Abnormal Traffic Detection: Traffic Feature Extraction and DAE-GAN with Efficient Data Augmentation

Zecheng Li*, Shengyuan Chen*, Hongshu Dai*, Dunyuan Xu*, Chengkang Chu+, and Bin Xiao*

*The Hong Kong Polytechnic University,
Email:{zecheng.li, shengyuan.chen, francis.dai, dunyuan.xu}@connect.polyu.hk, csbxiao@comp.polyu.edu.hk
+Huawei Technologies Co Ltd Singapore,
Email: Chu.Cheng.Kang@huawei.com

***Abstract*—Abnormal traffic detection is the core component of the network intrusion detection system. Although semi-supervised methods can detect zero-day attack traffic, previous work suffers from high false alarms because the trained model is simply based on normal traffic. In this paper, we propose an accurate abnormal traffic detection method using pseudo anomaly, consisting of an efficient feature extraction framework and a novel Denoise AutoEncoder-Generative Adversarial Network (DAE-GAN) model. The feature extraction framework adopts an innovative packet window scheme to extract spatial and temporal features from traffic flows. The DAE-GAN model has multiple DAEs to achieve efficient data augmentation and generate high-quality pseudo anomalies. The pseudo anomalies are obtained by adding noise on normal traffic and enhanced by adversarial learning in DAE-GAN. Our semi-supervised detection method, exploiting both normal data and generated pseudo anomalies, achieves a precision of 98.6% on the NSL-KDD dataset and 98.5% on the UNSW-NB15 dataset. Compared with the state-of-the-art, the detection precision and recall under different user behaviors are significantly improved. The evaluation on four attack datasets shows that our method has a high flow-wise precision of over 99% and a high recall of 60.6%.**

***Index Terms*—Abnormal traffic detection, Generative Adversarial Networks, Adversarial learning, Deep learning, DNN.**

I. Introduction

Over the past two decades, with the development of communication technology, traffic identification techniques have emerged and evolved to monitor network status and provide users with better network services. One of the main issues in traffic identification is abnormal traffic detection. Abnormal traffic detection is important to many applications, such as intrusion detection systems, quality of service (QoS) control, and resource usage management. Identifying traffic flows in networks can not only improve the efficiency of the network resources but also maintain the security of local devices. However, abnormal traffic detection is challenged by the diverse and changing pattern of traffic and encryption technology. Existing abnormal traffic detection methods can be categorized as either misuse-based or anomaly-based.

Misuse-based (also referred to as signature-based) abnormal traffic detection, is to encode the traffic pattern of known attacks into signatures by experts. During diagnosis, a signature matching process is conducted to detect the presence of same attack traffic. To avoid costly human effort, signatures can be constructed using rule synthesis techniques [1]. Such rule matching-based methods can detect known attacks at high precision, high sensitivity, and high speed [2]. However, maintaining an updated list of signatures is still costly, requiring data labeling and signature generating. Without signatures, misuse-based methods cannot detect zero-day attacks. Although incremental learning has made it possible to detect newly discovered attacks [3], the period between the emergence of new attacks and construction of new signatures is too long for a device vulnerably exposed to attackers.

Anomaly-based abnormal traffic detection, on the other hand, can detect new attacks because it takes anomalies as deviations from a normal pattern distribution. The normal pattern is learned by training on a large amount of normal data, extracting statistical and time sequence features of the given normal data. One requirement is to maintain the latest anomaly-free traffic data.

Though being popular for the capability of detecting unseen attacks, anomaly-based methods perform differently depending on the algorithm they adopt. Reconstruction-based, clustering-based, and one-class classification are three dominant anomaly-based detection methods. All of them do not need supervised information and are trained on normal data. Though capable of detecting novel anomalies, the lack of anomaly pattern in training data entails low detection recall and high false alarm. To deal with this issue, an intuitive solution is to generate pseudo anomalies by sampling in the feature space. However, random sampling in all data spaces is computationally expensive and impractical, especially in high dimensional data spaces. Efficient data augmentation algorithm is needed when leveraging pseudo anomalies. Specifically, pseudo anomaly should be abnormal traffic generated manually rather than the real abnormal traffic (e.g., malware traffic) collected from the Internet.

An accurate and scalable abnormal traffic detection model based on pseudo anomaly should have the following proper-

ties:

- **Semi-Supervised Learning.** Preparing labeled data is not required before training. All data are simply obtained from normal traffic. The model can detect unseen attacks by training on only normal data.
- **Computationally Efficient.** Traffic is processed and analyzed in the real time to quickly identify anomalies with low latency.
- **Effective in High-dimensional Spaces.** An efficient pseudo anomaly generation method is needed even in high-dimensional data spaces.

To meet above property requirements, we propose Denoise AutoEncoder-Generative Adversarial Network (DAE-GAN) for abnormal traffic detection. The DAE-GAN is composed of multiple denoising autoencoders and a discriminator. Multiple DAEs constitute a pseudo anomaly (PA) generator to produce effective pseudo anomalies. The discriminator is trained on anomalies generated by the PA generator, and learns to distinguish abnormal traffic from normal ones on the Internet. Unlike most previous studies that focus on detecting anomalies using only normal data, our approach combines the idea of pseudo anomaly and adversarial learning. The overall high-level framework is a GAN-like structure composed of an efficient PA generator and a binary discriminator. In this way, we transform the anomaly detection problem into a binary classification problem. The sensitivity of the discriminator is further improved through adversarial learning.

For traffic data preprocessing, we introduce an innovative real-time feature extraction framework. A packet window is used for extracting spatial and temporal features. We select 20 important features by considering their information gain (IG), information gain ratio (IGR), χ^2, and ReliefF. In evaluation, the proposed DAE-GAN model is evaluated over NSL-KDD [4] and UNSW-NB15 [5]. Results demonstrate that DAE-GAN model significantly outperforms baseline methods. We also carry out experiments to detect abnormal traffic using four raw traffic datasets used in Kitsune [6] by combining the feature extraction framework and the DAE-GAN model. Experimental results show that our method achieves at least 99.5% precision in four datasets, while the highest recall is 97%. The performance of our method outperforms Kitsune in terms of precision and recall.

The contributions of this paper can be summarized as follows:

- An efficient spatial and temporal feature extraction framework is proposed for real-time traffic processing. Twenty relevant features are selected for training and classification, which lead to a low detection latency and high accuracy.
- A novel DAE-GAN model is proposed for efficient data augmentation in high-dimensional spaces and accurate semi-supervised anomaly detection. The PA generator can efficiently produce pseudo anomalies, adapted to high-dimensional spaces.
- A well-designed adversarial learning strategy is used in the DAE-GAN model. The detection sensitivity and precision of the model are further enhanced by training on pseudo anomalies.
- Experiments are conducted on a GPU-based platform and a mobile device. We compare the precision, recall, and f1-score of DAE-GAN over different attacks with several benchmark algorithms. We also verify the computational efficiency of the model implemented in mobile devices.

TABLE I: A table of acronyms.

Acronyms	Description
DAE	Denoise autoencoder
GAN	Generative adversarial network
IG	Information gain
IGR	Information gain ratio
kNN	k-Nearest-Neighbor
MLP	Multi layer perceptron
MSE	Mean square error
PA	Pseudo anomaly
SVM	Support vector machine

The rest of the paper is organized as follows. Section II reviews the related work of traffic data preprocessing and anomaly detection. Section III discusses the dataset used in our work and processing methods. Section IV presents our proposed abnormal traffic detection model DAE-GAN. Experimental results are presented in section V. Finally, we conclude the paper and provide future research directions in section VI.

II. Related Work

A. Data Preprocessing

As summarized in [7], the features of traffic datasets can be categorized into packet-based and flow-based features. A packet is a formatted data unit during communication carried by a packet-switched network. A flow is defined as a unidirectional sequence of packets sharing the same source/destination IP address, source/destination port, IP protocol, ingress interface, and IP type of service. Packet-based features, such as protocol and total length, contain **spatial** information about individual packets in the traffic stream. In contrast, the flow-based features in a data stream consisting of consecutive packets reflect **temporal** information of the data stream.

