Handling Labeled Data Insufficiency: Semi-supervised Learning with Self-Training Mixup Decision Tree for Classification of Network Attacking Traffic

Yubo Hou, Sin G. Teo, Zhenghua Chen *Senior Member, IEEE*, Min Wu *Senior Member, IEEE*, Chee-Keong Kwoh, Tram Truong-Huu *Senior Member, IEEE*

Abstract—Motivated by the fast advancements in artificial intelligence (AI) technologies, recent research has moved towards using machine learning and deep learning to detect and classify security attacks in computer networks. However, most prior works adopt supervised learning methods, and the performance heavily depends on the amount of labeled data used to train the detection models. Network attack detection and classification is not an exception due to the lack of labeled data, especially the attacking traffic, which is much less than the regular (legitimate) traffic. Yet, labeling network traffic is also challenging and requires specific domain expertise. This paper proposes an efficient semi-supervised learning method for the classification of network attacking traffic, known as Self-Training Mixup Decision Tree (STM-DT). STM-DT first trains a decision tree on a small amount of labeled data and then uses the obtained model to predict labels of unlabeled samples. Some noisy labels will be removed by consistency. The predicted samples will then be mixed with labeled samples using `mixup` to train a new decision tree, which is the final desired classifier. We evaluate STM-DT using four network traffic datasets. Experimental results demonstrate that the proposed STM-DT method achieves higher macro F1 scores over different minority labeled data percentages.

Index Terms—Network Attacks, Semi-supervised Learning, Self-Training, Decision Tree.

1 Introduction

The Internet aids people in multiple business applications [1] such as education, medical treatment, and business. The proliferation of Internet interconnections has caused massive cyber-security attacks [2], [3], [4], leading to economic and reputation damages for organizations, governments, and many more. Many researchers have proposed various anomaly detection methods to address the critical issue as discussed, *e.g.*, Isolated Forest [5], GAN-based methods [6], [7], [8], and ensemble learning [9]. Many of these methods are applied in Intrusion Detection Systems (IDS) [10]. However, most anomaly detection methods utilize unsupervised learning to detect attacks due to insufficient labeled data, which cannot classify attacks effectively. Generally, classification based on supervised learning requires sufficient labeled data for training. However, collecting enough labeled data in cyber-security is time-consuming and labeling the data requires specific domain expertise. For instance, collecting attacking traffic in computer networks is challenging due to high traffic velocity. Therefore, the main challenge is how to use minimal labeled data to train an effective network attack classifier.

Many semi-supervised learning methods have been developed [11], [12], [13] to address the challenges mentioned above. Even though labeled data is limited, unlabeled data can easily be collected in real-world applications. Semi-supervised learning utilizes both labeled data and unlabeled data to improve the performance of models. Self-training [14] is one of the semi-supervised learning techniques. A classifier is trained with a small amount of labeled data. Then it is used to classify unlabeled data. The most confident unlabeled samples with their predicted labels are then merged with the original labeled training data set forming a new labeled training set to retrain the classifier. The whole process of training and prediction is repeated until convergence or achieving the desired performance.

With the fast advancement in deep learning algorithms, more and more deep learning-based semi-supervised methods have been proposed [15]. Most deep learning-based methods utilize data consistency to design loss functions. Data augmentation plays an important role in data consistency. However, most data augmentation methods for deep learning are designed for image data [16] that is unsuitable for network traffic data, which normally contains categorical features such as type of applications and network protocols. Due to the challenge of data augmentation, using deep

- *Y. Hou, S.G. Teo, Z. Chen, M. Wu are with Institute for Infocomm Research (I²R), Agency for Science, Technology and Research (A*STAR), Singapore.*
- *Y. Hou and C.-K. Kwoh are with School of Computer Science and Engineering, Nanyang Technological University, Singapore.*
- *T. Truong-Huu is with Singapore Institute of Technology (SIT), Singapore.*
  E-mail: hou_yubo@i2r.a-star.edu.sg, teosg@i2r.a-star.edu.sg, chen0832@e.ntu.edu.sg, wumin@i2r.a-star.edu.sg, asckkwoh@ntu.edu.sg, truonghuu.tram@singaporetech.edu.sg

learning in semi-supervised network attack classification becomes more difficult as deep learning is known as data-hungry methods [17]. Obviously, one can always train a deep learning model with a small amount of labeled data. However, the performance of the trained model may not be good enough to be used for the prediction of unlabeled samples.

In this work, we propose a Self-Training Mixup Decision Tree (STM-DT) for network attack classification. STM-DT uses a self-training mechanism for data annotation. Moreover, it utilizes the `mixup` method [18] to mix labeled data with unlabeled data, improving the model performance. As such, STM-DT can boost the performance of network traffic classification, and improve efficiency. Experimental results on several network traffic datasets also demonstrate that our proposed STM-DT approach is efficient and effective. The main contributions of this paper are summarized as follows.

- We build a semi-supervised learning method, known as Self-Training Mixup Decision Tree (STM-DT) for the network attack classification problem with less labeled data.
- We propose an approach to reduce the number of noisy labels of unlabeled data by `mixup` and the consistency between two decision trees.
- We adopt `mixup` method [18] to combine labeled data and unlabeled data for model training, which improves the performance of the semi-supervised learning method.
- We carry out extensive experiments to evaluate the performance of the proposed STM-DT and compare with benchmarking methods in various performance metrics.

The rest of the paper is organized as follows. Section 2 discusses the background of data augmentation and semi-supervised learning. We introduce the proposed Self-Training Mixup Decision Tree (STM-DT) for classification of network attacking traffic in Section 3. We present the experiments and analyze the experimental results of the proposed approach in Section 4. We discuss related work in Section 5. We conclude the paper in Section 6.

2 Background

In this section, we discuss the difference between data augmentation of homogeneous data and heterogeneous data, which can explain why deep learning-based semi-supervised learning methods cannot directly apply to network traffic data. Then we briefly introduce decision trees. Besides, we introduce several deep learning-based semi-supervised learning methods and some applications in intrusion detection. Finally, we elaborate on `mixup` and the state-of-art semi-supervised learning (SSL) method, Mixmatch, to show how SSL works in image data.

2.1 Data Augmentation of Homogeneous Data and Heterogeneous Data

Homogeneous data and heterogeneous data are two main data types in machine learning. Homogeneous data means all features of a data sample are similar and uniform. For instance, all elements in an image data are pixels. So the image is homogeneous data. Heterogeneous data means characters of some features are different. For example, there are several elements in network traffic data, including source IP, destination port, flow duration, and so on. So the network traffic data is heterogeneous.

Many deep learning models are developed for computer vision tasks. Due to the data-hungry nature of deep learning, data augmentation is an important technology in deep learning [19]. Data augmentation generates more synthetic data samples based on provided training samples to handle the problem of limited labeled data or imbalanced data among different classes. In semi-supervised learning, using unlabeled data to improve the performance of the model is a practical problem. Several semi-supervised learning models utilize the consistency of unlabeled data after data augmentation to train the model and achieve a good performance. Data augmentation includes random horizontal flip, random crop, image padding on sides, etc. Such data augmentation methods can be applied to an image because features are homogeneous, and the preliminary information is not changed. However, the network traffic dataset is heterogeneous, and these data augmentation methods are unsuitable for heterogeneous data.

2.2 Decision Trees

The decision tree is a widely used model in traditional machine learning. It can be used to solve classification problems [20], [21]. Fig. 1 shows an example of decision trees. A decision tree contains a root node, some internal nodes, and some leaf nodes. The root node represents the whole dataset. Each internal node represents a condition to divide a source into two subsets by a feature. Leaf nodes represent the classification results. The path from the root node to the leaf node is a sequence of conditions for classification.

There are several methods to find optimal conditions. ID3 [22] uses information gain as a criterion to build a decision tree. However, the ability of the generation of ID3 is dis-satisfactory. C4.5 [23] utilizes gain ratio instead of information gain to enhance generation. The CART decision tree [24] utilizes Gini Index to optimize feature selection and division. Both Information Gain and Gini Index are widely used as split criteria. In [25], the authors compared these two criteria and found that the two criteria disagree only in 2% of all cases in their experiments, which explains why most previously published empirical results concluded that it is not possible to decide which one of the two methods performs better.

2.3 Semi-supervised Learning

A mean teacher method was proposed in [26], which consists of student and teacher models. They share the same network architecture and are expected to output consistent predictions when the input is the same unlabeled sample. The weights of the teacher model are exponential moving average (EMA) weights of the student model over training steps. And the weights of the student model are updated according to the prediction error and the difference between the outputs of the student model and teacher model.

Π-model is another semi-supervised learning method proposed in [27]. It utilizes labeled samples to calculate

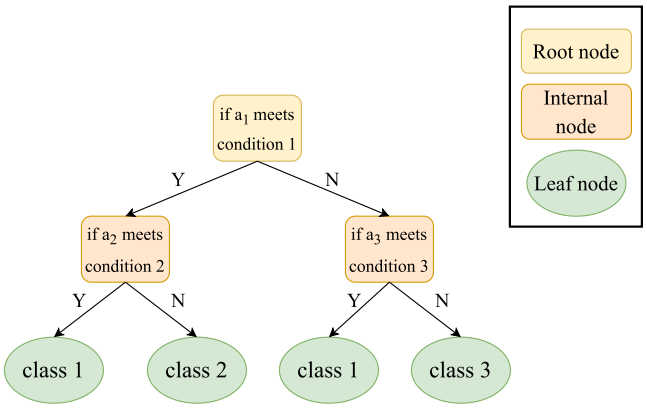


Fig. 1: Example of Decision Tree.

cross-entropy loss. When an unlabeled sample is fed to the model, it applies two different augmentations to the sample and calculates the mean squared error between two outputs. The network parameters are updated by these two losses.

