SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2024, Vol. 67, Iss. 8, 182302:1–182302:15
https://doi.org/10.1007/s11432-023-4010-4

Identifying malicious traffic under concept drift based on intraclass consistency enhanced variational autoencoder

Xiang LUO[1,2], Chang LIU[1,2], Gaopeng GOU[1,2], Gang XIONG[1,2*], Zhen LI[1,2] & Binxing FANG[3]

[1]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100085*, China;*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China;*
[3]*School of Computer Science and Technology, Harbin Institute of Technology* (*Shenzhen*)*, Shenzhen* 518055*, China*

Received 22 August 2023/Revised 16 February 2024/Accepted 11 April 2024/Published online 23 July 2024

Abstract Accurate identification of malicious traffic is crucial for implementing effective defense countermeasures and has led to extensive research efforts. However, the continuously evolving techniques employed by adversaries have introduced the issues of concept drift, which significantly affects the performance of existing methods. To tackle this challenge, some researchers have focused on improving the separability of malicious traffic representation and designing drift detectors to reduce the number of false positives. Nevertheless, these methods often overlook the importance of enhancing the generalization and intraclass consistency in the representation. Additionally, the detectors are not sufficiently sensitive to the variations among different malicious traffic classes, which results in poor performance and limited robustness. In this paper, we propose intraclass consistency enhanced variational autoencoder with Class-Perception detector (ICE-CP) to identify malicious traffic under concept drift. It comprises two key modules during training: intraclass consistency enhanced (ICE) representation learning and Class-Perception (CP) detector construction. In the first module, we employ a variational autoencoder (VAE) in conjunction with Kullback-Leibler (KL)-divergence and cross-entropy loss to model the distribution of each input malicious traffic flow. This approach simultaneously enhances the generalization, interclass consistency, and intraclass differences in the learned representation. Consequently, we obtain a compact representation and a trained classifier for non-drifting malicious traffic. In the second module, we design the CP detector, which generates a centroid and threshold for each malicious traffic class separately based on the learned representation, depicting the boundaries between drifting and non-drifting malicious traffic. During testing, we utilize the trained classifier to predict malicious traffic classes for the testing samples. Then, we use the CP detector to detect the potential drifting samples using the centroid and threshold defined for each class. We evaluate ICE-CP and some advanced methods on various real-world malicious traffic datasets. The results show that our method outperforms others in identifying malicious traffic and detecting potential drifting samples, demonstrating outstanding robustness among different concept drift settings.

Keywords concept drift, malicious traffic identification, variational autoencoder, intrusion detection, cyberspace security

1 Introduction

Malicious traffic has been implicated in numerous security incidents, such as data leakage and extortion, underscoring the critical need for effective detection methods. Machine learning-based models offer promise in accurately identifying malicious traffic by harnessing the insights derived from historical data and statistical characteristics [1]. A significant challenge arises from the assumption made by these models that the distribution of testing data remains consistent with the training data (also called "closed-world" assumption) [2]. This poses a vulnerability to concept drift, whereby adversaries persistently update their techniques, resulting in shifts in the distribution of testing data from the original training data. Consequently, the dynamic nature of malicious traffic can lead to the failure of these models [3].

* Corresponding author (email: xionggang@iie.ac.cn)

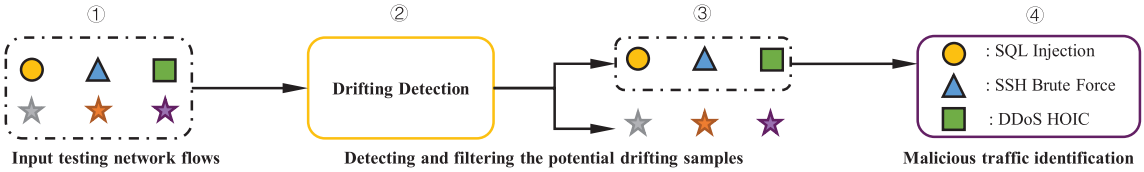

Figure 1 (Color online) Workflow of malicious traffic identification under concept drift.

Many researchers have devoted their efforts to mitigating the impact of concept drift on malicious traffic identification. They aim to achieve two primary objectives. On the one hand, the model should effectively detect the potential drifting samples to minimize the false positives that may arise. On the other hand, the model should accurately identify non-drifting malicious traffic, which is the fundamental requirement. By addressing both objectives, the model can uphold its accuracy in non-drifting malicious traffic identification while mitigating the detrimental effects of concept drift. The overall workflow is illustrated in Figure 1.

To achieve these objectives, researchers have focused on two main aspects: improving the separability of representation and constructing drift detectors. In terms of improving separability, techniques, such as DNN [4] and CNN [5, 6], have been utilized to learn discriminative representations of malicious traffic. These techniques ensure more accurate identification of non-drifting malicious traffic and enhance the ability to differentiate drifting samples. In terms of drift detector construction, existing methods can be categorized into reconstruction-based methods [7, 8] and prototype-based methods [2, 9, 10]. Reconstruction-based methods identify drifting samples based on high reconstruction loss. Prototype-based methods compare testing samples with prototypes of each class of non-drifting malicious traffic, considering samples that deviate significantly from all prototypes as potential drifting samples. Techniques such as contrastive learning [11] and siamese network [12] are used to improve the quality of prototypes. However, these current methods still face several common issues. (1) Lack of generalization: When improving the separability of representation, the discussed methods aim to learn specific features of malicious traffic rather than its overall distribution. This focus may cause distorted representations and less robust performance across various experimental settings. (2) Intraclass dispersion: These methods often overlook the importance of ensuring similar distributions among samples from the same malicious traffic class in the latent space, leading to many outliers within class representations. The outliers often produce false positives during drifting detection. (3) Insensitivity to class variations: These methods do not adequately account for the disparities in data distribution across different malicious traffic classes, which hampers their ability to set distinct decision boundaries for accurately detecting drifting samples.

In this paper, we propose a novel prototype-based approach called intraclass consistency enhanced variational autoencoder with Class-Perception detector (ICE-CP) to address the challenges encountered in identifying malicious traffic under concept drift. Our method comprises two critical modules: intraclass consistency enhanced (ICE) representation learning and Class-Perception (CP) detector construction. In the first module, we utilize a variational autoencoder (VAE) combined with Kullback-Leibler (KL)-divergence and cross-entropy loss to learn the distribution of the input network flows in the latent space. Specifically, the VAE framework ensures the generalization and robustness of the representation, and KL-divergence loss enhances the intraclass consistency of the learned representation as it encourages samples from the same malicious traffic class to exhibit similar distributions in the latent space. Moreover, the cross-entropy loss emphasizes the interclass differences of the representation and results in a trained classifier for non-drifting malicious traffic. In the second module, we develop a CP detector to generate the unique centroid and threshold for each class of non-drifting malicious traffic by levering the learned representation, which considers the class variation of the data distribution and contributes to finding the precise boundary between drifting and non-drifting malicious traffic. During testing, ICE-CP captures the representation of each testing sample and feeds it into the trained classifier and detector, outputting its predicted label and whether its distance to the centroid exceeds the corresponding threshold. By jointly evaluating the outputs, ICE-CP determines whether the testing sample is a potential drifting sample or belongs to one of the malicious traffic classes in the training data.

In summary, this paper has four main contributions.

• We propose ICE-CP to identify malicious traffic under concept drift accurately. By utilizing ICE representation learning and CP detector construction, we obtain a generalized and compact representation and find the precise boundary between drifting and non-drifting malicious traffic. These contribute to

Table 1 Summary of the related work about malicious traffic identification under concept drift

Category of identification methods		Design for malicious traffic	Improving representation separability			Detector construction	
			Interclass difference	Generalized representation	Intraclass consistency	Drift detection	Class-Perception detector
Closed -world	Rule-based [13, 14]	✓	×	×	×	×	×
	ML-based [5, 15]	✓	×	×	×	×	×
Under concept drift	Confidence-based [4, 16]	×	×	×	×	✓	×
	Gradient-based [17, 18]	×	×	×	×	✓	×
	Recon-based [7, 8]	✓	×	×	×	✓	×
	Prototype-based [2, 9, 10]	✓	✓	×	×	✓	×
	ICE-CP	✓	✓	✓	✓	✓	✓

the stable and prominent performance of our method.