Packet-based features can be extracted from a single packet without waiting for the arrival of the last packet in the traffic flow. Over the past years, packet-based analysis like deep packet inspection [8] [9] [10] has been widely used for fast and memory-efficient abnormal traffic detection. However, detecting whether a traffic flow is anomalous only after it ends does not enable real-time defense against anomalous traffic and is useful only in the so-called "post-mortem" traffic analysis.

Flow-based features are features extracted from a sequence of packets. Temporal information within flow-based features can be used to describe the potential behavior hidden in monitored traffic. Previous work model time sequences by Markov Chain, which has proved to be susceptible to noise [11]. Using flow-based features for traffic analysis has been

TABLE II: Comparison of work on abnormal traffic detection.

Category	Mechanism
Reconstruction	Deep autoencoder [6] Principal component analysis [16] Generative adversarial networks [17]
Clustering	Deep autoencoder + gaussian mixture model [18] Disagreement-based semi-supervised learning [19]
One-class Classification	One-class SVM [20] Principal component analysis + SVM [21] One-class SVM [22] multi modal deep autoencoder [23]

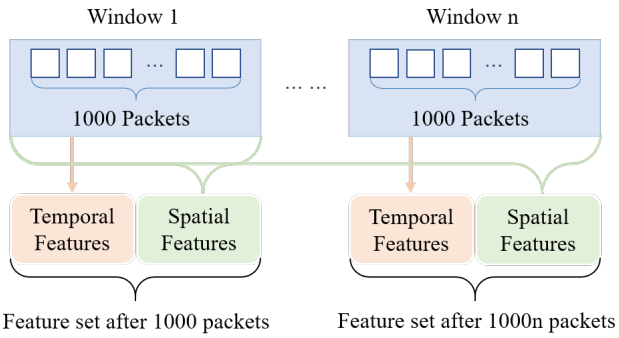

Fig. 1: The feature set will be extracted every 1000 packets, combining both temporal and spatial features. Temporal features record flow characteristics within every 1000-packet window, and spatial features store cumulative flow information across multiple windows.

a popular trend [12] [13] [14]. Furthermore, some studies encode both packet-based and flow-based features into image format for representation learning [15], thus improve the traffic identification performance.

Machine learning practitioners generally tend to use more features in a data sample to increase the information contained in each instance. However, abnormal traffic detection must consider real-time performance in a real-world application. An anomaly should be detected during a flow rather than after it is completed. Also, fitting a dataset with more features requires a larger model capacity, which has highly positive relevance to the computational complexity of the model, thus affect the application of an ML-based traffic identifier in mobile devices.

B. Abnormal Traffic Detection

Anomaly-based abnormal traffic detection is becoming popular due to the ability to detect unseen attacks. The core is modeling the pattern of normal traffic by training a neural network on a large amount of normal traffic data, where anomaly detection models are the cornerstones. Previous anomaly-based models can be categorized into three categories and are summarized in Table II:

- *Reconstruction-based:* In reconstruction-based methods, models learn the pattern of normal data by minimizing reconstruction loss during training. This method is based on the assumption that anomalies cannot be compressed to latent distribution. In this case, anomalies cannot be reconstructed with low error. Current approaches include deep autoencoder [6], principal component analysis [16], and generative adversarial networks [17]. While benefiting from the simple framework that only evaluates the reconstruction error, the performance of this kind of method is limited by the model complexity and the data structure, suffering from high false alarm and low accuracy.
- *Clustering Analysis:* Clustering analysis is a category of methods aiming at grouping data according to the similarity among samples. It is intractable to adopt distance-based methods to anomaly detection directly today because of the curse of dimensionality, which may damage the performance of clustering. Therefore, clustering is usually used after a dimensionality reduction process. To alleviate losing information for clustering during dimensionality reduction, researchers attempted to devise methods for joint learning of the parameter of both the dimensionality reduction module and the clustering analysis module. Zong et al. used the deep autoencoder to generate a low-dimensional representation and reconstruction error for each input data point, which was further input into a gaussian mixture model [18]. Li et al. proposed a semi-supervised method to classify raw data into different categories. Then, these classified data can be used to train a classifier for abnormal traffic detection [19].
- *One-class Classification*: One-class classification method follows the idea of density estimation. This approach requires a priori assumptions about the distribution of data used to train. A popular algorithm such as one-class support vector machine (SVM) [20] was used by Sotiris et al. in [21] and Eduardo in [22] to construct a one-class network intrusion detection system. This algorithm works well in low dimensions but suffers from performance degradation when running in high dimension spaces. The reason is that the distance-based evaluation function is weak at calculating robust similarities from noisy and sparse high dimension spaces. Bovenzi et al. [23] proposed a hierarchical deep autoencoder detection system that performs lightweight anomaly detection in the first stage and open-set attack classification in the second stage when a possible anomaly is detected.

As one kind of one-class classification algorithm, pseudo anomaly-based algorithm addresses the anomaly detection problem by sampling potential anomalies in the feature space spanned from normal data. In the absence of prior knowledge of unknown anomalies, pseudo anomalies are usually sampled in uniform distribution. However, sampling potential anomalies is not efficient in high-dimensional data spaces. Manevitz et al. pointed out that data points close to normal data can serve as potential anomalies during training, providing sufficient information about the boundary of normal data in an efficient

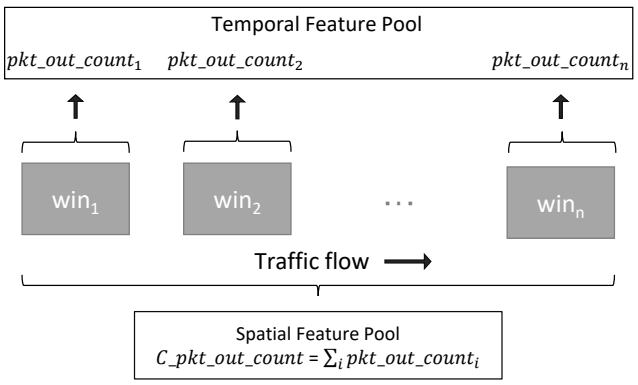

Fig. 2: The feature extraction process. Temporal features such as pkt_out_count are extracted in every packet window. Spatial features such as C_pkt_out_count are accumulated across multiple windows.

way [24].

Our approach uses multiple DAEs to produce pseudo anomalies efficiently. Only pseudo anomalies near the normal instances are effective, and training the discriminator on these data can promise a sensitive anomaly detector. As anomalies produced by DAE-GAN are close to the distribution of normal data, the number of outliers in generated anomalies is significantly decreased. To control convergence and performance of the trained model, hyper-parameters like noise factor and latent dimensionality of DAEs can be adjusted concerning the complexity of training data.

III. Feature Extraction

In this section, a new feature extraction method is proposed and evaluated under some comparison experiments. The following few subsections mainly discuss the reasons for choosing these features.

A. Description of Datasets

To show the stability and universality of the model, different types of dataset tests need to be performed on our proposed model. We used four datasets to represent four kinds of attacks, respectively.

Four flow-based datasets are extracted based on attack traffic data provided in Kitsune [6]. As we wanted to test the precision and recall of DAE-GAN under different classes of attack traffic, we chose four typical attack traffic datasets from Kitsune. These four attacks are Active Wiretap, ARP MitM (man-in-the-middle), Fuzzing, and Video Injection. Specifically, these four attack datasets also contain normal traffic. We present the number of normal and abnormal packets and flows in Table III.

B. Spatial and Temporal Features

As mentioned in some previous work [25] [26] [27], the first few packets exchanged in a flow, around 20 packets, are enough to distinguish whether this flow is benign or malicious. Li et al. also pointed out that the first few packets of a TCP connection are very considerable [28]. These approaches are proposed based on timeliness, meaning the malicious traffic needs to be detected as early as possible once it starts communicating with clients' devices.

As this paper aims at training models more precisely, new methods are proposed for extracting features. In the feature extraction process, spatial and temporal features are extracted because they are equally important. A time window is used to cut the spatial features.

Spatial features are considered as features containing no information related to time sequence. For example, the categorical feature flag_type is treated as a spatial feature. Some continuous features such as C_byte_sum are also spatial features.