MixMatch [28] is a recent state-of-the-art semi-supervised learning method. In MixMatch, labeled training data and unlabeled training data are augmented first. The model is trained on labeled data first. The prediction probability of unlabeled data is given by the trained model and is sharpened. Then labeled data with one-hot encoding and unlabeled data with predicted probability are mixed by Mixup. Parameters of the model are updated by cross-entropy loss based on mixed labeled data and L2 loss based on mixed unlabeled data. In [29], the authors presented a semi-supervised approach for intrusion detection and classification using Fuzzy-based method. It utilizes a single hidden layer neural network with random weights tested on the NSL-KDD dataset. In [30], a meta-learning-based semi-supervised learning framework is presented for attack detection in an unbalanced dataset. First, a visual representation method is introduced for describing packets using the RGB (red, green, blue) image format. Then, the attack type is classified using the obtained Delta score.

2.4 Mixup

`Mixup` [18] is a data augmentation method to improve the generalization of models and reduce the memorization of corrupted labels. It mixes pairs of samples and their labels. In essence, `mixup` generates new training examples by computing the convex combination of two existing ones. Algorithm 1 shows the detail for mixing two samples. α is a hyper-parameter to generate λ from the Beta distribution. Then λ is used to control the percentage of two samples in the mixed sample and take a value in the range between 0 and 1. Given two samples with their one-hot encoding or labels' probabilities (x_i, p_i), (x_j, p_j), a new sample $(\widetilde{x}, \widetilde{p})$ can be generated by `mixup` with a ratio of λ.

2.5 MixMatch

Similar to many other SSL methods, MixMatch utilizes a set of labeled samples X and a set of unlabeled samples U to train a model. Data augmentation is applied to both types of samples. For each labeled sample x, a transformed version

Algorithm 1 Mixup

Require: Sample x_i, label probability p_i, data sample x_j, label probability p_j, a positive number $\lambda \sim$ Beta(α, α)
1: $\widetilde{x}$ = λ * x_i + (1 - λ) * x_j
2: $\widetilde{p}$ = λ * p_i + (1 - λ) * p_j
3: **return** $\widetilde{x}, \widetilde{p}$

x' is generated by the data augmentation method. For each unlabeled sample u_j ($j = 1, \ldots, U$), K transformed versions $u'_{j,k}, k = 1, \ldots, K$ are generated by the data augmentation method. For each unlabeled sample u_j, MixMatch produces a pseudo label for the sample using the model trained above. This pseudo label is later used in the unsupervised loss term. To do so, it computes the average probability of the K augmented samples belonging to class y as follows:

$$\overline{q}_{j,y} = \frac{1}{K} \sum_{k=1}^{K} p_{\text{model}}(y|u'_{j,k}; \theta), \tag{1}$$

where $p_{\text{model}}(y|u'_{j,k}; \theta)$ is the prediction probability of sample $u'_{j,k}$ belonging to class y. Let L be the number of classes (i.e., $y = 1, \ldots, L$), we derive the model's predicted class distribution across K augmented samples as $\overline{q}_j = (\overline{q}_{j,y}|y = 1, \ldots, L)$. Then a sharpen function is used to reduce the entropy of the label distribution:

$$q_j = \text{Sharpen}(\overline{q}_j, T) = \frac{\overline{q}_j^{\frac{1}{T}}}{\sum_{y=1}^{L} \overline{q}_{j,y}^{\frac{1}{T}}}, \tag{2}$$

where T is a pre-defined hyper-parameter (set to 0.5 in our work).

The procedure of MixMatch is demonstrated in Algorithm 2. Labeled training data and unlabeled training data are augmented first. In [28], the data augmentation method includes random horizontal flip and random crop for image data. As this method is not suitable for network traffic feature records, in our work, data augmentation includes adding noise to some features randomly and removing some features (i.e., setting feature value to zero) randomly. The model is trained on labeled data. The trained model gives the probability of label of unlabeled data and is sharpened. Then a collection $\widehat{X}$ of labeled data x with one-hot encoding p and a collection $\widehat{U}$ of unlabeled data u with predicted probability q are concatenated and shuffled, whose result is represented by W. p is a one-hot encoding indicating the class of the respective sample (i.e., the 1's element) whereas q is a vector with the same length as p with each element representing the class probability. $\widehat{X}$ and $\widehat{U}$ are mixed with W separately by `mixup`. Parameters of the model are updated by the cross-entropy loss based on mixed labeled data X' and L2 loss based on mixed unlabeled data U'.

3 Methodology

In this section, we first present our data preprocessing method, known as frequency encoding, to encode categorical features. Then we elaborate on the structure of our proposed Self-Training Mixup Decision Tree (STM-DT).

Algorithm 2 MixMatch

Require: Labeled samples $X = \{(x_i, p_i); i = 1, ..., |X|\}$, unlabeled samples $U = \{u_j; j = 1, ..., |U|\}$

for i = 1 to $|X|$ **do**
  x'_i = DataAugment(x_i)
end for
$\widehat{X} = \{(x'_i, p_i); i = 1, ..., |X|\}$
for $j = 1$ to $|U|$ **do**
  for k = 1 to K **do**
    $u'_{j,k}$ = DataAugment(u_j)
  end for
  for $y = 1$ to L **do**
    $\overline{q}_{j,y}$ = $\frac{1}{K} \sum_{k=1}^{K} p_{\text{model}}(y|u'_{j,k}; \theta)$
  end for
  $\overline{q}_j = (\overline{q}_{j,y}|y = 1, \ldots, L)$
  $q_j = \text{Sharpen}(\overline{q}_j, T)$
end for
$\widehat{U} = \{(u'_{j,k}, q_j); j = 1, ..., |U|, k = 1, ..., K\}$
$W = \text{Shuffle}(\text{Concatenate}(\widehat{X}, \widehat{U}))$
$X' = \text{Mixup}((x'_i, p_i), (w_i, r_i)); (x'_i, p_i) \in \widehat{X}, (w_i, r_i) \in W, i = 1...|\widehat{X}|$
$U' = \text{Mixup}((u_i, q_i), (w_j, r_j)); (u_i, q_i) \in \widehat{U}, (w_j, r_j) \in W, i = 1...|\widehat{U}|, j = 1 + |\widehat{X}|...|W|$
return X', U'

3.1 Frequency Encoding

Some features in network traffic data are categorical features, such as port numbers and network protocols. They are essential for the model to make decisions but cannot be processed well by the model due to the following reasons. Firstly, even though a decision tree can handle categorical features, a model is easily over-fitted due to the vast amount of unique values in the categorical features. Secondly, different values in some categorical features of network traffic data cannot provide helpful information for classification. For instance, if attacks are only from one port in the training dataset, we cannot make a decision based on this port because the attacker can switch port numbers. So it is necessary to convert categorical features to meaningful numerical features. Commonly, such features can be converted to numerical features by one-hot encoding in a deep learning model. However, one-hot encoding is not applied in network traffic data due to the unique value of categorical features. For instance, there are 65536 port numbers. If we utilize one-hot encoding, we will need a vector with 65536 dimensions to represent it, which will cause a curse of dimensionality. Another problem is that data augmentation is hard to work with one-hot encoding.

To solve these problems, we propose a method called frequency encoding. It can convert a categorical feature to a numerical part according to the number of occurrences in the dataset. It counts the number of occurrences of each unique value in the training data and normalizes the result to get the frequency. Then it replaces the original feature value with its frequency. The frequency of each categorical feature value obtained from the training dataset will be used for the respective categorical feature value in the test set. This means that we perform frequency encoding on the training dataset and update the test set accordingly.

Destination Port
80
443
123

Destination Port
0.218
0.179
0.006

Fig. 2: Example of Frequency Encoding.

If a categorical value from the test dataset is not present in the training dataset, a zero value will be given as a default frequency. Equation (3) shows how the frequency is calculated.

$$Frequency(v) = \begin{cases} \frac{n_v}{N}; v \in V, \\ 0; v \notin V, \end{cases} \tag{3}$$

where V is a set of unique values from a categorical feature in the training data and v is one of the unique categorical values of V. n_v is the number of occurrences (counts) of value v in N samples in the training dataset. Fig. 2 shows an example of frequency encoding. A destination port is a categorical feature represented by a numerical value. Port 80 is commonly used for `HTTP` connection. If the count of port 80 is 218 in a total of 1000 training samples, the frequency of port 80 is 0.218 according to Equation (3). We replace port 80 with its frequency of 0.218 in both training and test datasets.

3.2 Self-Training Mixup Decision Tree

Network traffic samples are one-dimensional heterogeneous data. Thus, standard data augmentation methods such as random horizontal flip and crop used for images are not suitable for one-dimensional heterogeneous data. `Mixup` combines two samples linearly, increasing the diversity of training data and ensuring that the generated samples are in the data distribution. This is the rationale behind the use of `mixup` to combine two labeled samples and also mix labeled samples and unlabeled samples in different stages. Among machine learning and deep learning techniques, decision trees can handle one-dimensional data effectively and efficiently. It is effective in terms of the amount of labeled data needed for training a classification. It is also efficient as it does not need a long training time to make the model converge. We use decision trees in our work.