• We employ KL-divergence loss and cross-entropy loss when training VAE in ICE representation learning to enhance the intraclass consistency within the representation of each malicious traffic class. This improves the precision of our method when detecting potential drifting samples.

• We construct a Class-Perception detector based on the learned representation. This mechanism captures the class variation and effectively establishes the boundary between drifting and non-drifting samples, contributing to the prominent performance of our method in detecting drifting samples.

• Evaluated on real-world malicious traffic datasets and different settings, ICE-CP outperforms other advanced methods in comparison and cross-dataset experiments. The ablation study and detector comparison experiment show the soundness of each module in ICE-CP.

2 Related work

Our research focuses on malicious traffic identification under concept drift. To ensure the integrity of related research, we introduce closed-world malicious traffic identification and traffic identification under concept drift. We summarize the related studies in Table 1 [2, 4, 5, 7–10, 13–18].

2.1 Closed-world malicious traffic identification

Early research on malicious traffic identification can be categorized as signature-based methods [13, 14] and machine learning-based methods [1]. Signature-based methods rely on information such as port numbers, headers, payloads, and host behavior logs to create preconfigured rules for identifying malicious traffic [19]. Notably, these methods require maintaining a rule database and high overhead. Consequently, researchers have begun to explore machine learning-based methods.

Machine learning-based methods are usually based on statistical features and require feature design and selection. Aburomman et al. [15] proposed a novel ensemble construction method that integrates multiple models through majority voting. They use the 41 features produced by the KDD99 dataset, such as the duration length of the connection and the number of data bytes from source to destination. Selvakumar et al. [20] focused on reducing the number of features to improve the detection rate and apply a firefly meta-heuristic approach for feature selection. However, such methods need much expert knowledge to design manual features, which takes time and effort. The automatic feature extraction ability and versatility of deep learning supplement the deficiencies of developing manual features. Wang et al. [6] proposed the HAST-IDS system by using 1D-CNN and RNN to learn the low-level spatial features and high-level sequential features of malware traffic. Javaid et al. [21] employed a self-taught learning technique in their model, improving precision and recall.

Although these methods mentioned above perform well on their corresponding dataset, they work under the closed-world assumption vulnerable to concept drift.

2.2 Traffic identification under concept drift

To address the limitations of closed-world methods, researchers have started considering concept drift and designing strategies that not only identify non-drifting malicious traffic but also detect and filter the potential drifting traffic. As far as our knowledge goes, proposed methods are pursued from two perspectives, as shown in Table 1, improving representation separability and detector construction.

On the perspective of improving the separability of representation, researchers aim to force samples within the same class to gather compactly while keeping samples from different classes far apart, thus reserving extra space for drifting samples. For instance, Zhao et al. [9] designed distance-based cross-entropy loss (DCE) and metric regularization (MR) to enhance class discriminability of the malware traffic. Similarly, Yang et al. [2] employed a contrastive autoencoder to increase distances between samples from different classes while reducing distances between samples within the same class. Unfortunately, these methods primarily focus on making the malicious traffic data discriminative, neglecting the utilization of data consistency within each class of malicious traffic. Consequently, the obtained representation for each malicious traffic class is not compact, and outliers in the representation may be located close to drifting samples in the latent space during testing, hindering the identification performance of these methods. Moreover, these methods aim to model specific representations for each input malicious network flow, which may only work under limited settings, leading to a lack of robustness.

From the perspective of constructing drift detectors, related methods can be divided into four categories based on the elements they use: confidence-based methods, gradient-based methods, reconstruction-based methods, and prototype-based methods. The first two methods have been tested on app traffic, and the latter two methods have been tested on malicious traffic. Specifically, confidence-based methods are based on the hypothesis that a drifting sample will trigger relatively lower softmax confidence than the known one. For instance, Zhang et al. [16] proposed an autonomous model updating framework which directly filters the app traffic that triggers a low confidence score as the unknown app. Moreover, Pathmaperuma et al. [4] also exploited the probability distribution of the confidence score to discover the drifting app traffic. However, it has been shown that the softmax results are always over-confidence, which means the drifting traffic may still have a high confidence score. Thus, it is difficult to decide whether the app traffic is drifting solely depending on the confidence score.

Gradient-based methods use the back-propagation gradient to detect drifting app traffic [17,18]. Yang et al. [17] proposed GradDB to recognize zero-day applications by the magnitude of the back-propagation gradient. Based on Yang's research, Xia et al. [18] proposed GMAF which adopts addictive angular margin loss (ArcFace) during training to obtain highly discriminative features, improving the performance of detecting potential drifting app traffic. Nevertheless, these gradient-based methods mainly concentrate on detecting the potential drifting app traffic, failing to improve the accuracy of non-drifting app traffic identification.

Reconstruction-based methods are inspired by the observation that models perform poorly in reconstructing drifting samples that are not seen during training. For example, Jin et al. [7] proposed zero-day traffic identification (ZTI) to detect zero-day traffic. It utilizes the reconstruction loss produced by the autoencoder to determine whether the incoming traffic belongs to one of the known classes or zero-day traffic. Similarly, Tang et al. [8] proposed ZeroWall to effectively detect zero-day Web attacks, which trains a self-translation machine and uses an encoder-decoder recurrent neural network to capture benign requests' syntax and semantic patterns. Unfortunately, the reconstruction error produced by generative models often fails to distinguish between training data and some apparent out-of-distribution data [22], leading to the misclassification of drifting traffic and causing false positives.

Prototype-based methods aim to utilize prototypes to provide a concise representation for each malicious traffic class. During testing, the testing data is compared to each prototype to determine whether it has drifted. For instance, Yang et al. [2] utilized centroids as the prototypes. Working with a distance-based algorithm, they consider samples far from all the prototypes as potential drifting samples. Chen et al. [10] proposed UNSEEN, which leverages a Siamese neural network to achieve unknown malware traffic detection. UNSEEN selects a set of samples from each malicious traffic class as prototypes and compares the testing samples with each prototype to determine whether they have drifted or belong to one of the malicious traffic classes in the non-drifting samples. However, these detectors are not sensitive to the class variation of the distribution of malicious traffic, as they set the same threshold for all classes during testing, ignoring the various distributions within different malicious traffic classes.

3 Methodology

In this section, we present a comprehensive description of our proposed ICE-CP. Our approach comprises two key modules: ICE representation learning and CP detector construction. The overall workflow of ICE-CP is illustrated in Figure 2. To comprehensively understand our approach, we first introduce the

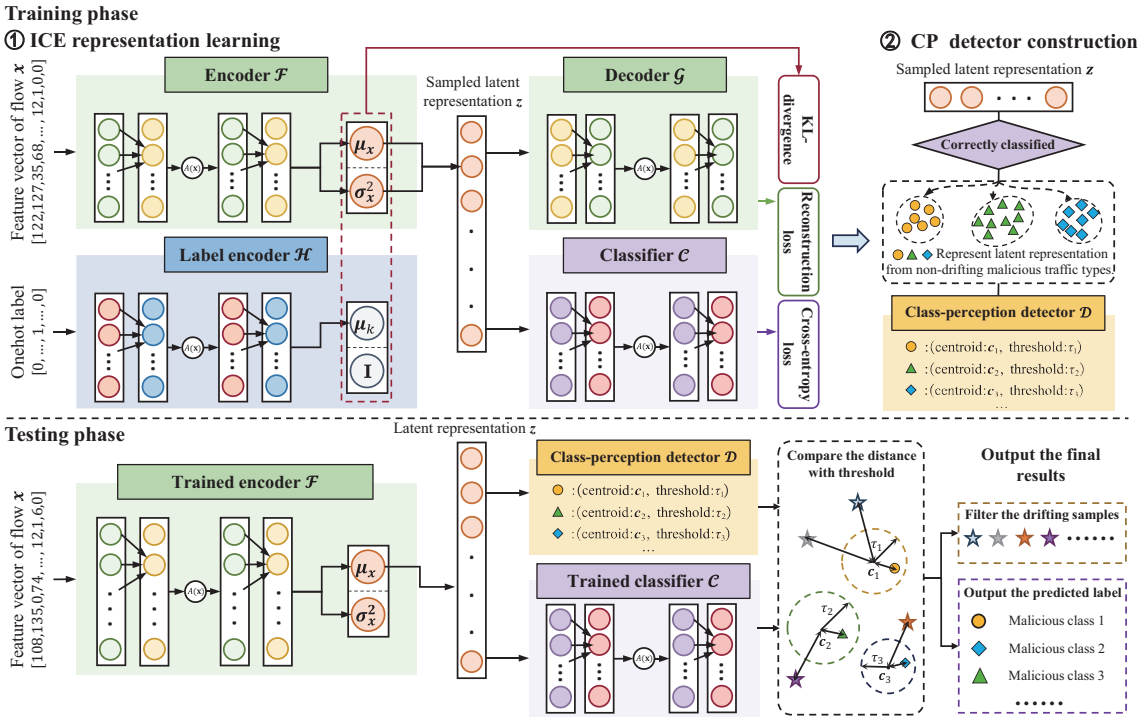

Figure 2 (Color online) Overall workflow of ICE-CP.

problem definition; then we outline the workflow in the subsequent subsections.