Temporal features are incrementally calculated in sliding packet windows to achieve a high response rate. In most cases, temporal features cannot capture the characteristic of a whole flow because of limited window sizes. To capture intact flow features, spatial features will be stored in the memory and updated for every packet's arrival. For example, flow features like packet number, average packet size, and inter-arrival time will be calculated since the capture started. Unlike temporal features, these features will not be erased every 1000-packet window. Instead, they will provide cumulative information about every flow captured until the flow ends. The mechanism is shown in Fig. 1. However, it is infeasible to find the end of each flow based on its packet flags since the flow may not be finished normally. In this case, cumulative flow information will take up lots of memory space. To handle this issue, a much larger window size (20000 packets) is set for spatial features to clear the memory space after a certain number of packets being captured. The feature extraction process is depicted in Fig. 2.

As shown in Table IV, 20 features are extracted from captured traffic data. Among them, 13 are temporal features, and 7 are spatial ones. These features are derived from four public datasets and then selected based on the combination of different selection algorithms, including Information Gain (IG), Information Gain Ratio (IGR), χ^2, and ReliefF. IG, IGR, and χ^2 reflect correlations between discrete features. While ReliefF, as a discretization algorithm, is advantageous for finding the best split points for continuous features. These four algorithms are all applied to each dataset mentioned above. To ensure all useful features are chosen, the union of four sets of selected features is taken. Also, the final feature set is the union of feature sets derived from four different public traffic datasets so as to ensure the generalization performance of the feature sets. Eventually, 20 features are selected from 76 features in the original feature set.

C. Time Window chosen for Temporal Features

Many papers have proposed statistical ways to extract features such as duration, number of packets, and average packet size [29] [30] [31]. However, these approaches are carried out only when the flow is finished. Detecting whether

TABLE III: The overview of Kitsune dataset.

	Active Wiretap	ARP MitM	Fuzzing	Video Injection
Normal packets	1355473	1358995	1811356	2369902
Abnormal packets	923216	1145272	432783	102499
Normal flows	63	25	17	52
Abnormal flows	81	87	1793	7

TABLE IV: 20 selected features for abnormal traffic detection in DAE-GAN. Features started with "C_" and flag_type are spatial, the other features are temporal.

Flow Feature	Description
duration	Duration of the connection
pkt_out_count	Number of outgoing packets in a window
C_pkt_out_count	Number of outgoing packets across multiple windows
byte_out_count	Sum of outgoing packet byte in a window
byte_out_median	Median of outgoing packet byte in a window
byte_out_mean	Mean of outgoing packet byte in a window
C_byte_out_count	Sum of outgoing packet byte across multiple windows
C_byte_out_mean	Mean of outgoing packet byte across multiple windows
C_pkt_out/pkt_in	Outgoing packets divided by incoming packets
IAT_out_median	Median of the inter-arrival time of outgoing packets
byte_sum	Sum of packet byte in a window
byte_mean	Mean of packet byte in a window
byte_skew	Skewness of packet byte in a window
C_byte_sum	Sum of packet byte across multiple windows
C_byte_mean	Mean of packet byte across multiple windows
IAT_wave_count	Total number of peak and trough of inter-arrival time
C_min_IAT	Min value of peak and trough of inter-arrival time
magnitude	$\sqrt{\mu_{byte\_in}^2 + \mu_{byte\_out}^2}$
QAIT	Outgoing to incoming packet inter-arrival time
flag_type	Type of flag status

a traffic flow is anomalous only after it ends does not enable real-time defense against anomalous traffic and does not help the overall security of the system [32].

This paper leverages both long-sequence and short-sequence features. Long-sequence features mean the statistical features are calculated to start from the beginning to the end of this flow. To ensure the timeliness in short-sequence features, the time window is used. For example, if the time window's size is set to be 1000, that is to say, all the features will be calculated since the first 1000 packets arrived. When the following 1000 packets arrive, these 1000 packets will be taken into calculation, and a new instance will be created. Further, the size of time windows could be changed according to different attacks. The default time window size is set to be 1000 because 1000 is found to be the most balanced number as it both ensures enough number of instances and the training precision.

In this section, three parameters are used for selecting: precision, recall, and f1-score (harmonic mean of precision and recall). The size of 1000 is better than 200 for all these parameters.

IV. Abnormal Traffic Detection

A. Model Architecture

As depicted in Fig. 3, our proposed anomaly detection model, DAE-GAN, consists of a pseudo anomaly (PA) generator and a discriminator. The PA generator comprises a plurality of denoising autoencoders. The normal data is added with noise, fed into the DAEs, and the reconstructed data is taken as pseudo abnormal samples. We restrict the reconstruction ability by the latent size of each DAE and the noise factor. The discriminator is a binary classifier, as shown in the right part of Fig. 3. The ability to identify abnormal samples is learned through training on the samples generated by the pseudo-data generator and normal samples. The training is adversarial learning between the PA generator and the discriminator.

The structure of a DAE is shown in Fig. 3. Random noise is applied to the input sample, and by minimizing the reconstruction error, the model learns to reconstruct the original sample's most robust features. Generally speaking, a DAE learns the features of normal traffic from noisy normal traffic and adds a certain distribution of noise to the resulting normal data as a pseudo anomaly to train the discriminator. The noise applied to the original data is generated by multiplying a standard normal random noise $N(0,1)$ and a hyper-parameter called the noise factor f_{noise}. By setting the noise factor, we adjust the approximate degree of generated samples and original samples, thus changing the "abnormal" degree of generated samples. We take advantage of the fact that the reconstruction ability of DAEs can be modulated by adjusting f_{noise} and $latent\_dim$, thus avoid the "mode of collapse" problem and achieve higher sensitivity using multiple DAEs as PA generators, as shown in Fig. 4.

The discriminator used in this work is a multi-layer fully connected neural network. Its function is to learn the main characteristics of normal samples during training and to identify abnormal samples that deviate from the distribution of normal samples. A trained discriminator can distinguish abnormal samples from normal data.

B. Model Training

User actions and operating system behaviors lead to different traffic characteristics in real-world applications. Under the concern of privacy, labeled datasets containing all scenarios' features are unavailable. Even if it is possible to collect a network traffic dataset with enough features, training a model capable of classifying abnormal traffic would be tough. Classification training on large datasets typically requires a tremen-

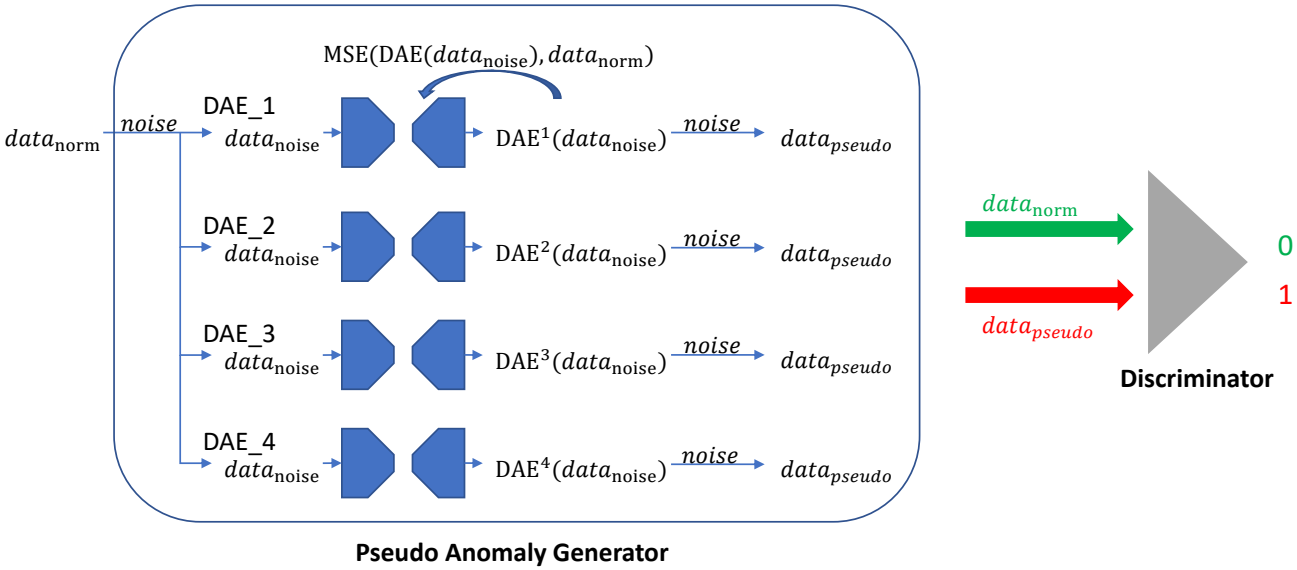

Fig. 3: The architecture of DAE-GAN. To train a DAE, original data is polluted by using noise or masks. The polluted data is reconstructed in the output, minimizing the loss function applied to the original data, and the reconstructed data can enforce the DAE to learn robust features.