There are four phases in our proposed method, Self-Training Mixup Decision Tree (STM-DT): pre-training, labeling, training, and testing. Fig. 3 shows the structure of our proposed method. After data pre-processing, we utilize labeled sample set X to pre-train a decision tree model, f_1. And we mix X by `mixup` to pre-train another decision tree model f'_1. Then pre-trained models f_1 and f'_1 produce two pseudo labels for each sample in the unlabeled sample set, U. We keep the samples whose pseudo labels from f_1 and f'_1 are the same and mark these unlabeled samples with the pseudo labels as $\widehat{U''}$. This way, we can remove the samples with noisy labels. In the training phase, we mix labeled sample set X and unlabeled sample set $\widehat{U''}$ using `mixup` to train a new decision tree f_2 with the labeled sample set, X, and from `mixup`. In the end, we can test the performance

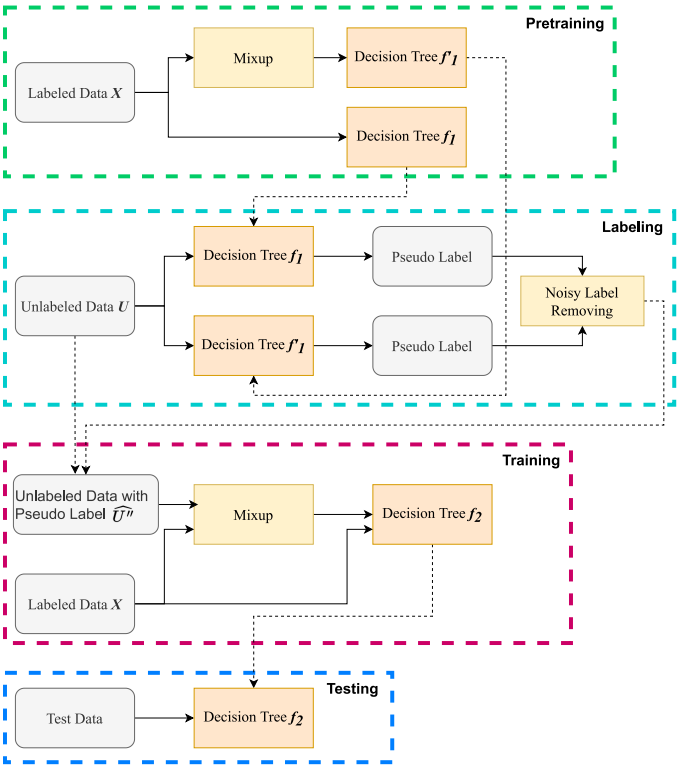


Fig. 3: Structure of Self-Training Mixup Decision Tree.

Algorithm 3 Pre-training

Require: Labeled dataset $X = \{(x_i, y_i); i = 1, \ldots, |X|\}$

$f_1 = \text{DTGenerator}(X)$
$p_i = f_1(x_i); i = 1, \ldots, |X|$
$\widehat{X} = \{(x_i, p_i); i = 1, \ldots, |X|\}$
$\widehat{X'} = \text{Mixup}((x_i, p_i), (x_j, p_j)); (x_i, p_i) \in \widehat{X}, i = 1, \ldots, |\widehat{X}|, j = 1, \ldots, |\widehat{X}|, i \neq j$
$y'_i = \text{Argmax}(p'_i); (x'_i, p'_i) \in \widehat{X'}, i = 1, \ldots, |\widehat{X'}|$
$X' = \{(x'_i, y'_i); i = 1, \ldots, |\widehat{X'}|\}$
$W = \text{Concatenate}(X, X')$
$f'_1 = \text{DTGenerator}(W)$
return Decision trees f_1 and f'_1

Algorithm 4 Labeling

Require: Unlabeled dataset $U = \{u_i; i = 1, \ldots, |U|\}$, decision tree f_1 and f'_1

$q_i = f_1(U_i); i = 1, \ldots, |U|$
$q'_i = f'_1(U_i); i = 1, \ldots, |U|$
$y_i = \text{Argmax}(q_i)$
$y'_i = \text{Argmax}(q'_i)$
$\widehat{U} = \{(u_i, y_i); i = 1, \ldots, |U|\}$
$\widehat{U'} = \{(u_i, y'_i); i = 1, \ldots, |U|\}$
$\widehat{U''} = \{(u_i, q_i); y_i = y'_i, y_i \in \widehat{U}, y'_i \in \widehat{U'}, i = 1, \ldots, |U|\}$
return Unlabeled dataset with probability vector of pseudo label $\widehat{U''}$

of the decision tree f_2 on test data. While Random Forest (RF) and XGBoost are more powerful tree-based ensemble models, there are two reasons why we do not use RF or XGBoost. First, our baseline methods are a single model and it is not fair to compare the performance of a single model with that of an ensemble model. Second, RF and XGBoost easily overfit noisy labels of pseudo-labeled data in the training set, which will reduce the performance of the model on the test set.

3.2.1 Pre-training Phase

We train two decision trees f_1 and f'_1 for labeling unlabeled sample set U. We first train decision tree f_1 on labeled sample set X. We refer DTGenerator to as the training process of a decision tree, given a training data set. The training process follows the training procedure of CART [24]. Before training decision tree f'_1, we get the probability p_i of each sample x_i in labeled sample set X. Labeled data with its probability vector is marked as $\widehat{X}$. Next, we mix $\widehat{X}$ with itself using `mixup` to get $\widehat{X'}$. Then, we train decision tree f'_1 on labeled sample set X and mixed labeled sample set X'. Algorithm 3 shows the procedure of the pre-training phase. The function `Argmax` converts probability to a concrete label for each sample.

3.2.2 Labeling Phase

After training two decision trees f_1 and f'_1 in the pre-training stage, we label unlabeled samples in U using these two decision trees. First, we label unlabeled samples in U using f_1 and get pseudo label y_i for each sample u_i. Since decision tree f_1 is trained with a small number of labeled samples, there must be some noisy labels in pseudo labels. We utilize the other decision tree f'_1 to reduce the number of noisy labels. The idea is that the pseudo label of the same sample from f_1 and f'_1 should be consistent. Specifically, we label the same unlabeled samples in U using f'_1 and get pseudo label y'_i for each sample u_i. Then we remove samples whose pseudo labels y_i and y'_i are different. The remaining unlabeled samples with consistent labels predicted by f_1 and f'_1 are saved to set $\widehat{U''}$ with their respective prediction probability. Algorithm 4 demonstrates the detail of the labeling phase.

3.2.3 Training Phase

After getting unlabeled data with pseudo label $\widehat{U''}$, we mix labeled sample set $\widehat{X}$ with $\widehat{U''}$ using `mixup` and get mixed sample sets $\widehat{X''}$ and $\widehat{U'''}$. Then we concatenate $\widehat{X}$, $\widehat{X''}$ and $\widehat{U'''}$ together and obtain $\widehat{W'}$. We convert probability r'_i of sample w'_i in $\widehat{W'}$ to label y'_i and get mixed data with label W'. Then we train a new decision tree f_2 with W'. Decision tree f_2 is the final classifier for testing. Algorithm 5 presents the detail of the training phase.

4 Experiments

In this section, we present the evaluation methodology and experimental results. We first present the detailed descriptions of datasets and the setting of model training. Then we discuss the evaluation results.

4.1 Experiment Setting

4.1.1 Datasets

In the evaluations, we use four datasets. Two of them are the latest and representative intrusion detection evaluation

Algorithm 5 Training

Require: Labeled dataset $\widehat{X} = \{(x_i, p_i); i = 1, \ldots, |X|\}$, unlabeled dataset $\widehat{U''} = \{(u_i, q_i); i = 1, \ldots, |\widehat{U''}|\}$,
1: $\widehat{W} = \text{Shuffle}(\text{Concatenate}(\widehat{X}, \widehat{U''}))$
2: $\widehat{X''} = \text{Mixup}((x_i, p_i), (w_i, r_i)); (x_i, p_i) \in \widehat{X}, (w_i, r_i) \in \widehat{W}, i = 1, \ldots, |\widehat{X}|$
3: $\widehat{U'''} = \text{Mixup}((u_i, q_i), (w_j, r_j)); (u_i, q_i) \in \widehat{U}, (w_j, r_j) \in \widehat{W}, i = 1, \ldots, |\widehat{U''}|, j = 1 + |\widehat{X}|...|\widehat{W}|$
4: $\widehat{W'} = \text{Concatenate}(\widehat{X}, \widehat{X''}, \widehat{U'''})$
5: $y'_i = \text{Argmax}(r'_i); (w'_i, r'_i) \in \widehat{W'}, i = 1, \ldots, |\widehat{W'}|$
6: $W' = \{(w'_i, y'_i); i = 1, \ldots, |\widehat{W'}|\}$
7: $f_2 = \text{DTGenerator}(W')$
8: **return** decision tree f_2

datasets, i.e., the CIC-IDS2017 [31] and Kitsune [32]. Another two datasets are generated in our lab, using raw traffic stored in the form of PCAP[1] files from the Stratosphere IPS Project [33]. We name these two generated datasets as Stratosphere IPS 5a dataset and Stratosphere IPS 5b dataset. We refer the reader to [34] for a detailed description on the feature extraction of these two datasets.

The CIC-IDS2017 dataset [31] is a network traffic dataset for intrusion detection collected by the Canadian Institute of Cybersecurity and the University of New Brunswick. The dataset has 14 kinds of classes, including one Benign class and 13 attack classes. Table 1 shows the number of samples for each class, and Fig. 4a shows the ratio of each class in the whole dataset. A number represents each class in Fig. 4a, and the number corresponds to the 'No.' column in Table 1. The Benign class accounts for around 80%, so it is a highly imbalanced dataset. There are 84 features for each sample. Several features used for sample identification such as Flow ID, Source IP, Source Port, Destination IP, and Timestamp are not used for model training. As shown in Table 2, several data types of features exist in the dataset, including numerical features (integer, float & timestamp) and category features (string). The model cannot directly process the categorical features, which are converted to numerical features using frequency encoding. Frequency is calculated based on the training dataset. Several samples include Infinity or NaN as a feature value of Flow Bytes/s and Flow Packets/s. These samples will be removed from the dataset. After data pre-processing, 79 features remain for model training.