3.1 Problem definition

In this section, we formulate the problem in this paper. We aim to design a malicious traffic identification model that identifies malicious traffic and detects the drifting samples.

Given a malicious traffic dataset $\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N_{\text{train}}}$ for training, where $\boldsymbol{x}_i \in \mathbb{R}^D$ represents a malicious network flow and $y_i \in \{1, \ldots, C\}$ is the corresponding malicious class. At the testing phase, the testing dataset $\mathcal{D}_{\text{test}} = \{(\boldsymbol{x}_k, y_k)\}_{k=1}^{N_{\text{test}}}$ consists of malicious network flows among $C+U$ classes, where U is the number of malicious classes that are only available during testing, i.e., $y_k \in \{1, \ldots, C, \ldots, C+U\}$. Network flows from these U classes are regarded as drifting traffic. In our approach, we aim at learning an identification model $M : \boldsymbol{x} \to \mathcal{Y} = \{1, \ldots, C, \text{drifting}\}$, where the malicious network flow $\boldsymbol{x}$ that is labeled as drifting follows the different data distribution from the training data.

3.2 ICE representation learning

In this stage, our goal is to obtain a compact and discriminative representation from the training malicious traffic data, which contributes to distinguishing the drifting samples from the non-drifting samples. We achieve this by using a variational autoencoder combined with KL-divergence and cross-entropy loss. This approach allows us to learn the distribution of each class of malicious traffic and enhance the consistency within each class and the difference between classes.

Specifically, the input is the feature vector of the malicious network flow from the training data, defined as $\boldsymbol{x} \in \mathbb{R}^D$. D is the dimension of vector $\boldsymbol{x}$. To begin with, encoder $\mathcal{F}$ is used to generate posterior distribution in latent space for the input feature vector representing the malicious network flow. The posterior distribution is defined as $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{x}}, \boldsymbol{\sigma}_{\boldsymbol{x}}^2)$, where $\boldsymbol{\phi}$ is the parameter of encoder $\mathcal{F}$, k is the malicious traffic class of $\boldsymbol{x}$, and $\boldsymbol{z}$ is the latent representation of $\boldsymbol{x}$. As shown in Figure 2, $\boldsymbol{x}$ is fed into $\mathcal{F}$ and $\boldsymbol{\mu}_{\boldsymbol{x}}$, $\boldsymbol{\sigma}_{\boldsymbol{x}}^2$ are the output of $\mathcal{F}$, and thus the parameters of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k)$ are determined. Formally, let f denote the backbone DNN consists of l layers, and the output of DNN is $\boldsymbol{x}_l = f(\boldsymbol{x})$. Then $\boldsymbol{x}_l$ will be fed to a single linear layer to obtain $\boldsymbol{\mu}_{\boldsymbol{x}}$, which is formularized as $\boldsymbol{\mu}_{\boldsymbol{x}} = \texttt{Linear}(\boldsymbol{x}_l)$. On the other hand, to ensure that $\boldsymbol{\sigma}_{\boldsymbol{x}}^2$ is non-negative, we feed $\boldsymbol{x}_l$ to a Softplus layer to obtain $\boldsymbol{\sigma}_{\boldsymbol{x}}^2$, which is formularized as $\boldsymbol{\sigma}_{\boldsymbol{x}}^2 = \texttt{Softplus}(\boldsymbol{x}_l)$. After obtaining $\boldsymbol{\mu}_{\boldsymbol{x}}$ and $\boldsymbol{\sigma}_{\boldsymbol{x}}^2$, the latent representation $\boldsymbol{z}$ is sampled from $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k)$,

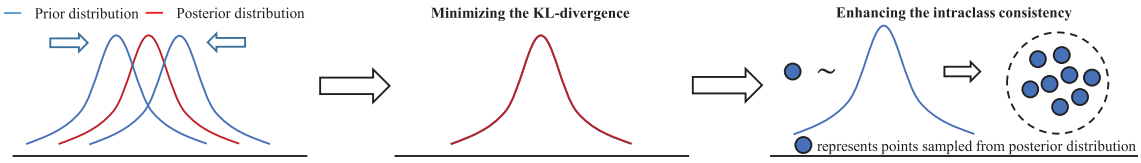

Figure 3 (Color online) Impact of KL-divergence loss.

which is calculated as

$$\boldsymbol{z} = \boldsymbol{\mu}_{\boldsymbol{x}} + \boldsymbol{\sigma}_{\boldsymbol{x}} \odot \epsilon, \tag{1}$$

where $\epsilon \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and $\odot$ is the element-wise product.

Meanwhile, label encoder $\mathcal{H}$ maps the one-hot label vector of $\boldsymbol{x}$ to the latent space, providing the mean vector $\boldsymbol{\mu}_k$ of the prior distribution of $\boldsymbol{x}$, defined as $p_{\boldsymbol{\theta}}^{(k)}=\mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_k, \boldsymbol{I})$, where k represents the malicious traffic class of $\boldsymbol{x}$ and $\boldsymbol{\theta}$ represents the parameter of $\mathcal{H}$. As $p_{\boldsymbol{\theta}}^{(k)}$ contains the label information of $\boldsymbol{x}$, it represents the overall distribution of the malicious traffic class k in latent space.

After the latent representation $\boldsymbol{z}$ is sampled, it will be fed into decoder $\mathcal{G}$ and classifier $\mathcal{C}$. Decoder $\mathcal{G}$ aims to use latent representation $\boldsymbol{z}$ to reconstruct the original input $\boldsymbol{x}$ as similar as possible. Formally, it is expressed as $\hat{\boldsymbol{x}} = g(\boldsymbol{z})$, where g represents another backbone DNN and $\hat{\boldsymbol{x}}$ is the reconstructed vector of $\boldsymbol{x}$. Classifier $\mathcal{C}$ is a softmax layer that produces the probability distribution over the malicious traffic classes in the training data and outputs the predicted label $\hat{y}$ of $\boldsymbol{x}$. It is formularized as $\hat{y} = \mathcal{C}(\boldsymbol{z})$.

In summary, the loss function during training consists of three parts shown as

$$\mathcal{L} = \mathcal{L}_{\mathrm{KL}} + \mathcal{L}_r + \mathcal{L}_{\mathrm{ce}}. \tag{2}$$

$\mathcal{L}_{\mathrm{KL}}$ represents the difference between prior distribution $p_{\boldsymbol{\theta}}^{(k)}$ and posterior distribution $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k)$. By minimizing $\mathcal{L}_{\mathrm{KL}}$, $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k)$ is forced to approximate $p_{\boldsymbol{\theta}}^{(k)}$, which means flows from the same class of malicious traffic follow the same prior distribution. Thus, the consistency of data distribution in each malicious traffic class is enhanced, making the latent representation of each malicious traffic class more aggregated in latent space, as shown in Figure 3. Let J be the dimension of z. $\mathcal{L}_{\mathrm{KL}}$ is calculated as

$$\begin{aligned} \mathcal{L}_{\mathrm{KL}} &= -D_{\mathrm{KL}}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}, k) \| p_{\boldsymbol{\theta}}^{(k)}(\boldsymbol{z})) \\ &= \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - (\mu_j - \mu_j^{(k)})^2 - \sigma_j^2). \end{aligned} \tag{3}$$

$\mathcal{L}_r$ represents the reconstruction error between the reconstructed vector $\hat{\boldsymbol{x}}$ and the original input vector $\boldsymbol{x}$. By minimizing $\mathcal{L}_r$, we ensure that the latent representation $\boldsymbol{z}$ maintains rich information of $\boldsymbol{x}$, which guides $\boldsymbol{z}$ to learn boosted features, and thus improves its representational ability. $\mathcal{L}_r$ is calculated as

$$\mathcal{L}_r = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2 . \tag{4}$$

$\mathcal{L}_{\mathrm{ce}}$ is the cross-entropy loss between the predicted label and the ground truth label, aiming to obtain the discriminative representation $\boldsymbol{z}$ by enhancing its interclass difference, as shown in Figure 4. Given the probability distribution q_i over malicious traffic class for the i-th input vector $\boldsymbol{x}_i$ $(i = 1, \ldots, N)$, $\mathcal{L}_{\mathrm{ce}}$ is calculated as

$$\mathcal{L}_{\mathrm{ce}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} I(y_i = c) \log q_i(c), \tag{5}$$

where $I(y_i = c) = 1$ if y_i is c, else 0, and $q_i(c)$ is the probability of malicious class c for the input $\boldsymbol{x}_i$.