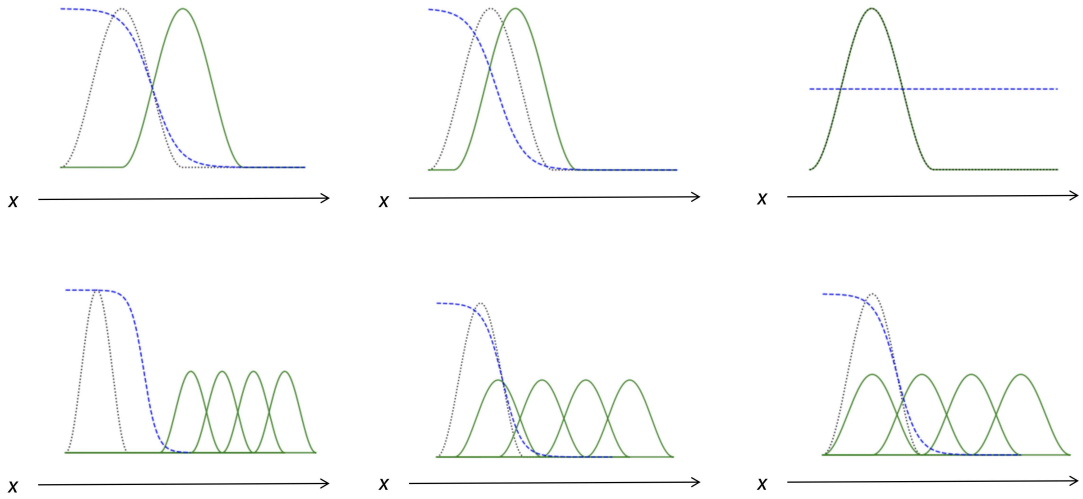

Fig. 4: Anomaly detector (blue dashed line) trained against pseudo data (black dotted line) generated by PA generator so that it learns to distinguish anomaly from normal data distribution (green solid line). It has been pointed out by Goodfellow et al. [33] that the adversarial training against one single generator may face the "mode of collapse" problem (i.e., top right figure). In DAE-GAN, using multiple generators with different reconstruction abilities can avoid the "collapse of mode" problem. In bottom figures, DAEs with weaker reconstruction ability still ensure that the classification boundary converges outside the normal data when DAE with the highest reconstruction ability overlaps the normal data distribution.

dously large network, which is computationally expensive and thus unsuitable for mobile devices. Our model adopts a semi-supervised method to train the model with only normal data. Training the network contains two phases: adversarial training and threshold selection. The detailed process is presented below:

- **Adversarial training using normal data**

The training process is presented in algorithm 1. Specifically, the abnormal traffic detection problem is a binary classification problem. We first train several DAEs by minimizing reconstruction loss, namely mean square error (MSE) between the normal data and DAE-generated data, thus push the reconstructed data close to the original data distribution.

Algorithm 1 Model Training

Input:
Real normal data: $data_{\text{normal}}$
DAE numbers: n
Training epochs: n_epoch
Noise factor: $f^i_{\text{noise}}, i \in \{1, 2, ..., n\}$
Output:
A trained discriminator Dis;

// **Training DAEs**
for $i \in [1, n]$ **do**
  $data^i_{\text{noise}} \leftarrow data_{\text{normal}} + N(0, 1) * f^i_{\text{noise}}$
  $DAE_{\text{loss}} = \text{MSE}(\text{forward}^i_{\text{DAE}}(data^i_{\text{noise}}), data_{\text{normal}})$
  $\text{Min}(DAE_{\text{loss}})$
end for
// **Training discriminator**
for $n \in [1, n\_epoch]$ **do**
  for $i \in [1, n]$ **do**
    $data^i_{\text{noise}} \leftarrow data_{\text{normal}} + N(0, 1) * f^i_{\text{noise}}$
    $data^i_{\text{DAE}} \leftarrow \text{forward}^i_{\text{DAE}}(data^i_{\text{noise}})$
    $pred^i_{\text{noise}} \leftarrow \text{Dis}(data^i_{\text{noise}})$
  end for
  $pred_{\text{real}} \leftarrow \text{Dis}(data_{\text{normal}})$
  $data_{\text{size}} \leftarrow \text{len}(data_{\text{normal}})$
  // target tensor
  $target_{\text{zeros}} \leftarrow \text{zeros}(data_{\text{size}})$
  $target_{\text{ones}} \leftarrow \text{ones}(data_{\text{size}})$
  // loss function
  $A = \text{BCE}(pred_{\text{real}}, target_{\text{zeros}})$
  $B = 1/n * \sum_{i=1}^{n} \text{BCE}(pred^i_{\text{noise}}, target_{\text{ones}})$
  $Dis_{\text{loss}} \leftarrow \text{A + B}$
  Backwards(Dis_{loss})
end for
return Dis

After that, an adversarial training process is conducted to train an anomaly detector (i.e., discriminator). In the training process, the discriminator is trained to predict the reconstructed noisy into 1 and the original data into 0. A cross-entropy function is used to measure the performance of the discriminator. By minimizing the cross-entropy loss, the discriminator learns to distinguish the noisy data from the normal data. The adversarial training process can be illustrated by the minimax objective:

$$\min_{P} \max_{D} V(D, P) \tag{1}$$

$$V(D, P) = \mathbb{E}_{x \sim pX}[\log D(x) + \log(1 - D(P(x)))] \tag{2}$$

In the adversarial training process, DAEs are trained to reconstruct the data to confuse the discriminator. The reconstructed data approaches the distribution of normal data, increasing the effectiveness and efficiency of the produced pseudo data. With a preset noise factor, the similarity between the restored and original data can be adjusted. We conduct an alternate training in the discriminator during the DAE training phase, which helps the discriminator learn the boundary of normal data.

Algorithm 2 Threshold Selection

Input:
Real normal data: $data_{\text{normal}}$
Target normal recall: $recall_{\text{target}}$
Output:
A threshold for distinguishing the anomaly: $Thres$

$thres_{\text{empirical}} \leftarrow 0$
$data_{\text{size}} \leftarrow \text{len}(data_{\text{normal}})$
$pred_{\text{real}} \leftarrow \text{Dis}(data_{\text{normal}})$
$target_{\text{zeros}} \leftarrow \text{zeros}(data_{\text{size}})$
$i \leftarrow 0$
while $i < 1$ **do**
  $recall_{\text{normal}} \leftarrow \textbf{Sum}(pred_{\text{real}} > i)/data_{\text{size}}$
  if $recall_{\text{normal}} \geq recall_{\text{target}}$ **then**
    $thres_{\text{empirical}} \leftarrow i$
    break
  end if
  $i \leftarrow i + 0.01$
end while
$thres_{\text{stat}} \leftarrow \textbf{mean}(preds) + 3 * \textbf{std}(preds)$;
if $thres_{\text{stat}} > 1$ **then**
  $Thres \leftarrow thres_{\text{empirical}}$
else
  $Thres \leftarrow \max(thres_{\text{empirical}}, thres_{\text{stat}})$
end if
return $Thres$

- **Threshold selection**
  After the training process, we have a discriminator for anomaly detection. The output of the discriminator for a given input is a continuous scalar ranging from 0 to 1. Selecting a threshold can significantly affect the model's generalization performance in test sets.
  Since we do not know the normal data distribution, we take a balanced value between empirical best-threshold and statistical best-threshold. As shown in algorithm 2, the empirical threshold $thres_{\text{empirical}}$ is obtained when the trained model classifies the normal data into 0 with the minimum threshold satisfying $recall > recall_{\text{target}}$. The statistical threshold $thres_{\text{stat}}$ is calculated by $thres_{\text{stat}} = \text{mean}(preds) + 3 * \text{std}(preds)$. Finally, the threshold can be obtained by taking the maximum of $thres_{\text{stat}}$ and $thres_{\text{empirical}}$.

