79 ± 0.213	0.67 ± 0.135	0.28 ± 0.285	0.62 ± 0.374
ZTI	0.77 ± 0.149	0.24 ± 0.243	0.14 ± 0.205	0.85 ± 0.112	0.12 ± 0.199	0.30 ± 0.218	0.79 ± 0.142	0.15 ± 0.142	0.37 ± 0.272
CADE	0.81 ± 0.099	0.24 ± 0.205	0.20 ± 0.087	0.81 ± 0.098	0.22 ± 0.220	0.37 ± 0.159	0.77 ± 0.152	0.28 ± 0.317	0.47 ± 0.205
ICE-CP	0.87 ± 0.114	0.15 ± 0.228	0.92 ± 0.082	0.88 ± 0.102	0.11 ± 0.203	0.97 ± 0.042	0.85 ± 0.115	0.15 ± 0.223	0.90 ± 0.189
FlowDiff	**0.95 ± 0.020**	**0.04 ± 0.038**	**0.95 ± 0.024**	**0.96 ± 0.005**	**0.01 ± 0.011**	**0.99 ± 0.012**	**0.93 ± 0.048**	**0.06 ± 0.091**	**0.98 ± 0.031**

Note: Red arrows indicate whether higher (↑) or lower (↓) values are preferred for each metric.

TABLE V: Open-world performance of different methods

Method	1 OOD class			2 OOD classes			3 OOD classes		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
DIAI	0.87 ± 0.033	0.84 ± 0.008	0.84 ± 0.013	0.83 ± 0.036	0.82 ± 0.010	0.80 ± 0.017	0.81 ± 0.047	0.79 ± 0.029	0.77 ± 0.022
OpenMax	0.81 ± 0.042	0.81 ± 0.040	0.78 ± 0.044	0.83 ± 0.073	0.83 ± 0.045	0.81 ± 0.055	0.66 ± 0.304	0.64 ± 0.300	0.58 ± 0.355
ZTI	0.80 ± 0.088	0.78 ± 0.093	0.75 ± 0.088	0.85 ± 0.078	0.83 ± 0.062	0.80 ± 0.068	0.78 ± 0.239	0.75 ± 0.211	0.73 ± 0.232
CADE	0.86 ± 0.032	0.84 ± 0.027	0.82 ± 0.027	0.87 ± 0.034	0.83 ± 0.034	0.81 ± 0.039	0.86 ± 0.040	0.82 ± 0.049	0.80 ± 0.049
ICE-CP	0.84 ± 0.010	0.88 ± 0.023	0.85 ± 0.018	0.83 ± 0.011	0.88 ± 0.023	0.84 ± 0.021	0.79 ± 0.033	0.85 ± 0.034	0.81 ± 0.034
FlowDiff	**0.96 ± 0.004**	**0.95 ± 0.005**	**0.94 ± 0.004**	**0.95 ± 0.002**	**0.94 ± 0.001**	**0.94 ± 0.001**	**0.94 ± 0.012**	**0.93 ± 0.015**	**0.93 ± 0.014**

TABLE VI: Ablation study results of *FlowDiff*

Method	Closed-world metrics			Drift detection mertics		
	Precision ↑	Recall ↑	F1 ↑	NA ↑	FD ↓	PRED ↑
Full	0.951 ± 0.005	0.937 ± 0.006	0.934 ± 0.007	0.950 ± 0.030	0.036 ± 0.057	0.968 ± 0.029
w/o R	0.949 ± 0.006	0.935 ± 0.007	0.932 ± 0.007	0.861 ± 0.115	0.206 ± 0.226	0.846 ± 0.209
w/o C	0.935 ± 0.011	0.922 ± 0.009	0.918 ± 0.009	–	–	–
w/o D	–	–	–	0.671 ± 0.149	0.513 ± 0.339	0.379 ± 0.283

Note: The meanings of the red arrows are consistent with the previous tables.– means this component does not affect the corresponding metric.