3.3 CP detector construction

In this stage, we focus on constructing a drift detector that retains the distribution information of each class of malicious traffic in the latent space. The detector is designed to operate in a Class-Perception manner, utilizing unique thresholds assigned to each class of malicious traffic, as shown in Figure 5. After ICE representation learning, we obtain the trained encoder, which maps the drifting network flows significantly far from the centroids of non-drifting samples across all malicious traffic classes during

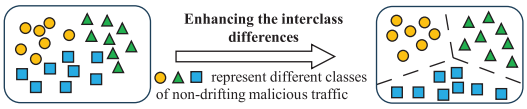

Figure 4 (Color online) Impact of cross-entropy loss.

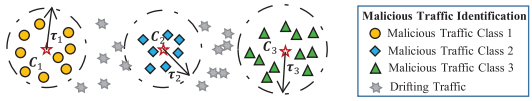

Figure 5 (Color online) Motivation of Class-Perception detector.

Algorithm 1 Testing procedure

Input:
The centroid $\boldsymbol{c}_i$ and distance threshold τ_i for each malicious traffic class i, $i = 1, \ldots, C$;
Testing network flow $\boldsymbol{x}_k$, $k = 1, \ldots, K$, K is the total number of testing samples;
Trained encoder $\mathcal{F}$ and classifier $\mathcal{C}$;
Constructed detector $\mathcal{D}$;
Output:
Label for each testing network flow $\boldsymbol{x}_k$;

for each $k \in [1, K]$ **do**
  $\boldsymbol{\mu}_{\boldsymbol{x}_k}, \boldsymbol{\sigma}_{\boldsymbol{x}_k} = \mathcal{F}(\boldsymbol{x}_k)$;
  for each $i \in [1, C]$ **do**
    $d_i^{(k)} = \left\|\boldsymbol{\mu}_{\boldsymbol{x}_k} - \boldsymbol{c}_i\right\|_2$;
  end for
  $d^{(k)} = \min(d_i^{(k)})$, $i = 1, \ldots, N$;
  $i_k = \text{argmin}(d_i^{(k)})$;
  Predicted label $\hat{y} = \text{argmax}(\mathcal{C}(\boldsymbol{z}_k))$;
  if $d^{(k)} > \tau_{i_k}$ **then**
    Predict $\boldsymbol{x}^{(k)}$ as drifting;
  else
    Predict $\boldsymbol{x}^{(k)}$ with label $\hat{y}$;
  end if
end for

testing. Therefore, the critical task in this stage is to determine the centroid and reject threshold for each class of malicious traffic.

Specifically, assume that the training data comprises C malicious traffic classes. For each malicious class i $(i = 1, \ldots, C)$, we obtain the latent representation $\boldsymbol{z}_{i,j}$ $(j = 1, \ldots, n_i)$ of each correctly classified training network flows $\boldsymbol{x}_{i,j}$. We calculate the centroid $\boldsymbol{c}_i$ for each malicious traffic class i by taking the mean value of the latent representation of this malicious traffic class shown as

$$\boldsymbol{c}_i = \text{mean}_i(\boldsymbol{z}_{i,j}) = \frac{1}{n_i}\sum_{j=1}^{n_i}\boldsymbol{z}_{i,j}. \tag{6}$$

As different classes of malicious traffic may have a different data distribution of latent representation, which leads to different tightness levels, we set unique threshold τ_i for each malicious traffic class i according to the mean and standard deviation of $d_i^{(j)} = \left\|\boldsymbol{z}_{i,j} - \boldsymbol{c}_i\right\|_2$, as $d_i^{(j)}$ reflects the dispersion of the i-th malicious traffic class in latent space. τ_i is calculated as

$$\tau_i = \text{mean}_i(d_i^{(j)}) + \alpha \times \text{std}_i(d_i^{(j)}), \tag{7}$$

where α is a constant. Using $\boldsymbol{c}_i$ and τ_i, the detector is able to determine whether the malicious traffic is an outlier of each malicious traffic class in latent space during testing, then determines the testing sample as a drifting sample if it is an outlier for all of the C classes.

3.4 Testing

The workflow of the testing procedure is available in the testing phase of Figure 2. During testing, we use the trained model and constructed detector to jointly determine whether the testing malicious network flow is from one of the malicious traffic classes in the training data or is the drifting sample.

The overall testing procedure is shown in Algorithm 1. Given the testing malicious network flow $\boldsymbol{x}_k$ $(k = 1, \ldots, N_{\text{test}})$, we use the trained encoder $\mathcal{F}$ to map it into latent space. Different from the latent representation learning stage during training, we take $\boldsymbol{\mu}_k$ as the corresponding latent representation of $\boldsymbol{x}_k$ instead of $\boldsymbol{z}_k = \boldsymbol{\mu}_{\boldsymbol{x}_k} + \boldsymbol{\sigma}_{\boldsymbol{x}_k} \odot \boldsymbol{\epsilon}$, because we do not need sample option in this phase. Then we calculate the Euclidean distance between the $\boldsymbol{x}_k$ and each centroid of malicious traffic class in latent space, i.e.,

Table 2 Statistics information of CIC-IDS2018 dataset

Class	Num of flows	Class	Num of flows	Class	Num of flows
DDOS attack-HOIC	686012	SSH-BruteForce	187589	DoS attacks-Slowloris	10990
Dos attacks-Hulk	461912	DoS attacks-SlowHTTPTest	139890	DDOS attack-LOIC-UDP	1370
Bot	286191	Infiltration	93063	Brute Force-Web	611
FTP-BruteForce	193360	DoS attacks-GoldenEye	41508	Brute Force-XSS	230

$d_i^{(k)} = \|\boldsymbol{\mu}_{\boldsymbol{x}_k} - \boldsymbol{c}_i\|_2$. We find the minimum value of $d_i^{(k)}$ and the corresponding malicious traffic class i_k, defined as $d^{(k)}$. If $d^{(k)}$ exceeds the threshold τ_{i_k}, we determine $\boldsymbol{x}_k$ as the drifting sample. Else, we output the predicted malicious traffic class of $\boldsymbol{x}$.

The advantage of setting different thresholds for each malicious traffic class is that it considers the variability in data distribution between different classes of malicious traffic in the latent space. This approach helps to identify the precise decision boundary of drifting samples, leading to more accurate detection results.

4 Performance evaluation

4.1 Experimental settings

4.1.1 *Datasets*

We conduct our comparison experiments on the CIC-IDS2018 dataset. This dataset [23] results from a controlled attacks campaign conducted by the Canadian Institute for Cybersecurity. For our evaluation, we select 12 classes of malicious traffic from this dataset. The detailed information is shown in Table 2.

4.1.2 *Data prepossess*

The features of this dataset are extracted by CICFlowMeter[1)], resulting in a total of 79 original features for each network flow. Following the previous work CADE [2], categorical features such as "destination port" and "network protocol" are encoded with one-hot encoding. Specifically, the "destination port" is classified into three categories based on its frequency in the training dataset: ports that appeared more than 10000 times are classified as high-frequency ports, those that appear more than 1000 times are classified as medium-frequency ports, and rest are classified as low-frequency ports. The original "destination port" feature is then replaced with its corresponding 3-dimensional one-hot feature vector based on its category. The "network protocol" is encoded in the same way as the "destination port", and these two categorical features are replaced with two 3-dimensional one-hot vectors. The other 77 statistical features are normalized using a MinMaxScaler between 0 and 1. As a result, each network flow is represented by an 83-dimensional feature vector.