C. Real-Time Detection

Once the model of DAE-GAN is trained, DAEs are discarded, and the discriminator is deployed on mobile devices for real-time anomaly traffic detection. As depicted in Fig. 5, traffic packets are preprocessed over a window to get a state vector in each flow monitoring point. The interval between two adjacent monitoring points is the window size. The preprocessed data is fed into the discriminator for identification. Each abnormal state in the same flow is accumulated as anomaly

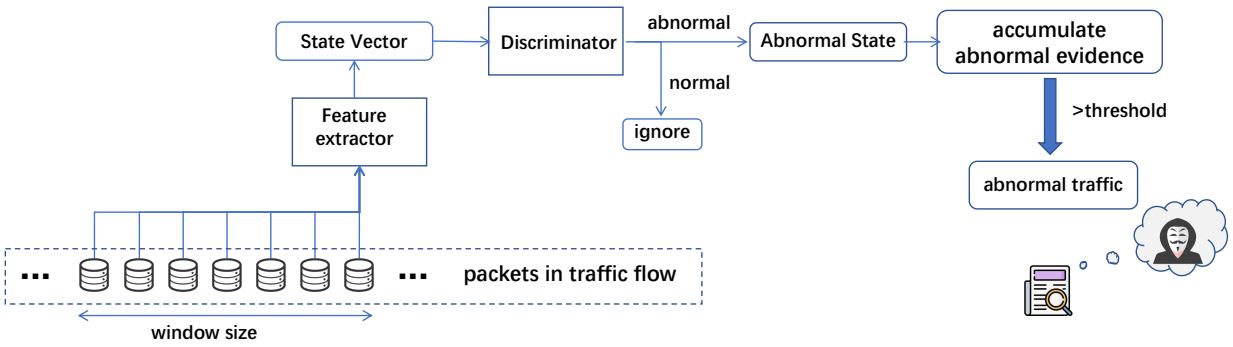

Fig. 5: The feature extractor extracts state vectors from collected packets and passes them to the discriminator. The abnormal states detected by the discriminator are accumulated to judge whether the flow is abnormal.

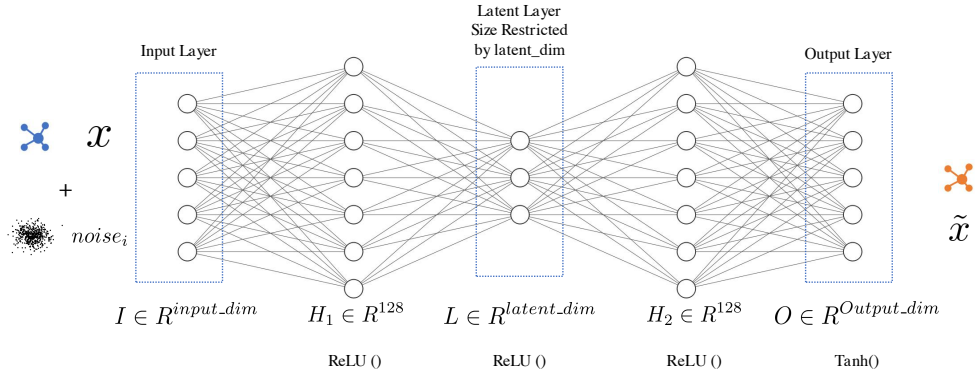

Fig. 6: The structure and parameters of a DAE. The input layer and output layer is denoted by capital letter I and O, respectively, the hidden layer is denoted by H_i, and the latent layer is denoted by L. The size of I and O is identical to the data size, and the size of hidden layers is shown in the graph. The latent size is set by a hyper-parameter $latent\_dim$ listed in the experiment configuration.

evidence. Once the accumulated abnormal evidence reaches a preset threshold, the traffic is identified as abnormal.

V. Experiments

With the extracted spatial and temporal features over a time window, detecting an abnormal traffic flow can be divided into two tasks: detecting momentary abnormal state (also referred to as evidence of anomaly) and detecting abnormal traffic by accumulated evidence. To evaluate the performance of DAE-GAN, we conduct four experiments: preliminary experiments, abnormal state detection, abnormal traffic detection, and packet parsing efficiency. Details of these four experiments are described below:

- *Preliminary Experiment*: This experiment is used to evaluate the anomaly detection performance of the DAE-GAN model on two popular datasets: NSL-KDD and UNSW-NB15. Both datasets are divided into normal and abnormal data, where abnormal data is used as the test set. The performance of DAE-GAN is compared with some baseline methods. For NSL-KDD, we include five state-of-the-arts that use identical dataset and one traditional classification method. For UNSW-NB15, we compare it with several traditional baseline methods. We also conduct an ablation study, which replaces the discriminator with a threshold test.
- *Abnormal State Detection*: This experiment is devised to detect a momentary anomaly in a monitoring point. As demonstrated in Fig. 5, the traffic flow is a sequence of packets cut into many segments using a window. We calculate statistic features and temporal features over a window to get a state vector, indicating the monitoring point's anomaly.
- *Abnormal Traffic Detection*: This experiment is designed to accumulate evidence of anomalies for detecting abnormal traffic flow. Evidence of abnormal states is accumulated on the same flow. The flow can be judged as abnormal once the accumulated evidence has exceeded a preset threshold. The detection sensitivity can be evaluated by the ratio of anomaly states detected in a flow. The real-time performance can be reflected by the processing time when the number of anomaly states has reached the threshold in an abnormal flow.
- *Packets Parsing Efficiency*: This experiment evaluates the efficiency of parsing packets in a flow. The performance

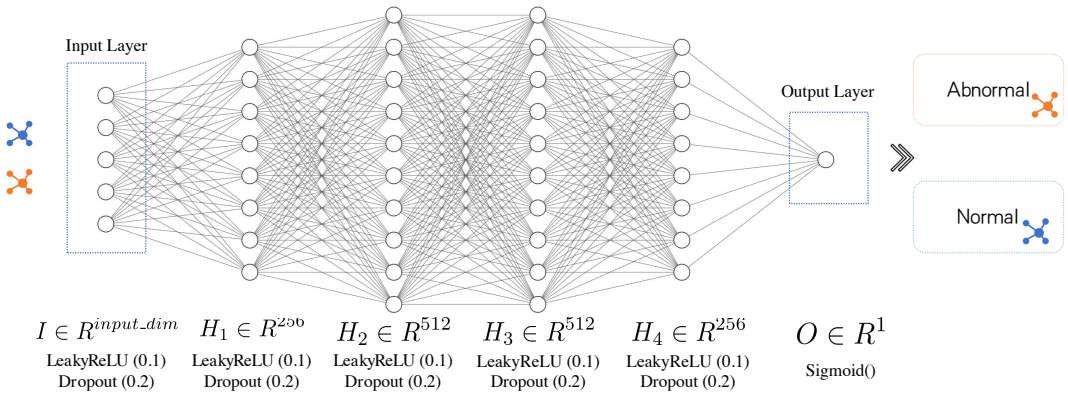

Fig. 7: The structure and parameters of the discriminator. The size of each layer is given in the figure. Capital I denotes input layer, H_i ($i \in \{1, 2, 3, 4\}$) denotes the i_{th} hidden layer, while capital O represent the output layer.

TABLE V: The environment used to perform the evaluations.

	GPU Sever **Ubuntu 18.04.5**	Mobile Device **Android 10**
CPU	Intel(R) Xeon(R) CPU 2.30GHz 1 Cores	Qualcomm Snapdragon 865 2.84GHz*1, 2.42GHz*3, 1.8GHz*4 8 Cores
RAM	13.3 GB	12 GB

is evaluated by calculating the number of packets parsed per second on the mobile device.