(1) Effect of the CFM. Removing this module (w/o R) leads to a noticeable degradation in drift detection performance (e.g., NA drops from 0.950 to 0.861, while FD increases from 0.036 to 0.206). This demonstrates that compact feature representation is essential for maintaining the separability between in-distribution and drift samples. Moreover, a slight decline in closed-world metrics (e.g., Precision and F1) can also be observed, indicating that the module not only enhances drift awareness but also contributes to more stable and discriminative feature spaces for classification.

(2) Effect of the In-Distribution Sample Classifier. When the In-Distribution Sample Classifier is removed (w/o C), the closed-world classification performance declines across all metrics (Precision, Recall, and F1). This component enables FlowDiff to better distinguish between known categories, ensuring reliable classification under stable conditions. Without it, the model struggles to form clear decision boundaries, even though drift detection remains unaffected.

(3) Effect of the CDM. Eliminating the CDM (w/o D) severely degrades drift detection performance, with NA dropping to 0.671 and PRED to 0.379. This confirms that the diffusion mechanism is crucial for modeling the underlying data distribution and capturing fine-grained variations caused by concept drift. The diffusion process provides probabilistic robustness, enabling the system to effectively detect and adapt to evolving data distributions.

V. Discussion

The proposed *FlowDiff* framework benefits from a decoupled and modular design that separates feature representation learning, drift detection, and in-distribution classification. This separation reduces task interference and enables the learning of compact and discriminative features, contributing to the strong closed-world classification performance observed in the experiments. Moreover, the adoption of class-conditional diffusion modeling provides an expressive characterization of feature distributions, yielding a sensitive signal for detecting distributional shifts.

The experimental results indicate that *FlowDiff* maintains stable performance across different numbers of in-distribution classes and drift configurations. This robustness can be attributed to enhanced feature separability and the class-level loss-adaptive detection strategy. In addition, by excluding explicit host identifiers and timestamp-related features, the framework focuses on intrinsic flow-level behaviors, which improves generalization and reduces overfitting to dataset-specific artifacts.

From a practical perspective, *FlowDiff* balances effectiveness and efficiency through the use of compact feature representations, a lightweight classifier, and reduced-dimensional diffusion modeling. These design choices ensure low computational overhead and make the framework suitable for real-world deployment. While the current implementation operates on aggregated flow-level features, future work may explore incorporating richer temporal or packet-level information to further capture complex traffic dynamics.

VI. Conclusion

In conclusion, We proposed FlowDiff, a novel framework for malicious traffic identification and drift detection in both closed-world and open-world scenarios. FlowDiff combines compact feature representation learning, CDM for high-fidelity reconstruction, and a LightGBM classifier for in-distribution recognition.

Experiments on real-world malicious traffic datasets show that FlowDiff outperforms existing methods in classification accuracy and drift detection, particularly for previously unseen traffic. The framework demonstrates strong robustness and generalization, making it suitable for dynamic network environments. In summary, FlowDiff provides an effective, scalable solution for real-time monitoring and detection of evolving malicious traffic, highlighting the potential of diffusion-based reconstruction in cybersecurity applications.

References

[1] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, "CADE: Detecting and explaining concept drift samples for security applications," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 2327–2344.

[2] M. A. Shyaa, N. F. Ibrahim, Z. Zainol, R. Abdullah, M. Anbar, and L. Alzubaidi, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," *Engineering Applications of Artificial Intelligence*, vol. 137, p. 109143, 2024.

[3] D. Jin, J. Xie, S. Chen, J. Yang, X. Liu, and W. Wang, "Zero-day traffic identification using one-dimension convolutional neural networks and auto encoder machine," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 559–563.

[4] X. Luo, C. Liu, G. Gou, G. Xiong, Z. Li, and B. Fang, "Identifying malicious traffic under concept drift based on intraclass consistency enhanced variational autoencoder," *Science China Information Sciences*, vol. 67, no. 8, p. 182302, 2024.

[5] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.

[6] D. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," *Advances in neural information processing systems*, vol. 34, pp. 21 696–21 707, 2021.

[7] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*. IEEE, 2019, pp. 1171–1179.

[8] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 633–642.