The Kitsune Network Attack Dataset is established by Yisroel Mirsky *et al.* [32] in a real IP camera video surveillance network. It consists of nine network attack datasets. Each dataset is a binary classification dataset, containing Benign samples and some cyber-attack samples. We concatenate these nine datasets together for our multi-classification evaluation. The number of samples for each class is shown in Table 3 and the class distribution is shown in Fig. 4b. It is also a highly imbalanced dataset, as the Benign ratio is around 75%. There are 115 features pre-processed for machine learning by Yisroel Mirsky.

1. PCAP is also known as the libpcap/packet capture. It is an application programming interface that allows capturing of live network packet data.

TABLE 1: Number of samples in CIC-IDS2017 dataset

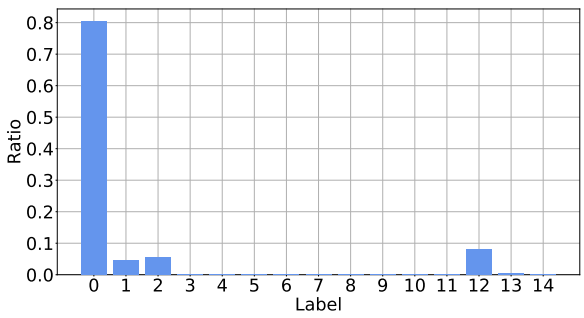
No.	Class	Number of Samples
0	Benign	2271320
1	DDoS	128025
2	PortScan	158804
3	Bot	1956
4	Infiltration	36
5	Web Attack – Brute Force	1507
6	Web Attack – XSS	652
7	Web Attack – Sql Injection	21
8	FTP-Patator	7935
9	SSH-Patator	5897
10	DoS slowloris	5796
11	DoS Slowhttptest	5499
12	DoS Hulk	230124
13	DoS GoldenEye	10293
14	Heartbleed	11

TABLE 2: Partial Features of CIC-IDS2017 dataset

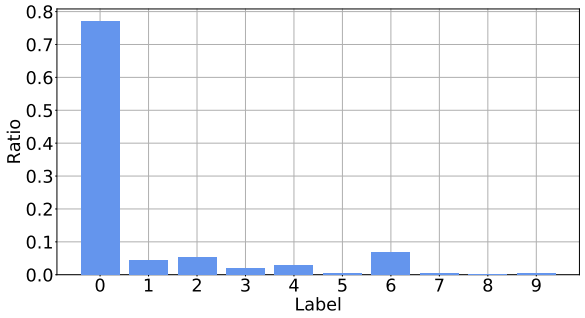
Feature	Data Type
Flow ID	string
Source IP	string
Source Port	string
Destination IP	string
Destination Port	string
Protocol	string
Timestamp	timestamp
Flow Duration	float
Total Fwd Packets	float
Fwd IAT Std	float
Fwd IAT Max	int
Fwd IAT Min	int
Bwd IAT Total	int
...	...

The Stratosphere IPS Project provides malware and normal data stored in the PCAP files. We extract 38 features using a traffic aggregation tool developed in [8]. After feature extraction with different aggregation criteria, we have two tabular datasets, Stratosphere IPS 5a, and 5b datasets. Then we remove some noise samples with incorrect labels. As shown in Table 4 and Table 5, the two datasets include one normal class and 11 botnet attacks.

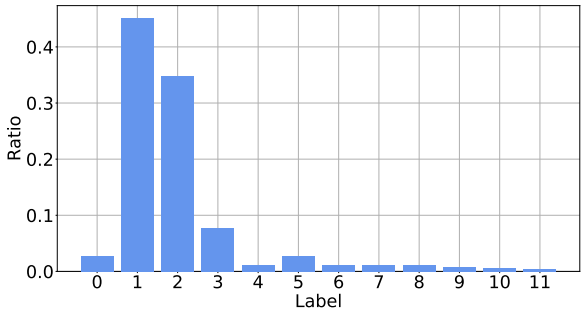
In our experiments, we divide each dataset into the training set and test set with a pre-defined ratio: 80:20 for the CIC-IDS2017 dataset, Stratosphere IPS 5a, and Stratosphere IPS 5b; 20:80 for the Kitsune Network Attack Dataset as there is sufficient data for training. The training set is further divided into a labeled training set and an unlabeled training set. The labeled training set accounts for a low ratio in the whole training data. It could happen that the number of labeled samples of several classes is fewer than 1 with such a low ratio (e.g., 2%). So we set a minimal number of samples for each class in the labeled training set to 30 to make it more realistic. If the total number of samples of a class in the training set is fewer than 30, we just use all the samples as the labeled training set. The ratio of the labeled training



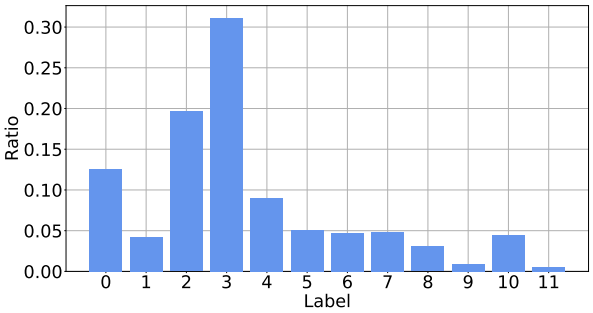
(a) CIC-IDS2017 dataset



(b) Kitsune dataset



(c) IPS Stratosphere 5a dataset



(d) IPS Stratosphere 5b dataset

Fig. 4: Distribution of classes on different datasets.

TABLE 3: Number of samples in Kitsune dataset

No.	Class	Number of Samples
0	Benign	16166316
1	Active Wiretap	923216
2	ARP MitM	1145272
3	Fuzzing	432783
4	Mirai Botnet	642517
5	OS Scan	65700
6	SSDP Flood	1439604
7	SSL Renegotiation	92652
8	SYN DoS	7038
9	Video Injection	102499

TABLE 4: Number of samples in Stratosphere IPS 5a dataset

No.	Class	Number of Samples
0	Normal	6648
1	Zeus	111221
2	Htbot	85793
3	Emotet	19252
4	Miuref	6943
5	Vawtrak	3083
6	Andro	3083
7	Sality	3061
8	Geodo	2813
9	Barys	2096
10	Yakes	1567
11	Necurse	1243

set is 0.2% for the CIC-IDS2017 and 0.05% for the Kitsune. For the Stratosphere IPS 5a and 5b datasets, this ratio is 2%.

4.1.2 Evaluation Metrics

We use precision, recall, and F1 score as the evaluation metrics. The precision and recall can be illustrated by four metrics, True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). The definitions of precision and recall are shown in Equation (4) and Equation (5).

$$Precision = \frac{TP}{TP + FP}. \tag{4}$$

$$Recall = \frac{TP}{TP + FN}. \tag{5}$$

Precision and Recall are usually a pair of contradictory performance metrics, and the higher the Precision, the lower the Recall. Thus, to characterize the model's all-around performance, we also use the F1 score, which can be interpreted as a harmonic mean of Precision and Recall. The F1 score reaches its best value at one and worst score at 0. The relative contribution of Precision and Recall to the F1 score is equal. The formula for the F1 score is defined as follows:

$$F1 = \frac{2 \times (Precision * Recall)}{(Precision + Recall)}. \tag{6}$$

We do not measure the performance by accuracy since the four datasets are highly imbalanced. Accuracy will be over 90% if the model predicts several majority classes well while it fails to classify all the minority classes. To eliminate the effect of the imbalanced problem, we take the macro av-

TABLE 5: Number of samples in Stratosphere IPS 5b dataset

No.	Class	Number of Samples
0	Normal	43683
1	Zeus	14570
2	Htbot	68191
3	Emotet	108003
4	Miuref	17551
5	Vawtrak	31403
6	Andro	16328
7	Sality	16882
8	Geodo	10612
9	Barys	3033
10	Yakes	15397
11	Necurse	1868

erage on the metrics for multi-classification evaluation. We obtain metrics for each class and calculate their unweighted mean as the final performance. Equation (7), Equation (8) and Equation (9) show definitions.

$$\text{Precision-macro} = \frac{1}{N}\sum_{i=1}^{N} Precision_i, \tag{7}$$

$$\text{Recall-macro} = \frac{1}{N}\sum_{i=1}^{N} Recall_i, \tag{8}$$

$$\text{F1-macro} = \frac{1}{N}\sum_{i=1}^{N} F1_i, \tag{9}$$

where N is the number of classes and i indicates the metric of which class.

4.1.3 Baseline Method

We compare the performance of the following deep learning-based semi-supervised learning models with our proposed STM-DT: Mean Teacher, MixMatch, and Π-model. The neural network architecture of all these baseline methods is a 2-layer 1D-CNN followed by 3 fully-connected layers. Table 6 shows the detailed parameters of 1D-CNN. Different datasets have different $latent\_size$ and num_class due to the difference in length of samples' features. For CIC-IDS2017, $latent\_size$ is 1136 and num_class is 15. For Kitsune, $latent\_size$ is 1712 and num_class is 10. For Stratosphere IPS 5a and 5b, $latent\_size$ is 560 and num_class is 12. The optimizer is SGD and the learning rate is 0.1 for large batch size. The batch size is 1000 samples for labeled data and 10000 samples for unlabeled data. The number of training epochs is 80. We use the model obtained after the last training epoch to predict the label for the samples in the test dataset and compute the performance.