A. Experiment Configurations

The parameters of a DAE are shown in Fig. 6. All DAEs have the same architecture, composed of fully connected layers activated by ReLU functions. The dimension of each layer is listed in the figure. Each DAE is different from others with respect to the dimension of latent layer $latent\_dim$ and noise factor f_{noise}. Selecting a larger f_{noise} or a smaller $latent\_dim$ means constraining the reconstructing ability of the original data. In other words, the reconstructed data will deviate further from the original data distribution. In this experiment, four DAEs are used to play the role of a PA generator. Parameters of them are listed below:

- DAE_1: $latent\_dim = 20$, $f_{\text{noise}} = 0.4$
- DAE_2: $latent\_dim = 40$, $f_{\text{noise}} = 0.3$
- DAE_3: $latent\_dim = 80$, $f_{\text{noise}} = 0.02$
- DAE_4: $latent\_dim = 100$, $f_{\text{noise}} = 0.01$

Parameters of the discriminator are listed in Fig. 7. We adopt a fully connected network to construct a binary classifier. Each hidden layer is fully connected to the previous layer by a weight matrix and activated by a leaky ReLU function. Dropout is adopted to avoid overfitting and improve generalization performance. To achieve binary classification, a Sigmoid function is placed at the last layer of the discriminator.

The detection performance is evaluated on a GPU server, and the parsing efficiency is evaluated on a mobile device. Details of both platforms are shown in Table V.

B. Evaluation Criterion

In the abnormal state detection experiment, we use three assessment criteria to measure the model's performance: precision, recall, and f1-score. In the experiment on abnormal traffic detection, we use flow-wise criteria to evaluate the model's performance. We further evaluate the speed of parsing packets in the feature extractor. Let TP, FP, TN, and FN denote true-positive, false-positive, true-negative, and false-negative, respectively. Details of the used criteria are listed below:

- Precision f = TP/(TP+FP). Precision is used to measure the proportion of truly abnormal samples in tested positive samples.
- Recall r = TP/(TP+FN). Recall is the proportion of samples detected as positive among all abnormal samples. It reflects the sensitivity of the model to abnormal data.
- F1-score $2pr/(p+r)$. F1-score is the harmonic mean of precision and recall, a measure of the accuracy of tests in binary classification and statistic analysis.
- Flow-wise performance. We calculate the rate of states identified as abnormal in each flow to show the flow-wise sensitiveness and robustness.
  In normal traffic, it can be calculated by *FP states in a normal flow* divided by *overall states in a normal flow*.
  In abnormal traffic, it can be calculated by *TP states in a normal flow* divided by *overall states in an abnormal flow*.
- Parsed packets per second. This metric measures how fast a mobile device parses packets.

We hope to improve the detection sensitivity and accuracy as much as possible on the premise of ensuring a low FP rate to ensure that the precision and recall are as high as possible.

C. Preliminary Experiment

The preliminary experiment compares our proposed DAE-GAN model with some baseline methods over NSL-KDD [4] and UNSW-NB15 [5] datasets.

For comparison over NSL-KDD, baseline methods are described in Table VI. The NSL-KDD dataset contains 67,343

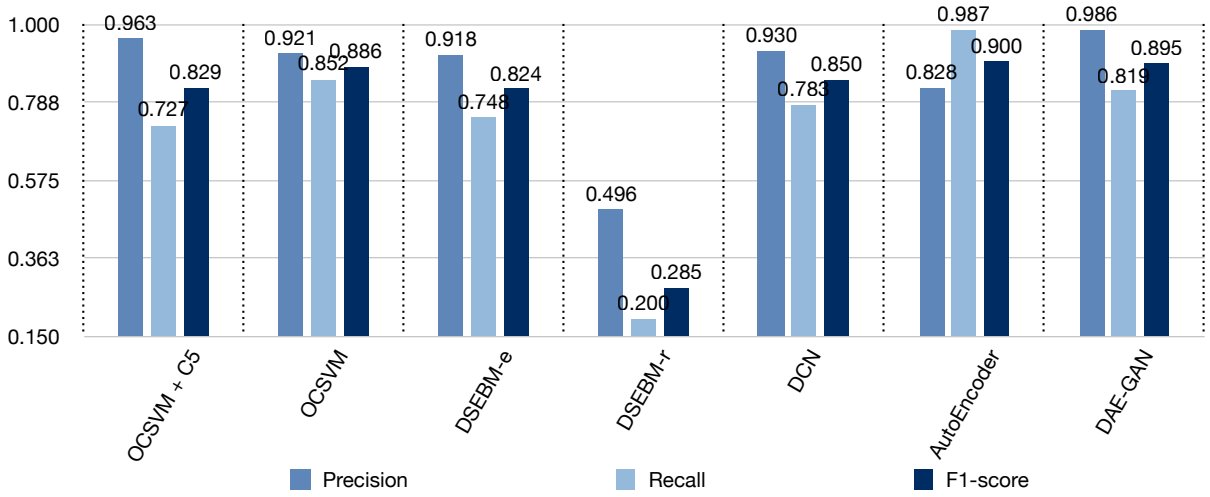

Fig. 8: Preliminary experiment on NSL-KDD. The results of DAE-GAN and baseline methods are demonstrated in histograms. Precision, recall, and f1-score are shown in different colors.

TABLE VI: Anomaly detection baseline methods.

Baseline Methods	Descriptions
DCN [34]	Deep clustering network (DCN) is a state of the art cluster method proposed by Bo Yang et al. The method jointly optimize dimensionality reduction component and k-means-based clustering component. Distance from sample to cluster center is used as the criterion in DCN-based anomaly detection.
DSEBM-e [35]	DSEBM-e is an unsupervised anomaly detection method. The method calculate the energy of the sample which can be then used to judge the anomaly of the sample.
DSEBM-r [35]	DSEBM is proposed by Zhai et al.,sharing the same architecture with DSEBM-e. Different from DSEBM-e, the DSEBM-r use reconstruction error as criterion for anomaly detection.
OCSVM [36]	One Class SVM (OCSVM) is a generalized version of support vector machine, popular in anomaly detection task for its ability to detect novelty.
OCSVM+C5 Decision Tree [37]	The method recently proposed by Khraisat detect anomaly by stacking C5 Decision Tree and OCSVM.
AutoEncoder	A simple autoencoder trained on normal data is used to detect anomaly, high reconstruction error is used as indication of anomalies.

normal samples and 58,630 abnormal samples. We randomly select 60,000 normal samples for training the model. In the test set, 7,000 normal samples are selected from the remaining normal data, and 7,000 abnormal samples are randomly selected from 58,630 abnormal samples. Experiment results are shown in Fig. 8. According to the results, our proposed method has the highest precision, and the f1-score is comparable to autoencoder-based methods. Autoencoder-based methods have a relatively high recall.

For comparison over UNSW-NB15, we choose some classic classification methods as baseline methods. They are k-Nearest-Neighbor (kNN) Classifier, multi-layer perceptron (MLP) Classifier, and decision tree classifier. The UNSW-NB15 dataset contains 19,488 normal samples and 61,686 abnormal samples. We randomly select 15,000 normal samples for training the model. In the test set, 4,000 normal samples are selected from the remaining normal data, and 4,000 abnormal samples are randomly selected from 61,686 abnormal samples. Experiment results show that DAE-GAN can achieve a precision of 98.54%, recall 69.38%, and f1-score 81.43% over the UNSW-NB15 dataset. In contrast, the precision of kNN is 98.31%, MLP is 98.37%, and the decision tree is 98.09%.

This experiment results demonstrate the high performance of the DAE-GAN model in anomaly detection. Training on only normal data, DAE-GAN has significantly high precision and recall in execution mode. In other words, the model can generalize well to unseen normal data and is sensitive to novel attacks.

We also replace the discriminator to conduct an ablation study. Four DAEs are used to generate pseudo anomalies. We evaluated the maximum, mean, and minimum value of the MSE between the original normal data and generated pseudo anomalies. Results are displayed in Table VIII. We can see that the generated anomalies show great differences from normal data. In addition, the difference becomes constant as the noise factor gets smaller.

D. Abnormal State Detection

This experiment evaluates the impact of feature extraction parameters on identifying a momentary state value in terms of accuracy and real-time performance. This experiment uses the dataset provided by Kitsune [6]. As shown in Table VII, we do this by trying different window sizes and different numbers of features. The win_200 and win_1000 mean the window sizes are set to 200 and 1000 packets, respectively. Meanwhile,

TABLE VII: Performance comparison with different window sizes and feature selection. Best results are shown in bold.