[9] Y. Wen, X. Han, W. Zuo, and W. Liu, "A knowledge distillation-driven lightweight cnn model for detecting malicious encrypted network traffic," in *2024 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2024, pp. 1–8.

[10] M. Ahmed, J. Chen, E. Akpaku, and A. Latif, "Birnn-sa: Context-aware malicious network traffic detection using self-attentive bidirectional rnns," *Computer Networks*, p. 111658, 2025.

[11] X. Han, S. Cui, J. Qin, S. Liu, B. Jiang, C. Dong, Z. Lu, and B. Liu, "Contramtd: An unsupervised malicious network traffic detection method based on contrastive learning," in *Proceedings of the ACM Web Conference 2024*, 2024, pp. 1680–1689.

[12] M. Ahmed, J. Chen, E. Akpaku, and R. N. A. Sosu, "Mtcr-ae: A multiscale temporal convolutional recurrent autoencoder for unsupervised malicious network traffic detection," *Computer Networks*, vol. 261, p. 111147, 2025.

[13] L. Yang, Y. Wang, L. Liu, J. Huang, J. Shi, S. Fu, and S. Guo, "unflows: An unsupervised construction scheme of flow spectrum for network traffic detection," *IEEE Transactions on Information Forensics and Security*, 2025.

[14] J. Ghadermazi, S. Hore, A. Shah, and N. D. Bastian, "Gtae-ids: Graph transformer-based autoencoder framework for real-time network intrusion detection," *IEEE Transactions on Information Forensics and Security*, 2025.

[15] J. Zhang, F. Li, F. Ye, and H. Wu, "Autonomous unknown-application filtering and labeling for dl-based traffic classifier update," in *IEEE INFOCOM 2020-IEEE conference on computer communications*. IEEE, 2020, pp. 397–405.

[16] M. H. Pathmaperuma, Y. Rahulamathavan, S. Dogan, and A. M. Kondoz, "Deep learning for encrypted traffic classification and unknown data detection," *Sensors*, vol. 22, no. 19, p. 7643, 2022.

[17] R. Tang, Z. Yang, Z. Li, W. Meng, H. Wang, Q. Li, Y. Sun, D. Pei, T. Wei, Y. Xu *et al.*, "Zerowall: Detecting zero-day web attacks through encoder-decoder recurrent neural networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2479–2488.

[18] Y. Chen, Z. Li, J. Shi, G. Gou, C. Liu, and G. Xiong, "Not afraid of the unseen: a siamese network based scheme for unknown traffic discovery," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–7.

[19] B. Tang, Y. Lu, Q. Li, Y. Bai, J. Yu, and X. Yu, "A diffusion model based on network intrusion detection method for industrial cyber-physical systems," *Sensors*, vol. 23, no. 3, p. 1141, 2023.

[20] M. A. Merzouk, E. Beurier, R. Yaich, N. Boulahia-Cuppens, F. Cuppens, and F. Khomh, "Diffusion-based adversarial purification for intrusion detection," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2025, pp. 351–370.

[21] S. Cai, Y. Zhao, J. Lyu, S. Wang, Y. Hu, M. Cheng, and G. Zhang, "Ddpdar: Network intrusion detection based on denoising diffusion probabilistic model and dual-attention residual network," *Neural Networks*, vol. 184, p. 107064, 2025.

[22] C. Yang, X. Xiao, G. Hu, Z. Ling, H. Li, and B. Zhang, "Dtpn: A diffusion-based traffic purification network for tor website fingerprinting," in *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 2025, pp. 642–650.

[23] Z. Sheng, D. Yuan, J. Ding, and Y. Li, "Unveiling the power of noise priors: Enhancing diffusion models for mobile traffic prediction," *arXiv preprint arXiv:2501.13794*, 2025.

[24] N. Sivaroopan, D. Bandara, C. Madarasingha, G. Jourjon, A. P. Jayasumana, and K. Thilakarathna, "Netdiffus: Network traffic generation by diffusion models through time-series imaging," *Computer Networks*, vol. 251, p. 110616, 2024.

[25] A. Bendale and T. E. Boult, "Towards open set deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1563–1572.