4.1.3 *Comparative methods*

We compare the performance of ICE-CP with the following relevant methods, which cover all categories of related work in malicious traffic under concept drift as shown in Table 1.

• DNN for in-app identification (DIAI) [4] rejects low-confidence samples as drifting samples. This is typical work of confidence-based methods.

• OpenMax [24] is a more modified confidence-based method which modifies the activation vectors of the backbone DNN by using the properties of Weibull distribution.

• ZTI [7] is a typical work of reconstruction-based methods that rejects samples with high reconstruction error as potential drifting samples during testing.

• GMAF [18] is a typical work of gradient-based methods that rejects drifting samples by the magnitude of the back-propagation gradient.

• CADE [2] is a prototype-based method which employs contrastive learning and median absolute deviation (MAD) distance in its workflow.

1) https://github.com/ahlashkari/CICFlowMeter.

4.1.4 *Evaluation metrics*

We use five metrics to fully evaluate ICE-CP and other comparative methods, including normalized accuracy (NA), false detection rate on drifting samples (FD), precision on drifting samples ($\mathrm{PRE_D}$), misclassification rate on non-drifting samples (MIS), and outliers rate in representation (Outliers). We introduce these five metrics in the following passages.

NA [25] is used to evaluate the overall identification performance of each method under concept drift. Suppose the training data contain C malicious traffic classes, and the drifting network flows are labeled as $C+1$. Let TP_i, TN_i, FP_i, FN_i respectively denote the true positive, true negative, false positive, and false negative for the ith class of traffic where $i \in \{1, \ldots, C, C+1\}$. NA is defined as

$$\mathrm{NA} = \lambda_r \mathrm{AKS} + (1-\lambda_r)\mathrm{AUS}, \tag{8}$$

where AKS represents the ratio of correctly identified non-drifting malicious traffic, while AUS represents the ratio of correctly detected drifting samples. We calculate AUS and AKS separately instead of using traditional accuracy because in real-world scenarios, the number of drifting samples is typically small compared to non-drifting samples. Therefore, even if all drifting samples are not correctly detected, the detection method may still achieve high accuracy. We set $\lambda_r = 0.5$ in our evaluation to show the equal importance of these two terms.

MIS [25] is used to evaluate the performance of identifying non-drifting samples. It measures the proportion of non-drifting malicious traffic misclassified as other malicious traffic classes in the training dataset, demonstrating the ability of each method to model the malicious traffic. MIS is defined as

$$\mathrm{MIS} = \frac{\sum_{i=1}^{C} \mathrm{FN}_i - \mathrm{FP}_{C+1}}{\sum_{i=1}^{C}(\mathrm{TP}_i + \mathrm{FN}_i) - \mathrm{FP}_{C+1}}. \tag{9}$$

FD and $\mathrm{PRE_D}$ are used to evaluate the performance on detecting potential drifting samples on different perspectives. On the one hand, FD measures the proportion of drifting samples that are not correctly detected, reflecting the method's ability to mitigate concept drift. On the other hand, $\mathrm{PRE_D}$ measures the ratio of true drifting samples detected by the model, indicating the model's drift detection capability from another perspective. FD and $\mathrm{PRE_D}$ are calculated as follows:

$$\mathrm{FD} = \frac{\mathrm{FN}_{C+1}}{\mathrm{FN}_{C+1} + \mathrm{TP}_{C+1}},\ \mathrm{PRE_D} = \frac{\mathrm{TP}_{\mathrm{C+1}}}{\mathrm{FP}_{\mathrm{C+1}} + \mathrm{TP}_{\mathrm{C+1}}}. \tag{10}$$

Outliers are used to evaluate the compactness of the representation learned from the malicious traffic. We use the same notation as Subsection 3.3 to introduce this metric. The outlier is defined as the sample from class i whose distance to the centroid c_i is greater than $Q_1 + 1.5\mathrm{IQR}$, where Q_1 represents the first quartile of $d_i^{(j)}$ and IQR represents the interquartile range of $d_i^{(j)}$. Outliers is calculated as

$$\mathrm{Outliers} = \frac{\mathrm{Number\ of\ Outliers}}{\mathrm{Number\ of\ Samples}}. \tag{11}$$

4.1.5 *Parameters settings*

In our proposed method, we use the Adam optimizer with a learning rate of 0.001 and fix the batch size to 512 inspired from CADE [2]. We set the backbone DNN for encoder $\mathcal{F}$ as an MLP with the architecture of 83-64-32-C in our experiments, where C is the number of malicious traffic classes in training data. The decoder $\mathcal{G}$ parameters are symmetric to the encoder $\mathcal{F}$. The activation function for each hidden layer is the ReLU function. The training of the networks is conducted over 100 epochs. During testing, we set $\alpha = 4$ for the drift detector.

4.2 Comparison experiment

In this subsection, the CIC-IDS2018 dataset is used to construct various concept drift settings for testing (introduced in Subsection 4.2.1). We compare our method with other 5 related advanced methods (mentioned in Subsection 4.1.3) under these settings in Subsection 4.2.2.

Table 3 NA, FD (lower is better), and PRE_D results of ICE-CP and other comparative methods on CIC-IDS2018 dataset[a)]

Method	3 classes in drifting samples			4 classes in drifting samples			5 classes in drifting samples		
	NA	FD	PRE_D	NA	FD	PRE_D	NA	FD	PRE_D
DIAI	0.61±0.14	0.70±0.25	0.37±0.29	0.57±0.11	0.75±0.18	0.38±0.28	0.57±0.12	0.75±0.20	0.41±0.29
OpenMax	0.50±0.15	0.53±0.32	0.08±0.05	0.53±0.14	0.46±0.32	0.12±0.06	0.45±0.09	0.62±0.22	0.11±0.06
ZTI	0.56±0.07	0.85±0.15	0.55±0.37	0.67±0.05	0.64±0.11	0.78±0.18	0.68±0.04	0.62±0.10	0.82±0.15
GMAF	0.57±0.09	0.77±0.16	0.40±0.32	0.58±0.06	0.75±0.10	0.47±0.26	0.61±0.05	0.70±0.09	0.56±0.26
CADE	0.94±0.03	0.01±0.03	0.52±0.12	0.95±0.03	0.01±0.02	0.58±0.11	0.94±0.03	0.02±0.05	0.63±0.11
ICE-CP	**0.98±0.01**	**0.00±0.00**	**0.74±0.16**	**0.98±0.02**	**0.00±0.00**	**0.78±0.14**	**0.98±0.02**	**0.01±0.02**	**0.83±0.12**

a) The bold text in the table indicates the optimal performance.

4.2.1 *Experimental setup*

We use the CIC-IDS2018 [23] dataset to create multiple concept drift scenarios. As there are 12 classes of malicious traffic in this dataset and 9 of them have more than 10000 flows, we randomly select 6 classes of these 9 classes as non-drifting malicious traffic to build a training set, while the samples from the remaining classes are regarded as drifting samples. In the training phase, 80% of data from each class of non-drifting malicious traffic is utilized for model training, with 10% reserved for model validation. In the testing phase, the remaining 10% of data was combined with the drifting samples for testing.

To thoroughly assess the model's identification performance, we set the number of malicious traffic classes in the drifting samples as 3, 4, and 5 in the experiments. To mitigate the effect of class selection, we select the malicious traffic classes 5 times under each number setting of classes in drifting samples, resulting in 15 combinations of malicious traffic classes. For each combination, we perform 10 repeated experiments and average the results to ensure data stability and reliability.

For metrics such as NA, FD, and PRE_D which are specific to the number of classes in drifting samples, we take the average results of the 5 combinations for each number setting. For metrics such as MIS and Outliers that are not specific to the number of classes in drifting samples, we take the average results of the overall 15 combinations.