For STM-DT, we use a decision tree with Gini as a criterion, and the minimum number of samples required to be at a leaf node is 5 to address the over-fitting problem and improve robustness under label noise [35].

4.1.4 Hardware Specifications

The experiments were conducted on a deep learning server with Intel@Core i9-9900K CPU@3.60GHz × 16, 64 GB of RAM, and one NVIDIA GeForce RTX 2080Ti GPU with 11 GB of memory.

TABLE 6: The structure of 1D-CNN

Layer No.	Layers
1	Conv1d(1, 6, kerneal size=[5,5,5])
2	BatchNorm1d + ReLu
3	Conv1d(6, 16, kerneal size=[5,5,5])
4	BatchNorm1d + ReLu
5	Linear($latent\_size$, 120) + ReLu
6	Linear(120, 84) + ReLu
7	Linear(84, num_class) + ReLu

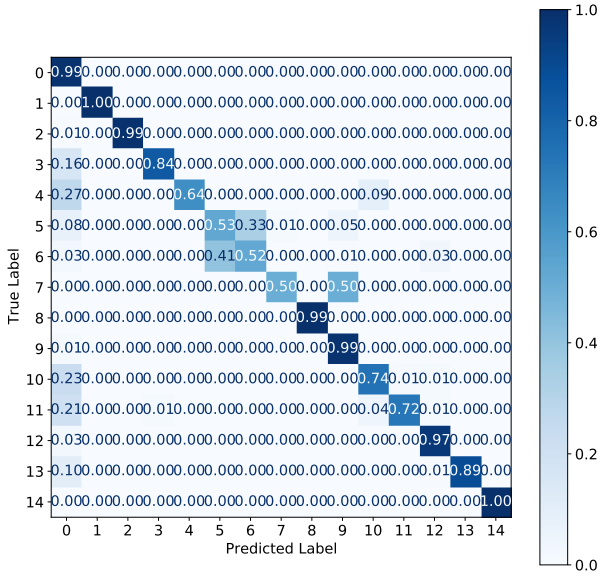


Fig. 5: Confusion Matrix of STM-DT on CIC-IDS2017.

4.2 Performance Analysis

To demonstrate why the decision tree is selected in our proposed method, we compare the performance of three classifiers, including SVM, decision tree, and 1D-CNN, trained on a small set of labeled training data. The ratio of labeled data of each dataset is described in Section 4.1.1. As shown in Table 7, the decision tree outperforms the other two classifiers on all four datasets. Compared to the worst performance on each dataset, DT improves the F1-macro by up to 115%.

In Table 8, we present the performance of all semi-supervised learning methods on all four network traffic datasets. The results are averaged over 5 runs. We observe that the proposed STM-DT outperforms all other semi-supervised learning methods on all datasets. Specifically, on the CIC-IDS2017 and Stratosphere IPS 5a dataset, STM-DT performs better than baseline methods on macro F1 score and Recall, while Mixmatch achieves a better Precision. On the Stratosphere IPS 5b dataset, the performance of STM-DT is better than other baseline methods on F1 score and Recall, while Precision of Mean Teacher is better than STM-DT. On the Kitsune dataset, STM-DT significantly achieves a higher F1 score by around 8.8% compared with the second-best method, and STM-DT outperforms other comparison

TABLE 7: Performance of supervised learning methods on various network traffic datasets with insufficient labeled data

Dataset	Model	F1-macro	Precision-macro	Recall-macro
CIC-IDS2017	SVM	0.464 ± 0.027	0.521 ± 0.038	0.490 ± 0.026
	1D-CNN	0.584 ± 0.016	$\mathbf{0.702 \pm 0.038}$	0.674 ± 0.062
	DT	$\mathbf{0.644 \pm 0.021}$	0.601 ± 0.028	$\mathbf{0.842 \pm 0.028}$
Kitsune	SVM	0.384 ± 0.009	0.503 ± 0.015	0.568 ± 0.008
	1D-CNN	0.709 ± 0.002	0.755 ± 0.004	0.729 ± 0.006
	DT	$\mathbf{0.827 \pm 0.009}$	$\mathbf{0.812 \pm 0.015}$	$\mathbf{0.911 \pm 0.013}$
Stratosphere IPS 5a	SVM	0.562 ± 0.018	0.725 ± 0.013	0.525 ± 0.011
	1D-CNN	0.675 ± 0.002	$\mathbf{0.764 \pm 0.009}$	0.651 ± 0.018
	DT	$\mathbf{0.681 \pm 0.012}$	0.735 ± 0.011	$\mathbf{0.661 \pm 0.016}$
Stratosphere IPS 5b	SVM	0.523 ± 0.002	0.701 ± 0.015	0.483 ± 0.002
	1D-CNN	0.683 ± 0.011	$\mathbf{0.791 \pm 0.001}$	0.639 ± 0.013
	DT	$\mathbf{0.726 \pm 0.011}$	0.756 ± 0.014	$\mathbf{0.705 \pm 0.011}$

TABLE 8: Performance of semi-supervised learning methods on various network traffic datasets

Dataset	Model	F1-macro	Precision-macro	Recall-macro
CIC-IDS2017	Mean Teacher	0.668 ± 0.009	0.675 ± 0.019	0.810 ± 0.027
	Mixmatch	0.683 ± 0.021	$\mathbf{0.706 \pm 0.011}$	0.787 ± 0.012
	Π-model	0.652 ± 0.003	0.644 ± 0.010	0.810 ± 0.030
	STM-DT	$\mathbf{0.685 \pm 0.019}$	0.650 ± 0.021	$\mathbf{0.844 \pm 0.015}$
Kitsune	Mean Teacher	0.771 ± 0.005	0.818 ± 0.017	0.758 ± 0.004
	Mixmatch	0.751 ± 0.005	0.830 ± 0.007	0.741 ± 0.015
	Π-model	0.712 ± 0.018	0.821 ± 0.016	0.648 ± 0.011
	STM-DT	$\mathbf{0.859 \pm 0.022}$	$\mathbf{0.860 \pm 0.031}$	$\mathbf{0.883 \pm 0.021}$
Stratosphere IPS 5a	Mean Teacher	0.734 ± 0.003	0.781 ± 0.039	0.677 ± 0.063
	Mixmatch	0.715 ± 0.017	$\mathbf{0.811 \pm 0.004}$	0.665 ± 0.017
	Π-model	0.730 ± 0.003	0.783 ± 0.009	0.703 ± 0.002
	STM-DT	$\mathbf{0.742 \pm 0.012}$	0.791 ± 0.015	$\mathbf{0.715 \pm 0.016}$
Stratosphere IPS 5b	Mean Teacher	0.734 ± 0.011	$\mathbf{0.807 \pm 0.007}$	0.695 ± 0.011
	Mixmatch	0.683 ± 0.011	0.785 ± 0.042	0.639 ± 0.014
	Π-model	0.730 ± 0.018	0.785 ± 0.024	0.673 ± 0.038
	STM-DT	$\mathbf{0.749 \pm 0.012}$	0.802 ± 0.018	$\mathbf{0.720 \pm 0.007}$

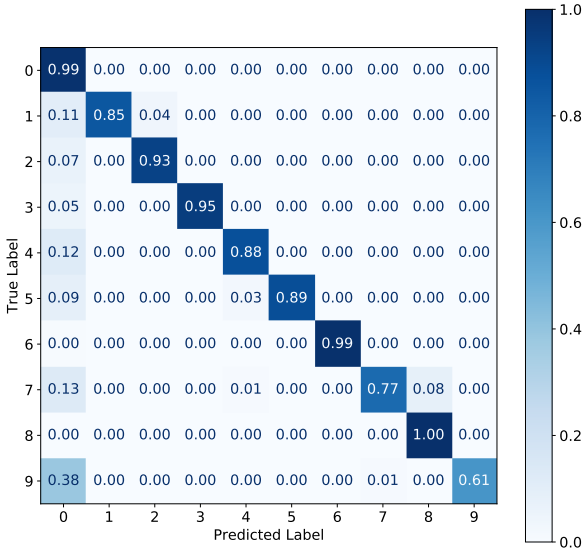


Fig. 6: Confusion Matrix of STM-DT on Kitsune Dataset.

methods in all metrics. According to Table 7, the performance of the Decision Tree on the Kitsune dataset is better than 1D-CNN. Thus, we can infer that the proposed STM-DT is also better than other semi-supervised methods based on 1D-CNN due to the high performance of DT. Since the proposed method is for a multi-classes classification task, we present the confusion matrix of STM-DT in Fig. 5–Fig. 8 to further illustrate the experimental results.

Due to the requirement of intrusion detection in efficiency, we also evaluate the training speed of the proposed method. Table 9 presents the training time of the proposed STM-DT and the baseline methods on various datasets. The training time of the deep learning-based method is measured over 80 training epochs on the entire training data. We observe that STM-DT can converge quickly. The speed is from 100 to 700 times faster than the baseline methods on various datasets because the proposed STM-DT only requires two training epochs, and in contrast, the baseline methods need dozens of training epochs.