Attack	win_200 & $feat\_76$			win_1000 & $feat\_76$			win_1000 & $feat\_20$		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
ARP	0.975	0.039	0.075	0.989	**0.089**	**0.163**	**0.990**	0.083	0.153
Fuzzing	0.988	0.103	0.186	**0.996**	0.258	0.409	0.991	**0.385**	**0.555**
Video Injection	0.975	0.039	0.075	**1**	0.215	0.354	0.941	**0.606**	**0.737**
Active Wiretap	0.992	0.124	0.221	**1**	0.134	0.236	0.993	**0.232**	**0.376**

TABLE VIII: Ablation study of DAE-GAN without discriminator.

MSE	Maximum	Mean	Minimum
DAE1 ($f_{\text{noise}} = 0.04$)	9.8936	1.6341	0.0065
DAE2 ($f_{\text{noise}} = 0.03$)	10.0474	1.6322	0.0094
DAE3 ($f_{\text{noise}} = 0.002$)	10.0474	1.6322	0.0094
DAE4 ($f_{\text{noise}} = 0.001$)	10.0474	1.6322	0.0094

$feat\_20$ denotes that the features extracted are 20 top-ranked features.

Comparing black and red dotted lines, we see that our model performs better under the setting of win_1000, which is in line with our expectations. Because features calculated over a larger window size typically contain more temporal information. We take this as a satisfying result for its higher performance and lower computational cost (theoretically 5 times lower than settings of win_200). Though a larger window size requires more time to collect and process data, the fast data transmission speed can alleviate the extra time consumption. Despite the satisfying results under win_1000, developers can also tune the window size to balance the efficiency and real-time processing of the traffic identifier concerning different circumstances.

To reduce the data complexity, feature selection is conducted to find the most relevant features. Based on the idea of making the features more concise, the method of information gain rate is used here to rank all features, and the top 20 features are chosen. As a control for the previous setting (3rd column), the 4th column reveals that the concise features are computationally efficient and ensure better performances (in terms of recall and f1-score). Furthermore, by reducing the number of processed features, there is a reduction in the complexity of the algorithm.

This experiment is conducted 5 times on four different attack datasets. The results demonstrate that DAE-GAN can detect abnormal state values at high precision, thus provide great potential for detecting abnormal flows in the following experiment.

E. Abnormal Flow Detection

To evaluate the efficiency of abnormal flow detection, DAE-GAN is compared with Kitsune [6] using the same dataset. Flow-wise performance is evaluated in this stage. Histograms in Fig. 9 compare the distribution of misjudgment rate (*misjudge points in a flow* divided by *detection points in a flow*) of DAE-GAN and Kitsune on each normal flow on four attack datasets. Histograms in Fig. 10 compare the distribution of successful detection proportion (*successful points in a flow* divided by *detection points in a flow*) for each abnormal flow in four attack datasets.

As demonstrated by the results, our proposed DAE-GAN shows higher robustness over normal traffic in four attacks. By contrast, Kitsune suffers from potential high false alarms, especially in the video injection dataset. Kitsune and DAE-GAN show different performance on abnormal traffic data. DAE-GAN has a higher detection rate than Kitsune on three datasets (ARP MitM, Active Wiretap, and Video Injection). However, Kitsune achieves a higher detection rate on the traffic flow for the Fuzzing dataset. One reason for the higher false alarm in Kitsune might be the trade-off between normal flow error and abnormal flow detection. Kitsune tends to improve the sensitiveness in abnormal flow detection by observing a slight deviation from the normal distribution. Therefore, some normal traffic flow will be misclassified.

As for three attack types (ARP MitM, Active Wiretap, and Video Injection) demonstrated in the figures, the detection rate is low using both techniques. This result may attribute to the possible case that these attack datasets generated by the author contain many normal packets, thus indistinguishable from normal traffic.

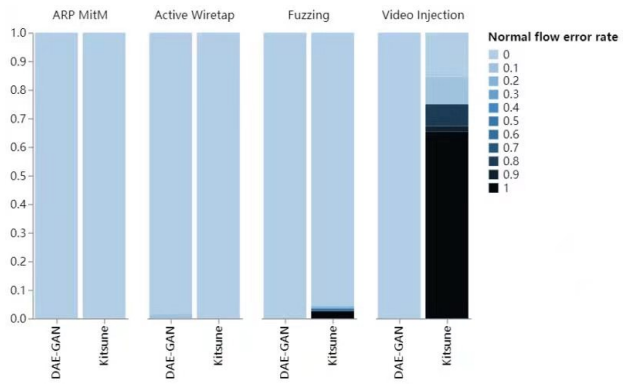

Fig. 9: Error rate of Kitsune and DAE-GAN on normal data (False positive states in a normal flow ÷ overall states in a normal flow). Darker color indicates that a normal traffic flow has higher possibility to be predicted as an anomaly.

Kitsune's datasets contain many flows with both 0 (normal) and 1 (abnormal) packets. Therefore, it is difficult to identify

TABLE IX: Performance comparison between Kitsune and DAE-GAN. Best results are shown in bold.

Attack	Kitsune			DAE-GAN		
	precision	recall	f1-score	precision	recall	f1-score
Active Wiretap	0.346	**0.342**	0.344	**0.993**	0.232	**0.376**
ARP MitM	0.677	**0.674**	**0.676**	**0.990**	0.083	0.153
Fuzzing	0.035	0.033	0.034	**0.991**	**0.385**	**0.555**
Video Injection	0.002	0.002	0.002	**0.941**	**0.606**	**0.737**

whether the flow is malicious. To solve this problem, each flow containing 0 and 1 labels is separated into different flows for convenience. The possible reason why 0 and 1 are mixed in the same flow is that the experimenter used the same channel to carry out both normal and abnormal activities.

By setting a detection-rate-based threshold, flow-wise precision, recall, and f1-score can be obtained. As shown in Table IX, the best result is shown in bold. The proposed DAE-GAN model achieves a high precision in all datasets. Also, the model shows a higher f1-score in three datasets.

Compared with the performance of Kitsune, our model is significantly generalizing well to new normal flow. In the DAE-GAN model, the mechanism of adversarial learning and the efficient pseudo anomaly generator help build high sensitiveness to abnormal data. On the other hand, the threshold selection based on empirical and statistical analysis ensures a low false alarm, thus achieve great generalization on normal traffic. It is worth mentioning that Kitsune has a high detection rate in the Fuzzing dataset. The reason for the huge difference is that Kitsune's performance is better at analyzing malicious flows with the small packet number. While for abnormal flows containing a large number of packets, Kitsune cannot detect them well.

F. Packets Parsing Efficiency

We further evaluate the packet parsing speed on the mobile device. The result is demonstrated in the Fig. 11. Most packets can be parsed at around 600 packets per second, while others can be parsed faster (around 700 packets per second). Compared with the packet arriving speed, the feature extractor can achieve real-time analysis of the arrived packets.

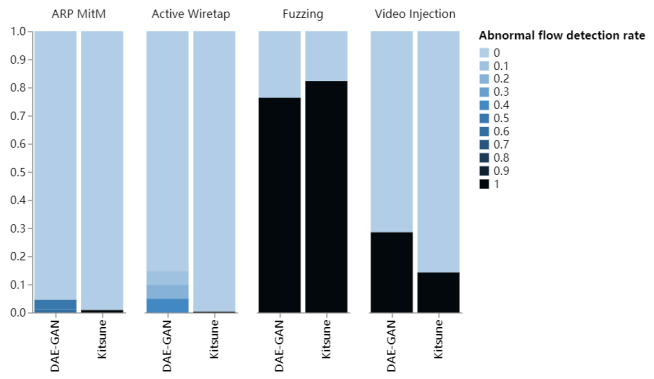

Fig. 10: Detection rate of Kitsune and DAE-GAN on abnormal data (True positive states in a normal flow ÷ overall states in an abnormal flow). Darker color indicates that an abnormal traffic flow has a higher possibility to be detected.

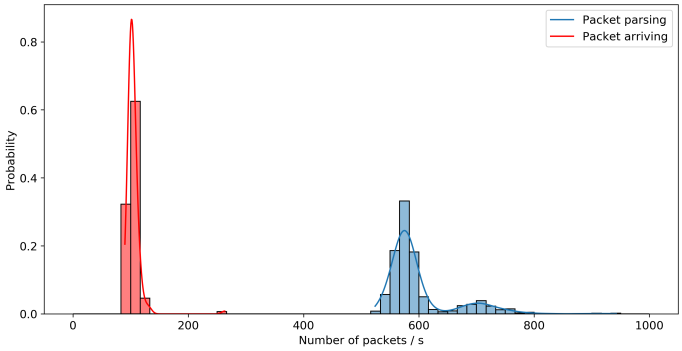

Fig. 11: Packets parsing efficiency. The packet parsing speed (number of packets per second) is shown in blue histogram. As a comparison, the packet arriving speed is shown in red histogram. We also draw the kernel density estimation curve of the respective histogram to visualize the density distribution.