We replicated all comparative methods and conducted parameter tuning to demonstrate their optimal performance in experiments. It is worth mentioning that the original data input formats for the comparative methods OpenMax are not in the form of statistical features. To ensure fairness, we used the same DNN model structure for these methods as mentioned in Subsection 4.1.5, following CADE's successful results with this structure on this dataset.

4.2.2 *Evaluation results*

Table 3 illustrates the NA, FD, PRE_D results of ICE and all the other comparative methods under 3 number settings of malicious traffic classes in drifting samples, and Figure 6 shows the average MIS and Outliers results of all the experimental settings, respectively. We can draw the following conclusion.

(1) ICE-CP consistently achieves the best performance, and outperforms the other methods. From Table 3, ICE-CP consistently achieves NA around 0.98 for all the experimental settings. This demonstrates the outstanding performance of ICE-CP in mitigating concept drift when identifying malicious traffic. Even compared to the state-of-the-art method CADE, our approach achieves higher NA in all experimental settings, with an average NA improvement of 3%, showing that ICE-CP fully enjoys the benefits of the ICE representation learning and the Class-Perception detector construction. In addition, ICE-CP has the slightest standard deviation on all metrics under all experimental settings, showing that the representation learned by the VAE framework is robust, contributing to stable performance.

(2) ICE-CP improves the interclass difference of the malicious traffic representation effectively. As shown in Figure 6, ICE-CP achieves an average MIS below 1×10^{-4} above all experimental settings. This indicates that our proposed method eliminates the misclassification among different classes of malicious traffic in the non-drifting samples, showing that the representation it learned has prominent separability. This is attributed to the enhancement of intraclass consistency and interclass differences brought by ICE representation learning. Conversely, methods such as DIAI, OpenMax, ZTI, and GMAF fail to eliminate MIS, indicating their failure to fully capture the interclass difference between each class of malicious traffic and their inability to establish precise decision boundaries for each class. As a result, they exhibit poor performance under concept drift.

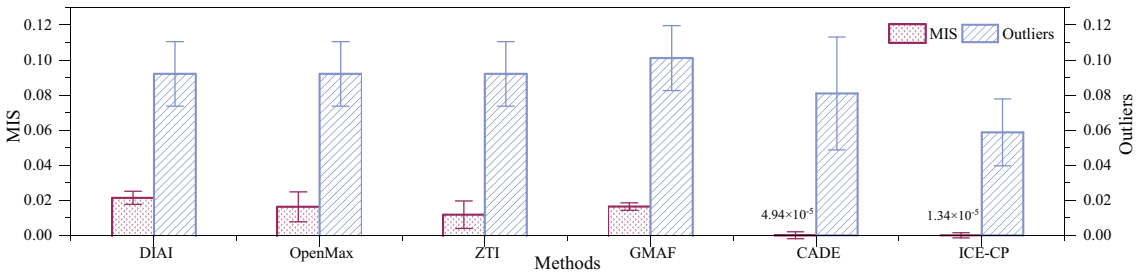

Figure 6 (Color online) MIS and Outliers results of ICE-CP and other comparative methods on CIC-IDS2018 dataset. Lower values are better for both metrics.

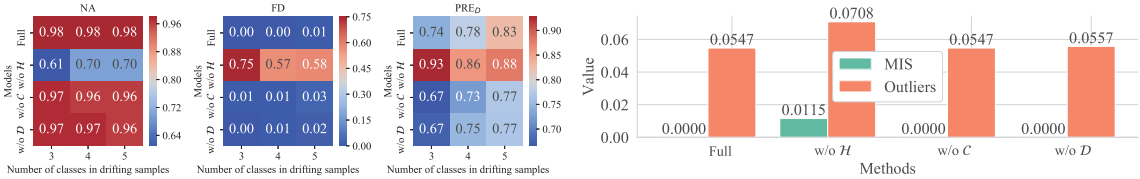

Figure 7 (Color online) Ablation study results of ICE-CP on CIC-IDS2018 dataset. (a) NA, FD (lower is better), and PRE_D results; (b) MIS and Outliers results (lower values are better for both metrics).

(3) ICE-CP effectively enhances the intraclass consistency of malicious traffic representation, leading to compact representation for the malicious traffic. The results in Figure 6 demonstrate that ICE-CP achieves an average outlier ratio of 0.06 in modeling known malicious traffic, which is lower than other methods. This indicates that the malicious traffic representations learned by ICE-CP are more compact in the latent space. This is attributed to the introduction of KL-divergence loss during the training of ICE-CP, which encourages inputs from the same class to follow similar distributions. Due to the more compact representations, our method generates fewer false positives when using distance-based detectors to detect drifting samples, resulting in higher precision for drifting samples, as shown in Table 3.

(4) The Class-Perception detector of ICE-CP demonstrates remarkable drift detection capabilities for the drifting samples. ICE-CP achieves the minimum FD compared to other methods, with FD values consistently below 0.02. This signifies that ICE-CP effectively reduces false positives caused by drifting samples, thereby mitigating the negative impact of concept drift. Moreover, the Class-Perception detector locates a more precise boundary between drifting and non-drifting samples, as it considers the difference in data distribution between each malicious traffic class. As a result, ICE-CP achieves the highest precision during drifting detection, as shown in Table 3.

4.3 Ablation study

As introduced in Section 3, the label encoder, classifier, and decoder are the critical components of ICE-CP designed to address concept drift. To demonstrate their effectiveness, we perform three separate ablation analysis experiments on each component. The comparison models used are as follows: (1) Full model of ICE-CP (Full), (2) ICE-CP without the label encoder (w/o $\mathcal{H}$), (3) ICE-CP without the classifier (w/o $\mathcal{C}$), and (4) ICE-CP without the decoder (w/o $\mathcal{D}$).

The ablation study is conducted on the CIC-IDS2018 dataset and we follow the same experimental settings as the comparison experiments. Figure 7(a) displays the results of NA, FD, and PRE_D of all the models under 3 experimental settings, respectively. Figure 7(b) demonstrates the average MIS and Outliers of each compared model in the ablation study. We can draw the following conclusions.

(1) ICE-CP with the complete components outperforms other comparison models under all experimental settings, showing the soundness of its components. When we remove any other component of ICE-CP, the NA decreases range from 2% to 31% and the FD increases range from 1% to 63%, as shown in Figure 7(a). Although the model without the label encoder shows an increase in PRE_D for detecting drifting samples, this apparent improvement is driven by the fact that a large number of drifting samples are undetected, which diminishes the practical significance of this improvement.

(2) From the results of ICE-CP without label encoder, we can find that the label encoder effectively improves the performance of detecting drifting samples. This can be attributed to the fact that the

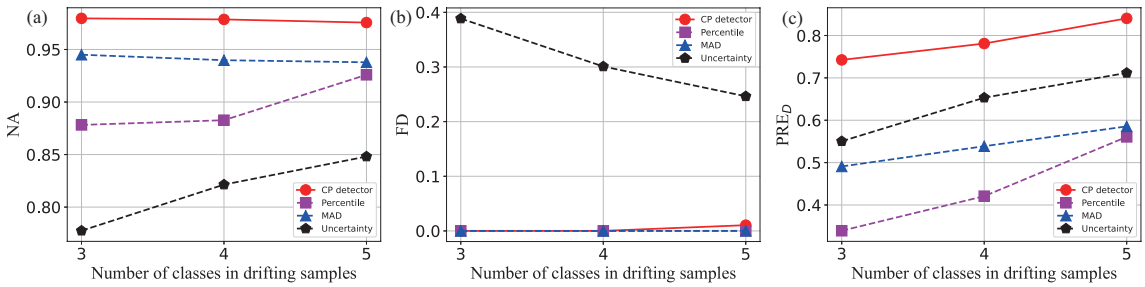

Figure 8 (Color online) Results of each detector on CIC-IDS2018 dataset. (a) NA; (b) FD (lower is better); (c) PRE_D.

introduction of the label encoder leads to samples from the same malicious traffic class following a similar distribution in the latent space, resulting in more compact representations with reduced distances between samples from the same class. This observation finds support in Figure 7(b), where the representation contains more outliers when removing the label encoder. Consequently, the performance of the detector is adversely affected, as evidenced by a significant increase in the FD.