4.3 Ablation Studies

Since our proposed method combines two semi-supervised learning mechanisms, we study the effect of each component to provide more insight. Specifically, we measure the effect of self-training (ST), noisy label removing (NLR) and `mixup` (M) on various datasets. Self-training means we use

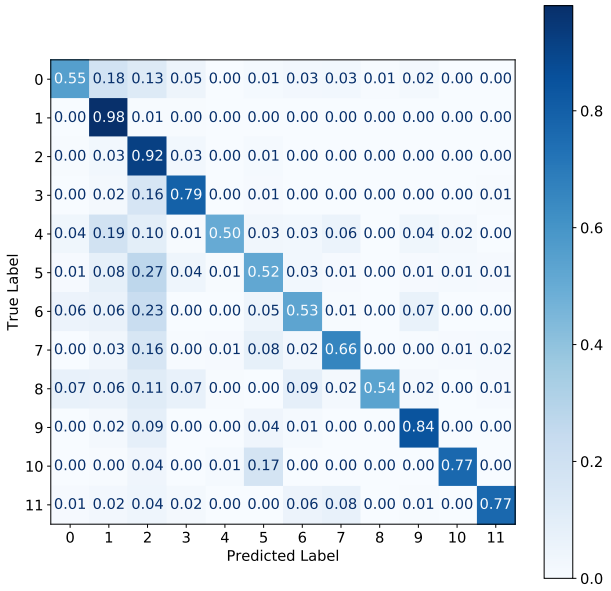


Fig. 7: Confusion Matrix of STM-DT on IPS Stratosphere 5a.

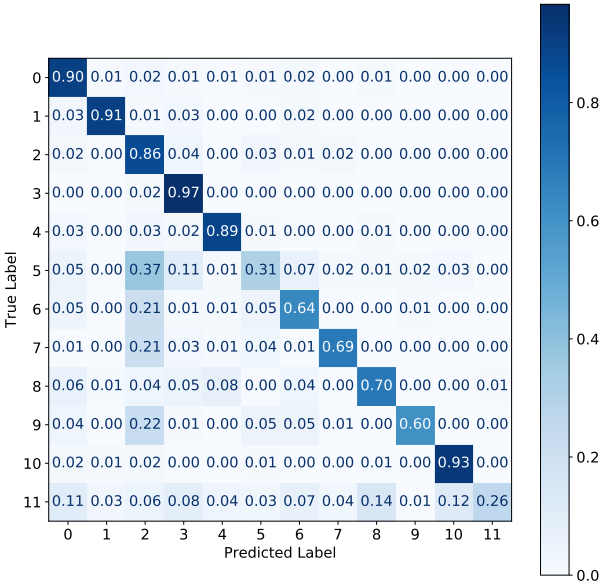


Fig. 8: Confusion Matrix of STM-DT on IPS Stratosphere 5b.

labeled data and unlabeled data during training, but noisy label exists in unlabeled data. Noisy label removing means we reduce the number of these noisy labels. `Mixup` means we mix labeled data and unlabeled data during training. The results are shown in Table 10. The ratio of labeled data on the CIC-IDS2017 is 0.2%. The ratio of labeled data on the Kitsune is 0.05%. And the ratio of labeled data on the Stratosphere IPS 5a and 5b is 2%. We observe that the F1 score of STM-DT with self-training improves by around 1.4% compared to STM-DT without self-training and `mixup` on the CIC-IDS2017 and Kitsune datasets. Self-training improves a little on the two Stratosphere IPS datasets due to noisy labels. After reducing noisy labels with NLR, the F1 score significantly increases because it provides sufficient data with pseudo labels for training. It is worth mentioning that the F1 score grows 3.5% on the Stratosphere IPS 5a

TABLE 9: Training time (in minutes) on datasets

Dataset	Method	Training Time
CIC-IDS2017	Mean Teacher	1607.25
	Mixmatch	1636.00
	Π-model	1617.03
	STM-DT	**4.53**
Kitsune	Mean Teacher	4003.15
	Mixmatch	3782.67
	Π-model	4017.60
	STM-DT	**36.32**
Stratosphere IPS 5a	Mean Teacher	90.00
	Mixmatch	88.90
	Π-model	87.08
	STM-DT	**0.11**
Stratosphere IPS 5b	Mean Teacher	125.68
	Mixmatch	125.13
	Π-model	124.15
	STM-DT	**0.16**

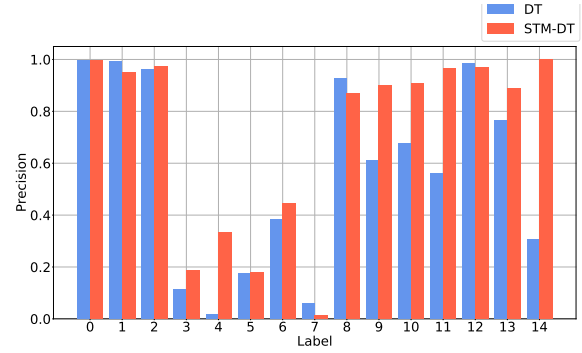
dataset. With the help of the `mixup`, STM-DT obtains higher performance than STM-DT with ST and NLR on all three datasets except the Kitsune dataset. Mixup decreases the performance on Kitsune because the number of unlabeled samples on Kitsune is much more than the other three datasets, and `mixup` may generate more samples with noisy labels that affect the performance of the models. This experimental result shows that all three components contribute to the performance of our proposed method.

STM-DT without self-training, noisy label removal, and `mixup` is the same as the decision tree (DT). Fig. 9 shows the comparison in precision for each class between DT and STM-DT. The X-axis is the class label and Y-axis is the precision of each class on the test data. The results show that the proposed method increases the precision of most minor classes such as classes 3, 4, 6, 9, 10, 11, 13, and 14. For instance, on the CIC-IDS2017 dataset, the precision of class 4 is improved from lower than 5% to around 36%. There is also an improvement in precision for several classes on Kitsune, Stratosphere IPS 5a and 5b datasets.

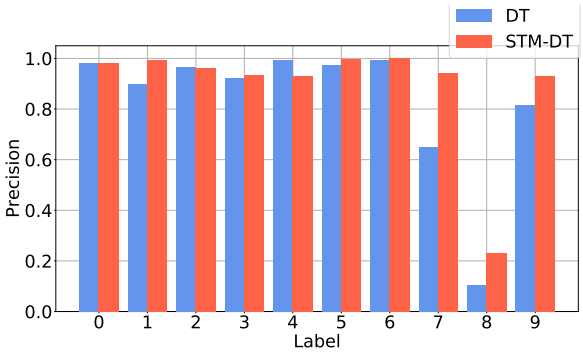
We compare the performance of DT and STM-DT for different ratios of labeled training data. Fig. 10 shows the detailed performance of DT and STM-DT on various datasets. The X-axis is the ratio of labeled training data and Y-axis is the F1 score of the model on the test data. As the number of samples in the labeled training data increases, the F1 score of both DT and our proposed STM-DT rises. When the ratio of labeled data reaches 100%, there is no unlabeled data to train STM-DT. So there is no F1 score of STM-DT when the ratio is 100% in Fig. 10. On the CIC-IDS2017 dataset and Stratosphere IPS 5a dataset, the performance curve of STM-DT is above the DT. On the Stratosphere IPS 5b dataset, the performance of STM-DT is better than DT when the ratio is less than 20% and quite close to DT when the ratio is more than 20%. It is because the degree of class imbalance is less in the CIC-IDS2017 and the Stratosphere IPS 5a datasets. Therefore, the number of samples of minor classes is enough for DT when the ratio increases slightly. On the Kitsune dataset, STM-DT

TABLE 10: Ablation study results

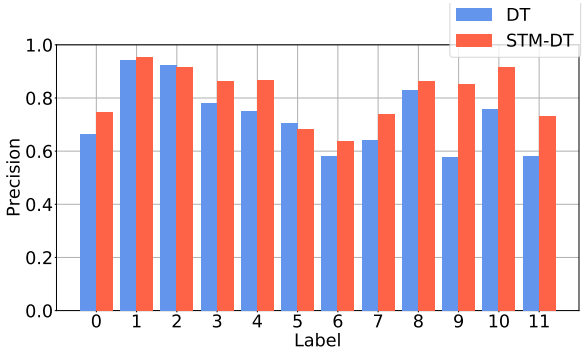
Dataset	Ablation	F1-macro	Precision-macro	Recall-macro
CIC-IDS2017	STM-DT without ST, NLR and M	0.644 ± 0.021	0.601 ± 0.028	0.842 ± 0.028
	STM-DT with ST only	0.659 ± 0.019	0.634 ± 0.016	0.820 ± 0.029
	STM-DT with ST and NLR	0.678 ± 0.042	$\mathbf{0.654 \pm 0.053}$	0.833 ± 0.013
	STM-DT	$\mathbf{0.685 \pm 0.019}$	0.650 ± 0.021	$\mathbf{0.844 \pm 0.015}$
Kitsune	STM-DT without ST, NLR and M	0.827 ± 0.009	0.812 ± 0.015	0.911 ± 0.013
	STM-DT with ST only	0.840 ± 0.004	0.836 ± 0.011	0.916 ± 0.007
	STM-DT with ST and NLR	$\mathbf{0.867 \pm 0.007}$	$\mathbf{0.861 \pm 0.012}$	$\mathbf{0.920 \pm 0.009}$
	STM-DT	0.859 ± 0.022	0.860 ± 0.031	0.883 ± 0.021
Stratosphere IPS 5a	STM-DT without ST, NLR and M	0.681 ± 0.012	0.735 ± 0.011	0.661 ± 0.017
	STM-DT with ST only	0.687 ± 0.015	0.747 ± 0.020	0.656 ± 0.014
	STM-DT with ST and NLR	0.722 ± 0.042	0.773 ± 0.025	0.693 ± 0.021
	STM-DT	$\mathbf{0.742 \pm 0.012}$	$\mathbf{0.791 \pm 0.015}$	$\mathbf{0.715 \pm 0.016}$
Stratosphere IPS 5b	STM-DT without ST, NLR and M	0.726 ± 0.011	0.756 ± 0.014	0.705 ± 0.011
	STM-DT with ST only	0.729 ± 0.006	0.765 ± 0.009	0.705 ± 0.005
	STM-DT with ST and NLR	0.745 ± 0.007	0.792 ± 0.006	0.719 ± 0.006
	STM-DT	$\mathbf{0.749 \pm 0.012}$	$\mathbf{0.802 \pm 0.018}$	$\mathbf{0.720 \pm 0.007}$