VI. Conclusion

To solve the abnormal traffic detection problem, we propose an efficient feature extraction framework and an adaptive semi-supervised learning model named DAE-GAN, exploiting both normal data and pseudo anomalies. The feature extraction framework can quickly extract spatial and temporal features from raw traffic and select the 20 most important features. The DAE-GAN can train a discriminator to detect anomalies accurately in high dimensional spaces by using efficient data augmentation containing DAE-generated pseudo anomalies. To further improve the accuracy and precision of the discriminator, an adversarial learning process is conducted to adjust the classification boundary of the discriminator closer to the distribution of normal traffic. Substantial experiments on public benchmark datasets showed that the proposed model could effectively extract features and detect abnormal traffic with high accuracy, outperforming the state-of-the-art model.

In future work, we expect to add new modules to optimize the sensitivity of our model by utilizing labeled malicious traffic. In addition, considering the sparsity of the malicious traffic, a few-shot learning approach could be effective. A

lightweight and robust model trained on small samples can also be designed and deployed on mobile devices.

REFERENCES

[1] L. Alcantara, G. Padilha, R. Abreu, and M. d'Amorim, "Syrius: Synthesis of rules for intrusion detectors," *IEEE Transactions on Reliability*, 2021.

[2] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[3] X. Li, M. Zhu, L. T. Yang, M. Xu, Z. Ma, C. Zhong, H. Li, and Y. Xiang, "Sustainable ensemble learning driving intrusion detection model," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2021.

[4] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[5] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018, pp. 1–15.

[7] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[8] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," in *11th Symposium on High Performance Interconnects, 2003. Proceedings.* IEEE, 2003, pp. 44–51.

[9] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "Algorithms to accelerate multiple regular expressions matching for deep packet inspection," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 339–350, 2006.

[10] F. Yu, Z. Chen, Y. Diao, T. V. Lakshman, and R. H. Katz, "Fast and memory-efficient regular expression matching for deep packet inspection," in *Proceedings of the 2006 ACM/IEEE symposium on Architecture for networking and communications systems*, 2006, pp. 93–102.

[11] N. Ye, Y. Zhang, and C. M. Borror, "Robustness of the markov-chain model for cyber-attack detection," *IEEE transactions on reliability*, vol. 53, no. 1, pp. 116–123, 2004.

[12] R. Alshammari and A. N. Zincir-Heywood, "Machine learning based encrypted traffic classification: Identifying ssh and skype," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–8.

[13] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *2004 IEEE/IFIP network operations and management symposium (IEEE Cat. No. 04CH37507)*, vol. 1. IEEE, 2004, pp. 599–612.

[14] W. Wang, Y. Shang, Y. He, Y. Li, and J. Liu, "Botmark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors," *Information Sciences*, vol. 511, pp. 284–296, 2020.

[15] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *International conference on neural information processing*. Springer, 2017, pp. 858–866.

[16] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 1–37, 2011.

[17] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient gan-based anomaly detection," *arXiv preprint arXiv:1802.06222*, 2018.

[18] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[19] W. Li, W. Meng, and M. H. Au, "Enhancing collaborative intrusion detection via disagreement-based semi-supervised learning in iot environments," *Journal of Network and Computer Applications*, vol. 161, pp. 1–9, 2020.

[20] D. Tax, "One-class classification: Concept learning in the absence of counter-examples," Ph.D. dissertation, Delft University of Technology, 2002.

[21] V. A. Sotiris, W. T. Peter, and M. G. Pecht, "Anomaly detection through a bayesian support vector machine," *IEEE Transactions on Reliability*, vol. 59, no. 2, pp. 277–286, 2010.

[22] E. G. da Silva, A. S. da Silva, J. A. Wickboldt, P. Smith, L. Z. Granville, and A. Schaeffer-Filho, "A one-class nids for sdn-based scada systems," in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2016, pp. 303–312.

[23] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "A hierarchical hybrid intrusion detection approach in iot scenarios," in *GLOBECOM 2020-2020 IEEE Global Communications Conference*. IEEE, 2020, pp. 1–7.

[24] L. M. Manevitz and M. Yousef, "One-class svms for document classification," *Journal of machine Learning research*, vol. 2, no. Dec, pp. 139–154, 2001.

[25] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[26] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE communications magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[27] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, pp. 1–17, 2021.

[28] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia, "Byte segment neural network for network traffic classification," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–10.

[29] D. Zuev and A. W. Moore, "Traffic classification using a statistical approach," in *International workshop on passive and active network measurement*. Springer, 2005, pp. 321–324.

[30] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," in *International workshop on passive and active network measurement*. Springer, 2004, pp. 205–214.

[31] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "Blinc: multilevel traffic classification in the dark," in *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, 2005, pp. 229–240.

[32] L. Bernaille, R. Teixeira, and K. Salamatian, "Early application identification," in *Proceedings of the 2006 ACM CoNEXT conference*, 2006, pp. 1–12.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, 2014, pp. 2672–2680.

[34] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," in *international conference on machine learning*. PMLR, 2017, pp. 3861–3870.

[35] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep structured energy based models for anomaly detection," in *International conference on machine learning*. PMLR, 2016, pp. 1100–1109.

[36] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," *Advances in neural information processing systems*, vol. 12, pp. 582–588, 1999.

[37] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine," *Electronics*, vol. 9, no. 1, pp. 173: 1–18, 2020.

Zecheng Li is currently pursuing his Ph.D. degree in the Department of Computing, The Hong Kong Polytechnic University under the supervision of Dr. Bin Xiao. He received his B.Eng. degree from the School of Information Science and Technology, Southeast University, Nanjing, China, in 2017. His research interest lies in network security, blockchain security, and smart contract security.

Shengyuan Chen is currently pursuing a Ph.D. degree in the Department of Computing, The Hong Kong Polytechnic University. He received his B.Eng. degree from the School of Information Science and Technology, Fudan University. His research interest lies in abnormal traffic detection and identification.

Hongshu DAI is currently pursuing a B.Eng. degree in the Department of Computing, The Hong Kong Polytechnic University. His research interests include Big Data technology and feature engineering.

Dunyuan XU is currently pursuing a B.Eng. degree in the Department of Computing at The Hong Kong Polytechnic University. His research interests include AI, Big data analytics.

Cheng-Kang Chu Dr. Chu received his Ph.D. in Computer Science from National Chiao Tung University, Taiwan. He is a senior researcher of Huawei International, Singapore. Before joining Huawei, Dr. Chu was a research scientist in Cryptography and Security department at Institute for Infocomm Research (I2R) in Singapore. Dr. Chu has had a long-term interest in the development of new technologies in applied cryptography, cloud computing security and IoT security. His research now focuses on mobile security, IoT security, decentralized digital identity, etc. Dr. Chu has published many research papers in major conferences and journals like PKC, CT-RSA, AsiaCCS, IEEE TPDS, IEEE TIFS, etc. and received the best student paper award in ISC 2007. He also served as vice chair of IEEE CCNC 2012 and on the program committee of many international conferences.

Bin Xiao (S'01–M'04–SM'11) is an associate professor at Department of Computing, the Hong Kong Polytechnic University, Hong Kong. Dr. Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China, and Ph.D. degree in computer science from University of Texas at Dallas, USA. After his Ph.D. graduation, he joined the Department of Computing of the Hong Kong Polytechnic University as an Assistant Professor. His research interests include AI and network security, data privacy, and blockchain systems. He published more than 180 technical papers in international top journals and conferences. Currently, he is the associate editor of IEEE IoTJ, IEEE TCC, IEEE TNSE, and Elsevier JPDC. He is the vice chair of IEEE ComSoc CISTC committee. He has been the symposium co-chair of IEEE ICC 2020, ICC 2018 and Globecom 2017, and the general chair of IEEE SECON 2018. He is a senior member of IEEE, the member of ACM and CCF.