(3) From the results of ICE-CP without the classifier module and without the decoder module, we can observe that these two modules improve the model's ability to identify drifting samples. On the one hand, the results in Figure 7(a) indicate that removing the classifier and decoder leads to a decrease in PRE_D. On the other hand, the FD increases when removing these two modules, as shown in Figure 7(a). We attribute these improvements to two reasons. Firstly, the introduction of the classifier increases the interclass differences in the representation of the non-drifting representation. Secondly, the reconstruction mechanism of the decoder module stores more boosted information in the learned representations, thus enhancing the effectiveness of the representation.

4.4 Comparison of different detectors

The drift detector plays a crucial role in identifying malicious traffic under concept drift, as the drifting samples can cause serious consequences. In this subsection, we compare our detection method with the following comparative Class-Perception detectors on the CIC-IDS2018 dataset under the same experimental settings and malicious traffic class combinations as in Subsection 4.2.1 to show the priority of our CP detector.

• Percentiles is the most trivial detection method. It chooses the 95th percentile of the distance to the centroid as the threshold for each class of malicious traffic.

• MAD is adopted by CADE [2], aiming to estimate the data distribution within each malicious traffic class i by calculating MAD_i.

• Uncertainty is proposed by an advanced open-set recognition method GMVAE [26], which is the ratio between the distance to the nearest centroid to the average distance to all other centroids.

As the change of detector does not influence the results of representation learning, we employ NA, FD, and PRE_D to evaluate the performance of each detector. We conduct the experiments on the CIC-IDS2018 dataset with the same experimental settings as mentioned in Subsection 4.2. Figure 8 presents the NA, FD, and PRE_D results for each detector under different experimental settings on the CIC-IDS2018 dataset. We draw the following conclusion.

(1) The overall identification performance of our CP detector surpasses that of other detectors under various concept drift experiment settings. This suggests that the CP detector we designed is the most suitable choice for our ICE representation learning module. Figure 8(a) demonstrates that our proposed method has a 2% to 20% improvement on NA over other detection methods under various concept drift settings.

(2) Our CP detector shows a marked improvement in the precision of detecting drifting traffic. On average, the precision is improved by 31% and 25% respectively as shown in Figure 8(c). This improvement can be primarily attributed to the class-specific threshold with distribution information precisely determining the boundary between drifting and non-drifting samples. In contrast, the thresholds set by the Percentile and MAD methods are sensitive to outliers in the training data of malicious traffic.

(3) The distribution information of the representation contributes to the superior performance in detecting drifting samples. Across all experimental settings, only uncertainty performs poorly on FD, as depicted in Figure 8(b). This observation indicates that the distribution information of each malicious

Table 4 Statistics information of DAPT2020 and CIC-DDoS2019 dataset

CIC-DDoS2019				DAPT2020			
Class	Num of flows	Class	Num of flows	Class	Num of flows	Class	Num of flows
UDP	7001800	NTP	1202642	Network Scan	7742	SQL Injection	84
MSSQL	10309945	SMMP	5159870	Backdoor	20	CSRF	7
NetBIOS	7750776	UDP-Lag	368334	Account Discovery	2408	Malware Download	2
WebDDoS	439	DNS	5071011	Directory Bruteforce	9970	Privilege Escalation	13
LDAP	4095052	SSDP	2610611	Web Vulnerability Scan	2574	Command Injection	12
Portmap	186960	Syn	6473789	Account Bruteforce	141	Data Exfiltration	6

traffic class is crucial for drifting detection. In contrast, uncertainty fails to consider it, as it simply compares the distance ratios of samples to different centroids.

4.5 Cross-dataset evaluation

In this subsection, we assess the identification capability of our model when confronted with drifting samples generated by other malicious traffic datasets. Specifically, we introduce two additional datasets for experimentation, namely the CIC-DDoS2019 dataset and the DAPT2020 dataset. These datasets encompass malicious traffic classes that differ from those contained within the CIC-IDS2018 dataset, thereby enabling them to serve as drift samples for the CIC-IDS2018 dataset. Conversely, the data contained within the CIC-IDS2018 dataset can also be considered drifting samples for these two datasets. Both datasets have 12 classes of malicious traffic available. The detailed statistic information of these two datasets is shown in Table 4.

4.5.1 *Experimental setup*

In the first step, we train models on CIC-IDS2018 dataset (D_1) and evaluate on CIC-DDoS2019 (D_2) and DAPT2020 (D_3) datasets. We select 9 classes of malicious traffic in the CIC-IDS2018 dataset for training, each of which has more than 10000 flows, namely SSH-BruteForce, DoS attacks-Hulk, DoS attacks-Slowloris, DoS attacks-GoldenEye, DDoS attacks-HOIC, and DoS attacks-SlowHTTPTest, respectively. For each malicious traffic class, we still choose 80% data for training, 10% for validation, and 10% for testing. After that, we add CIC-DDoS2019 and DAPT2020 datasets to the testing dataset respectively to construct two mixed datasets for evaluation. We randomly select 3000 flows from these two datasets respectively and add them to their corresponding testing set.

In the second step, we train models on D_2 and D_3, then evaluate the models on D_1. During the construction of the training set, to ensure sufficient data volume across different classes, we select malicious traffic data classes from D_2 and D_3 with a data volume exceeding 5000. This selection process yields 13 classes for the training set. We employ the same way as mentioned in the first step for data segmentation and construct the mixed testing set.

After constructing the testing datasets, we conduct experiments and evaluate the results under the same settings as in Subsection 4.2.

4.5.2 *Evaluation results*

The results of NA, FD, and $\mathrm{PRE_D}$ of ICE-CP and other comparative methods on the three cross-dataset settings are shown in Table 5. The overall MIS and Outliers results of these methods in cross-dataset evaluation are depicted in Figure 9. We can draw the following conclusion.

(1) ICE-CP achieves the best results on all metrics, outperforming other comparative methods under all cross-dataset settings. As shown in Table 5, ICE-CP achieves the highest NA and the lowest FD and MIS on all cross-dataset settings. Combining the results from Table 3, it is observed that by enhancing the interclass difference and intraclass consistency in the representation, along with designing an appropriate Class-Perception detector, we significantly improve the model's identification capability of malicious traffic under concept drift. Furthermore, our model outperforms other comparative methods across different cross-dataset settings, exhibiting reasonable standard deviations in various performance metrics and demonstrating robustness.

(2) ICE-CP demonstrates a superior capability in enhancing the differentiation between classes and ensuring consistency within classes in the representations learned from malicious traffic. The increasing

Table 5 NA, FD (the lower is better), and PRE_D results of cross-dataset evaluation on ICE-CP and each compared methods[a)]

Method	$D_1 \rightarrow D_2$			$D_1 \rightarrow D_3$			$D_2 + D_3 \rightarrow D_1$		
	NA	FD	PRE_D	NA	FD	PRE_D	NA	FD	PRE_D
DIAI	0.41±0.04	1.00±0.00	0.00±0.00	0.40±0.04	1.00±0.00	0.00±0.00	0.50±0.04	0.95±0.07	0.28±0.23
OpenMax	0.32±0.00	1.00±0.00	0.00±0.00	0.32±0.00	1.00±0.00	0.00±0.00	0.37±0.00	1.00±0.02	0.60±0.01
ZTI	0.93±0.07	**0.00±0.00**	0.84±0.15	0.98±0.00	**0.00±0.00**	0.90±0.00	0.49±0.00	0.99±0.00	0.61±0.04
GMAF	0.43±0.39	0.60±0.42	0.43±0.39	0.58±0.24	0.50±0.50	0.25±0.25	0.43±0.08	0.66±0.10	0.21±0.06
CADE	0.95±0.01	**0.00±0.00**	0.77±0.04	0.95±0.01	**0.00±0.00**	0.78±0.04	0.87±0.01	**0.00±0.00**	0.61±0.02
ICE-CP	**0.98±0.00**	**0.00±0.00**	**0.91±0.01**	**0.98±0.00**	**0.00±0.00**	**0.91±0.01**	**0.97±0.00**	**0.00±0.00**	**0.89±0.01**

a) The left side of " → " represents the dataset used for training, and the right side represents the dataset used as drifting samples. The bold text in the table indicates the optimal performance.