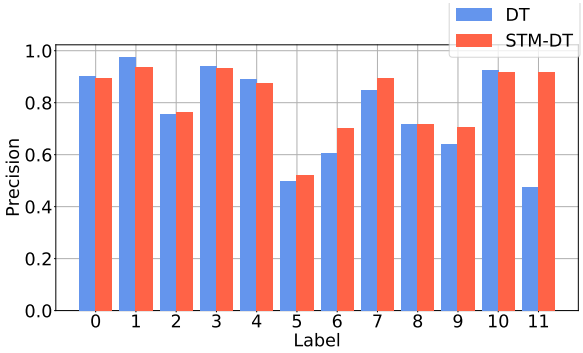
(a) CIC-IDS2017 dataset



(b) Kitsune dataset



(c) Stratosphere IPS 5a dataset



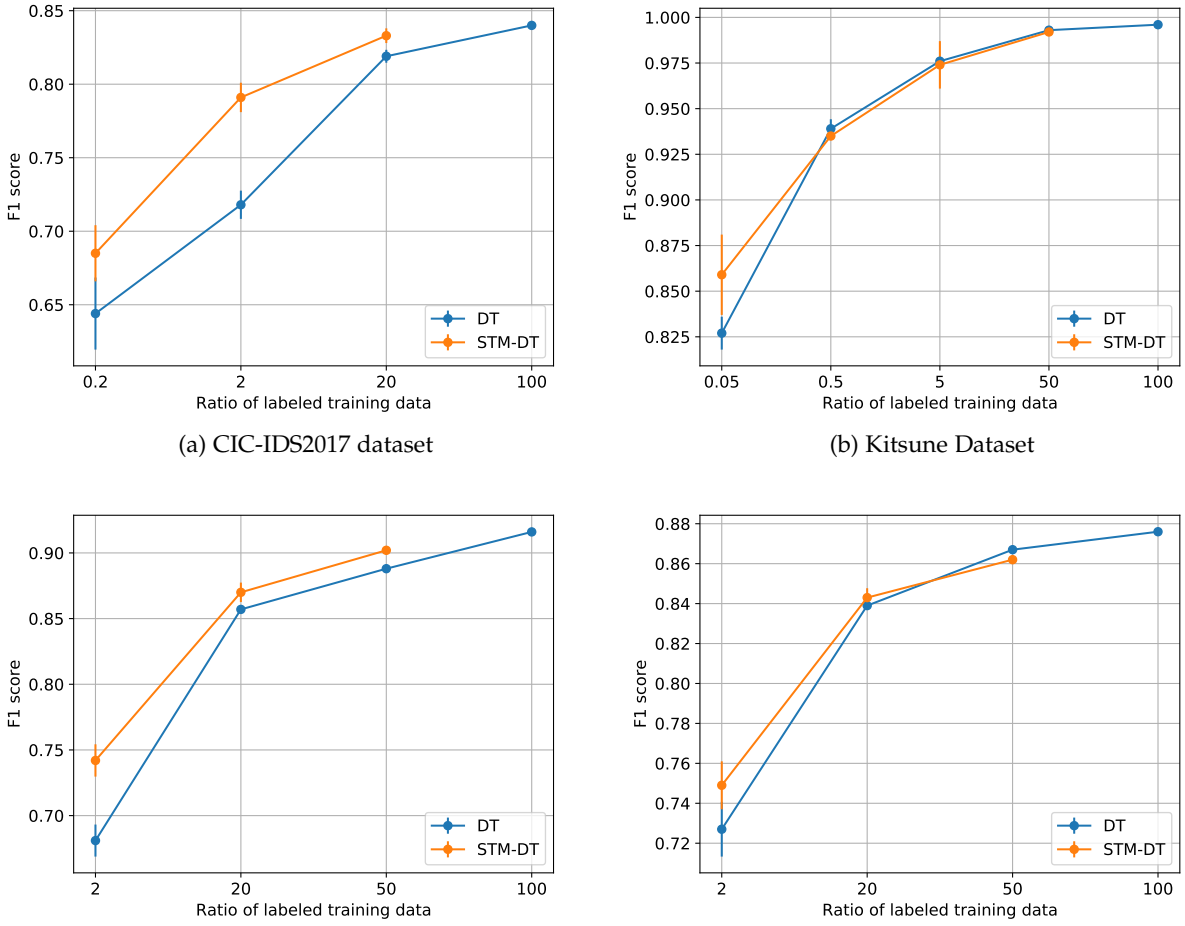
(d) Stratosphere IPS 5b dataset

Fig. 9: Precision for each class on different datasets.

outperforms DT when the ratio is less than 0.5%. There are more than 20,000,000 samples in the Kitsune dataset while the Stratosphere IPS 5b dataset consists of around 350,000 samples. Due to the large number of samples, DT can be trained with sufficient labeled data, even though the ratio increases slightly. That explains why the performance of the proposed STM-DT is close to DT when the ratio is larger than 0.5%. In summary, the proposed STM-DT outperforms DT when there is insufficient labeled training data, which reflects the realistic scenario of computer networks.

5 Related Work

Network traffic attack is the main threat in cyber-security. There are a lot of methods trying to provide solutions for this. The rule-based method captures attacks by a set of rules, which is mature in the industry. In [36], the authors proposed a rule-based method for anomaly detection. This kind of method is explainable but limited to more and more sophisticated attacks. The rule-based method also could not detect new types of attacks caused by different user behavior or software updates. Motivated by the success of AI, there have been many approaches using machine learning and deep learning. In [37], the authors analyzed



(a) CIC-IDS2017 dataset

(b) Kitsune Dataset

(c) IPS Stratossphere 5a dataset

(d) IPS Stratossphere 5b dataset

Fig. 10: F1 score with different ratio of labeled training data on different datasets. When ratio of labeled training data is 100%, STM-DT is the same as DT.

the NetFlow records by Random Forest to detect command-and-control servers. In [38], the authors proposed a supervised learning method based on recurrent neural networks (RNN). In [39], the authors combined CNN and LSTM by fusion to detect intrusions. In [40], the authors built an effective intrusion detection framework, which utilized an SVM classifier with feature embedding produced by Naive Bayes. These supervised learning approaches require vast amounts of labeled data for the training process. It is time-consuming to prepare such labeled data in practice.

To address the issue of supervised learning approaches, unsupervised learning approaches have been developed. In [41], a shallow autoencoder (AE) with a nearest-neighbor classifier was used to filter false positives. In [42], a kernel-based learning algorithm was developed to detect online DDoS attacks. In [43], the authors proposed an unsupervised deep learning method which consists of a Convolutional Neural Network and an autoencoder for auto-profiling the traffic patterns and filtering abnormal traffic. In [44], a variational autoencoder (VAE)-based unsupervised learning method was developed to detect anomalies in network traffic using reconstruction error. In [45], the authors proposed an anomaly detection method that adopts a bi-directional GANs to enable fast detection and stabilize the GAN training process using multiple improvements. In [8], two GAN-based methods were adopted for unsupervised network anomaly detection. All AE, VAE and GAN-based methods share the same assumption, which is most training samples are normal samples. After training, these methods are expected to reconstruct normal samples well. If the input sample is anomalous, it cannot be reconstructed well, allowing these methods to detect anomalies by the reconstruction error between input and output samples. Although unsupervised learning approaches do not require labeled data, they cannot be adopted for multi-class classification of network attacks. The classification of network attacking traffic is helpful for cyber-security researchers to learn about these attacks better and enhance the security of systems. Even with a limited amount of labeled data available in practice, such unsupervised learning approaches cannot fully utilize these labeled data to address the problem of network attack classification.

Motivated by the limitations of the mentioned approaches, several semi-supervised learning approaches are proposed for network traffic attacks. In [46], the authors proposed a clustering-based semi-supervised machine learning for DDoS attack classification. It applies agglomerative clustering and K-means on unlabeled data first. Then a voting

method is used to label the data and obtain classes to distinguish attacks from normal traffic. After labeling, supervised machine learning approaches are trained for classification. In [47], the authors proposed a semi-supervised K-means approach using a hybrid feature selection algorithm to detect DDoS. In [48], the authors proposed a semi-supervised machine learning approach based on network entropy estimation, co-clustering, information gain ratio, and extra-trees for DDoS detection. These semi-supervised learning approaches are only for binary classification problems and may not work well for multi-class classification problems. Our proposed semi-supervised learning method STM-DT can address the aforementioned issues and achieve a good classification performance.

6 Conclusion

In this paper, we proposed a semi-supervised learning method, known as Self-Training Mixup Decision Tree (STM-DT), for network attack detection and classification. It uses a self-training mechanism for data annotation by utilizing labeled data to train a decision tree for data annotation of unlabeled data. Then, it marks unlabeled data by the well-trained decision tree and filters some noisy labels by the consistency between two decision trees. One decision tree is trained on labeled data and the other is trained on the mixed labeled data generated by `mixup`. The extensive experiments on four network traffic datasets show that the proposed STM-DT is efficient and effective compared to benchmark semi-supervised learning methods with a small amount of labeled data. This significantly reduces the effort of data labeling as well as training resources (i.e., training time and computing resources) while maintaining the performance of classification models. Since data annotation of unlabeled data causes noisy labels, we will further investigate multiple methods, including deep learning to improve the performance of removing noisy labels.

References

[1] M. Uma and G. Padmavathi, "A survey on various cyber attacks and their classification." *Int. J. Netw. Secur.*, vol. 15, no. 5, 2013.