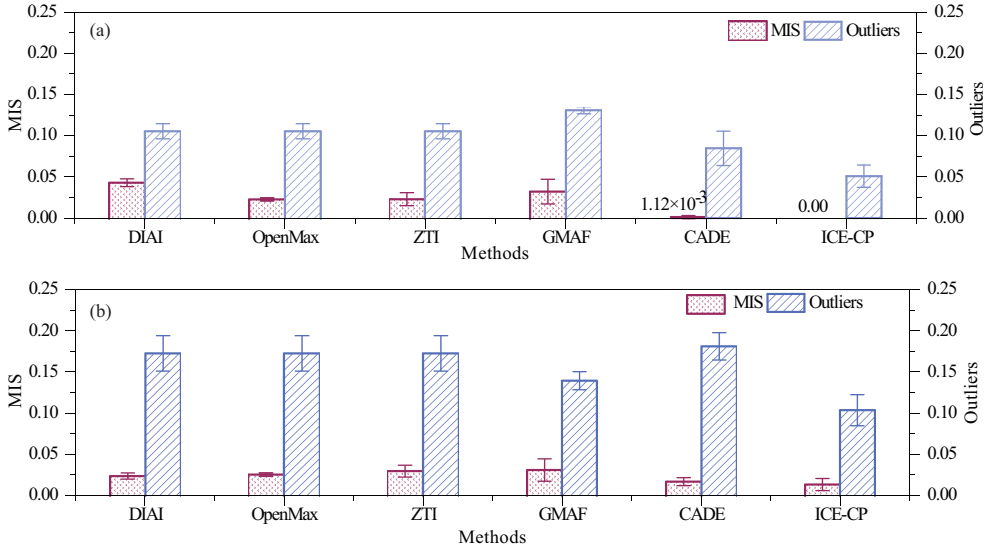

Figure 9 (Color online) MIS and Outliers results in cross-dataset evaluation. Lower values are better for both metrics. (a) Train on D_1; (b) train on $D_2 + D_3$.

number of malicious traffic classes in the training data in cross-dataset evaluation poses additional challenges to modeling such data. Notably, Figure 9 reveals an increase in the MIS results of the methods compared to those in Figure 6; yet ICE-CP achieves the lowest MIS, highlighting its exceptional ability to amplify interclass differences in the learned representations. Moreover, Figure 9 indicates that ICE-CP produces the lowest proportion of outliers across all cross-dataset settings, showing its prominent ability to enhance intraclass consistency in the learned representations.

(3) ICE-CP shows remarkable identification performance on identifying drifting samples produced by other datasets. As shown in Tabel 5, ICE-CP achieves the highest PRE_D and the minimum FD under all cross-dataset settings. On the other hand, DIAI and OpenMax still perform poorly, indicating the limitations of relying solely on confidence scores under concept drift as previously observed in Subsection 4.2. Moreover, from Table 5 and Figure 9, we find that although the reconstruction-based method ZTI performs well when the model is trained on CIC-IDS2018, it fails when the model is trained on CIC-DDoS2019 and DAPT2020. This indicates that the reconstruction-based approach lacks robustness across various cross-dataset settings.

5 Conclusion

In this paper, we propose ICE-CP to identify malicious traffic under concept drift. It comprises two key modules during training: ICE representation learning and Class-Perception detector construction. In the first module, we utilize a variational autoencoder to model the input malicious network flow distribution. Combining with KL-divergence loss and cross-entropy loss, we obtain a compact representation and a trained classifier for non-drifting malicious traffic. In the second module, we develop a Class-Perception detector that generates unique centroid and threshold for each malicious traffic class. During testing, ICE-

CP jointly identifies the testing samples based on the results of the classifier and detector. Experimental results demonstrate the superior identification accuracy and robustness of ICE-CP across various real-world datasets and malicious traffic classes, outperforming other existing methods under different concept drift settings.

Acknowledgements This work was supported by National Key Research and Development Program of China (Grant No. 2021YFB3101400).

References

1 Lin K, Xu X, Xiao F. MFFusion: a multi-level features fusion model for malicious traffic detection based on deep learning. Comput Networks, 2022, 202: 108658
2 Yang L, Guo W, Hao Q, et al. CADE: detecting and explaining concept drift samples for security applications. In: Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), 2021. 2327–2344
3 Gama J, Žliobaitė I, Bifet A, et al. A survey on concept drift adaptation. ACM Comput Surv, 2014, 46: 1–37
4 Pathmaperuma M H, Rahulamathavan Y, Dogan S, et al. Deep learning for encrypted traffic classification and unknown data detection. Sensors, 2022, 22: 7643
5 Wang W, Zhu M, Zeng X, et al. Malware traffic classification using convolutional neural network for representation learning. In: Proceedings of International Conference on Information Networking (ICOIN), 2017. 712–717
6 Wang W, Sheng Y, Wang J, et al. HAST-IDS: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access, 2017, 6: 1792–1806
7 Jin D, Xie J, Chen S, et al. Zero-day traffic identification using one-dimension convolutional neural networks and auto encoder machine. In: Proceedings of IFIP Networking Conference (Networking), 2020. 559–563
8 Tang R, Yang Z, Li Z, et al. ZeroWall: detecting zero-day web attacks through encoder-decoder recurrent neural networks. In: Proceedings of IEEE Conference on Computer Communications, 2020. 2479–2488
9 Zhao L, Cai L, Yu A, et al. Prototype-based malware traffic classification with novelty detection. In: Proceedings of the 21st International Conference on Information and Communications Security, Beijing, 2020. 3–17
10 Chen Y, Li Z, Shi J, et al. Not afraid of the unseen: a Siamese network based scheme for unknown traffic discovery. In: Proceedings of IEEE Symposium on Computers and Communications (ISCC), 2020. 1–7
11 Hadsell R, Chopra S, LeCun Y. Dimensionality reduction by learning an invariant mapping. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), 2006. 1735–1742
12 Bromley J, Guyon I, LeCun Y, et al. Signature verification using a "Siamese" time delay neural network. In: Proceedings of Advances in Neural Information Processing Systems, 1993. 6
13 Jamshed M A, Lee J, Moon S, et al. Kargus: a highly-scalable software-based intrusion detection system. In: Proceedings of the ACM Conference on Computer and Communications Security, 2012. 317–328
14 Nam J, Jamshed M, Choi B, et al. Haetae: scaling the performance of network intrusion detection with many-core processors. In: Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses, Kyoto, 2015. 89–110
15 Aburomman A A, Reaz M B I. A novel SVM-kNN-PSO ensemble method for intrusion detection system. Appl Soft Computing, 2016, 38: 360–372
16 Zhang J, Li F, Ye F, et al. Autonomous unknown-application filtering and labeling for dl-based traffic classifier update. In: Proceedings of IEEE Conference on Computer Communications, 2020. 397–405
17 Yang L, Finamore A, Jun F, et al. Deep learning and zero-day traffic classification: lessons learned from a commercial-grade dataset. IEEE Trans Netw Serv Manage, 2021, 18: 4103–4118
18 Xia Y, Xiong G, Li Z, et al. GMAF: a novel gradient-based model with ArcFace for network traffic classification. In: Proceedings of IEEE 23rd International Conference on High Performance Computing & Communications; 7th International Conference on Data Science & Systems; 19th International Conference on Smart City; 7th International Conference on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), 2021. 291–300
19 Hwang R H, Peng M C, Huang C W. Detecting IOT malicious traffic based on autoencoder and convolutional neural network. In: Proceedings of IEEE Globecom Workshops (GC Wkshps), 2019. 1–6
20 Selvakumar B, Muneeswaran K. Firefly algorithm based feature selection for network intrusion detection. Comput Secur, 2019, 81: 148–155
21 Javaid A, Niyaz Q, Sun W, et al. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), 2016. 21–26
22 Xiao Z, Yan Q, Amit Y. Likelihood regret: an out-of-distribution detection score for variational auto-encoder. In: Proceedings of the 34th International Conference on Neural Information Processing Systems, 2020. 20685–20696
23 Sharafaldin I, Lashkari A H, Ghorbani A A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of International Conference on Information Systems Security and Privacy, 2018
24 Bendale A, Boult T E. Towards open set deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 1563–1572
25 Júnior M P R, de Souza R M, Werneck R O, et al. Nearest neighbors distance ratio open-set classifier. Mach Learn, 2017, 106: 359–386
26 Cao A, Luo Y, Klabjan D. Open-set recognition with Gaussian mixture variational autoencoders. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 6877–6884