[2] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[3] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[4] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Transactions on Networking*, vol. 3, no. 3, pp. 261–271, 2020.

[5] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *IEEE ICDM 2008*, 2008, pp. 413–422.

[6] E. Seo, H. M. Song, and H. K. Kim, "Gids: Gan based intrusion detection system for in-vehicle network," in *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, 2018, pp. 1–6.

[7] Y. Hou, Z. Chen, M. Wu, C.-S. Foo, X. Li, and R. M. Shubair, "Mahalanobis distance based adversarial network for anomaly detection," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3192–3196.

[8] T. Truong-Huu, N. Dheenadhayalan, P. Pratim Kundu, V. Ramnath, J. Liao, S. G. Teo, and S. Praveen Kadiyala, "An Empirical Study on Unsupervised Network Anomaly Detection Using Generative Adversarial Networks," in *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, Taipei, Taiwan, Oct. 2020, p. 20–29.

[9] J. Liao, S. G. Teo, P. Pratim Kundu, and T. Truong-Huu, "ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, Rhodes, Greece, July 2021, pp. 81–88.

[10] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[11] J. E. Van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, no. 2, pp. 373–440, 2020.

[12] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 542–542, 2009.

[13] X. Zhu, "Semi-supervised learning literature survey," *world*, vol. 10, p. 10, 2005.

[14] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proc. IEEE/CVF CVPR 2020*, 2020.

[15] Y. Ouali, C. Hudelot, and M. Tami, "An overview of deep semi-supervised learning," *CoRR*, vol. abs/2006.05278, 2020. [Online]. Available: https://arxiv.org/abs/2006.05278

[16] J. Wang, L. Perez *et al.*, "The effectiveness of data augmentation in image classification using deep learning," *Convolutional Neural Networks Vis. Recognit*, vol. 11, pp. 1–8, 2017.

[17] K. Killamsetty, D. S, G. Ramakrishnan, A. De, and R. Iyer, "GRAD-MATCH: Gradient Matching based Data Subset Selection for Efficient Deep Model Training," in *Proceedings of the 38th International Conference on Machine Learning*, Jul. 2021, pp. 5464–5474.

[18] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[19] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[20] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[21] S. H. Yoo, H. Geng, T. L. Chiu, S. K. Yu, D. C. Cho, J. Heo, M. S. Choi, I. H. Choi, C. Cung Van, N. V. Nhung *et al.*, "Deep learning-based decision-tree classifier for covid-19 diagnosis from chest x-ray imaging," *Frontiers in medicine*, vol. 7, p. 427, 2020.

[22] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[23] ——, *C4.5: programs for machine learning*. Elsevier, 2014.

[24] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Routledge, 2017.

[25] L. E. Raileanu and K. Stoffel, "Theoretical comparison between the gini index and information gain criteria," *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004.

[26] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *Advances in neural information processing systems*, vol. 30, 2017.

[27] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017, pp. 1–13. [Online]. Available: https://openreview.net/forum?id=BJ6oOfqge

[28] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," *arXiv preprint arXiv:1905.02249*, 2019.

[29] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

[30] E. Min, J. Long, Q. Liu, J. Cui, Z. Cai, and J. Ma, "Su-ids: A semi-supervised and unsupervised framework for network intrusion detection," in *International Conference on Cloud Computing and Security*. Springer, 2018, pp. 322–334.

[31] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[32] 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.

[33] Stratosphere, "Stratosphere laboratory datasets," 2015, retrieved March 13, 2020, from https://www.stratosphereips.org/datasets-overview.

[34] P. P. Kundu, T. Truong-Huu, L. Chen, L. Zhou, and S. G. Teo, "Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–15, 2022.

[35] A. Ghosh, N. Manwani, and P. Sastry, "On the robustness of decision tree learning under label noise," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2017, pp. 685–697.

[36] N. Duffield, P. Haffner, B. Krishnamurthy, and H. Ringberg, "Rule-based anomaly detection on ip flows," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 424–432.

[37] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, "Disclosure: detecting botnet command and control servers through large-scale netflow analysis," in *Proc. 28th Annual Computer Security Applications Conference*, 2012, pp. 129–138.

[38] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21 954–21 961, 2017.

[39] R. Yao, N. Wang, Z. Liu, P. Chen, and X. Sheng, "Intrusion detection system in the advanced metering infrastructure: a cross-layer feature-fusion cnn-lstm-based approach," *Sensors*, vol. 21, no. 2, p. 626, 2021.

[40] J. Gu and S. Lu, "An effective intrusion detection approach using svm with naïve bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021.

[41] G. Kathareios, A. Anghel, A. Mate, R. Clauberg, and M. Gusat, "Catch it if you can: Real-time network anomaly detection with low false alarm rates," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2017, pp. 924–929.

[42] S. D. Çakmakçı, T. Kemmerich, T. Ahmed, and N. Baykal, "Online ddos attack detection using mahalanobis distance and kernel-based learning algorithm," *Journal of Network and Computer Applications*, vol. 168, p. 102756, 2020.

[43] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection," *IEEE Access*, vol. 8, 2020.

[44] Q. P. Nguyen, K. W. Lim, D. M. Divakaran, K. H. Low, and M. C. Chan, "Gee: A gradient-based explainable variational autoencoder for network anomaly detection," in *IEEE CNS 2019*, 2019.

[45] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *IEEE ICDM 2018*, 2018, pp. 727–736.

[46] M. Aamir and S. M. A. Zaidi, "Clustering based semi-supervised machine learning for ddos attack classification," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 4, pp. 436–446, 2021.

[47] Y. Gu, K. Li, Z. Guo, and Y. Wang, "Semi-supervised k-means ddos detection method using hybrid feature selection algorithm," *IEEE Access*, vol. 7, pp. 64 351–64 365, 2019.

[48] M. Idhammad, K. Afdel, and M. Belouch, "Semi-supervised machine learning approach for ddos detection," *Applied Intelligence*, vol. 48, no. 10, pp. 3193–3208, 2018.

Yubo Hou is a research engineer at Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. He received the Master degree in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, in 2017. He has won several competitive awards, such as First Place Winner for CVPR 2021 UG2+ Challenge, Third Place Winner for ICASSP 2021 COVID-19 Diagnosis Challenge, etc. His research interests include data analytics, machine learning, anomaly detection, domain adaption and related application.

Sin G. Teo is a Scientist at Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore. Currently, he is a principal investigator of several projects that blends the domains of Artificial Intelligence and Cybersecurity, and leads a team of A.I. for Cybersecurity in the Cybersecurity department. He obtained the Ph.D. degree from Monash University, Australia in 2016. His research interests include applied cryptography, data privacy and security, malware and network anomaly classification, federated learning, and deep learning.

Zhenghua Chen received the B.Eng. degree in mechatronics engineering from University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2011, and Ph.D. degree in electrical and electronic engineering from Nanyang Technological University (NTU), Singapore, in 2017. He has been working at NTU as a research fellow. Currently, he is a Scientist and Lab Head at Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He has won several competitive awards, such as First Place Winner for CVPR 2021 UG2+ Challenge, A*STAR Career Development Award, First Runner-Up Award for Grand Challenge at IEEE VCIP 2020, Finalist Academic Paper Award at IEEE ICPHM 2020, etc. He serves as Associate Editor for Elsevier Neurocomputing and Guest Editors for five journals. He is currently the Vice Chair of IEEE Sensors Council Singapore Chapter and IEEE Senior Member. His research interests include smart sensing, data analytics, machine learning, transfer learning and related applications.

Min Wu is currently a senior scientist in Data Analytics Department, Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore. He received his Ph.D. degree in Computer Science from Nanyang Technological University (NTU), Singapore, in 2011 and B.S. degree in Computer Science from University of Science and Technology of China (USTC) in 2006. He received the best paper awards in InCoB 2016 and DASFAA 2015. He also won the IJCAI competition on repeated buyers prediction in 2015. His current research interests include machine learning, data mining and bioinformatics.

Chee-Keong Kwoh received the bachelor's degree in electrical engineering (first class) and the master's degree in industrial system engineering from the National University of Singapore, Singapore, in 1987 and 1991, respectively. He received the Ph.D. degree from the Imperial College of Science, Technology, and Medicine, University of London, in 1995. He has been with the School of Computer Engineering, Nanyang Technological University (NTU), since 1993. He is the Deputy Executive Director of PaCE at NTU. His research interests include data mining, soft computing and graph-based inference; application areas include bioinformatics and engineering. He has done significant research work in his research areas and has published many quality international conferences and journal papers. He is a member of the Association for Medical and Bio-Informatics, Imperial College Alumni Association of Singapore. He has provided many services to professional bodies in Singapore and was conferred the Public Service Medal by the president of Singapore in 2008.

Tram Truong-Huu is an Assistant Professor at the Singapore Institute of Technology (SIT), Infocomm Technology (ICT) Cluster. He is currently holding a joint appointment with Agency for Science, Technology and Research (A*STAR), Singapore, where he has worked as a computer scientist at Institute for Infocomm Research (I^2R) since May 2019. He received the Ph.D. degree in computer science from the University of Nice - Sophia Antipolis (now Côte d'Azur University), France in December 2010. From January 2011 to June 2012, he held a post-doctoral fellowship at the French National Center for Scientific Research (CNRS), France. He worked at the National University of Singapore as a research fellow from July 2012, then senior research fellow from January 2017. His research interests include software-defined networks, Internet of Things, and application of artificial intelligence to cybersecurity. He won the Best Presentation Recognition at IEEE/ACM UCC 2013. He is a member of the IEEE since 2012 and Senior Member of the IEEE since 2015.