Malware Traffic Classification Using Domain Adaptation and Ladder Network for Secure Industrial Internet of Things

Jinhui Ning, *Graduate Student Member, IEEE*, Guan Gui, *Senior Member, IEEE*, Yu Wang, *Graduate Student Member, IEEE*, Jie Yang, *Member, IEEE*, Bamidele Adebisi, *Senior Member, IEEE*, Song Ci, *Senior Member, IEEE*, Haris Gacanin, *Fellow, IEEE*, and Fumiyuki Adachi, *Life Fellow, IEEE*

***Abstract*—Malware traffic classification (MTC) is a key technology for anomaly and intrusion detection in secure Industrial Internet of Things (IIoT). Traditional MTC methods based on port, payload, and statistic depend on the manual-designed features, which have low accuracy. Recently, deep-learning methods have attracted a significant attention due to their high accuracy in terms of classification. However, in practical application scenarios, deep-learning methods require a large amount of labeled samples for training, while the available labeled samples for training are very rare. Furthermore, the preparation of a large amount of labeled samples requires a lot of labor costs. To solve these problems, this article proposes three methods based on semisupervised learning (SSL), transfer learning (TL), and domain adaptive (DA), respectively. Our proposed methods use a large amount of unlabeled data collected in the Internet traffic, which can greatly improve the classification accuracy with few labeled samples. Then, we use the DA method to solve the mismatch problem between the source domain and the target domain in the TL process. The proposed method is not only applicable to the shallow network but also to the deep neural network structure, and can achieve better classification results. Experimental results show that our proposed methods can satisfy the requirement of MTC in the case of few labeled samples in IIoT. The source code for all the experiments is available at GitHub.[1]**

***Index Terms*—Convolutional neural network, domain adaptation (DA), malware traffic classification (MTC), semisupervised learning (SSL), transfer learning (TL).**

Manuscript received 29 August 2021; revised 19 October 2021; accepted 29 November 2021. Date of publication 2 December 2021; date of current version 7 September 2022. This work was supported in part by the Summit of the Six Top Talents Program of Jiangsu under Grant XYDXX-010; in part by the Program for High-Level Entrepreneurial and Innovative Team under Grant CZ002SC19001; in part by the Project of the Key Laboratory of Universal Wireless Communications (BUPT) of Ministry of Education of China under Grant KFKT-2020106; and in part the State Key Laboratory Power System and Generation Equipment of Tsinghua University under Grant SKLD20Z02. This article was presented in part titled "A novel malware traffic classification method using semi-supervised learning" at the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), online, 27–30 September 2021, pp. 1–5. *(Corresponding author: Guan Gui.)*

Jinhui Ning, Guan Gui, and Yu Wang are with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China (e-mail: 1220013514@njupt.edu.cn; guiguan@njupt.edu.cn; 1018010407@njupt.edu.cn).

Jie Yang is with the College of Telecommunications and Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China, and also with the Shaanxi Key Laboratory of Information Communication Network and Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, Shaanxi, China (e-mail: jyang@njupt.edu.cn).

Bamidele Adebisi is with the Department of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester M1 5GD, U.K. (e-mail: b.adebisi@mmu.ac.uk).

Song Ci is with the Department of Electrical Engineering, Tsinghua University, Beijing 100084, China (e-mail: sci@tsinghua.edu.cn).

Haris Gacanin is with the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, 52062 Aachen, Germany (e-mail: harisg@ice.rwth-aachen.de).

Fumiyuki Adachi is with the Research Organization of Electrical Communication, Tohoku University, Sendai 980-8577, Japan (e-mail: adachi@ecei.tohoku.ac.jp).

Digital Object Identifier 10.1109/JIOT.2021.3131981

I. Introduction

SECURE and reliable communications are considered as one of the key requirements for the Industrial Internet of Things (IIoT) [1]–[7]. The malware traffic classification (MTC) technology plays a significant role in the field of network security and IIoT [8]. In recent years, with the rapid development of IIoT, there have also been a proliferation of various applications, including smart factory, portals, weibo, video, and group buying software. Although these have made our lives much better to a certain extent, there has also been a remarkable increase in network security problems, such as hacker attacks, data leaks, and so on. Software traffic can record and reflect network operating conditions [9]–[11]. To ensure network security of future IIoT, it is necessary to identify malware traffic and prevent the occurrence of various attack incidents [12], [13]. For the problem of identifying malware traffic, various methods have been proposed. Traditional network traffic classification methods are usually based on supervised methods, with large amounts of labeled samples. Wu *et al.* [14] first proposed an effective feature-based learning method to classify offloading data and detect an anomaly event for ubiquitous IoT applications. The proposed method can reduce the computation overhead and energy consumption. However, there are few labeled samples in real life. Hence, supervised methods are not efficient in achieving IIoT performance requirements in terms of latency and reliability.

To solve the problem with a lack of labeled samples in supervised learning [15], [16], semisupervised learning (SSL) [17]–[19] is considered in this article. SSL combines

[1]The code of this article can be downloaded from GitHub link: https://github.com/yzjh/Keras-MTC-DA-Ladder.

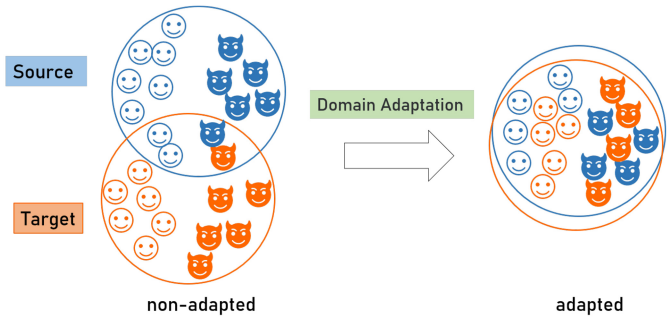

Fig. 1. Schematic of feature distribution of source domain and target domain before and after DA. The smile face represents the normal network traffic, and the demon represents the malware traffic. The blue part is the feature distribution of the source domain, and the orange part is the feature distribution of the target domain.

supervised learning and unsupervised learning, by training small amounts of labeled data and large amounts of unlabeled data, simultaneously. In this article, we propose the ConvLaddernet method based on SSL to improve the classification accuracy with a limited number of labeled samples. The proposed method employs the Laddernet [36] as a classifier. In reality, for the problem of malware traffic identification, there is only a small amount of labeled data and a large amount of unlabeled data. We adopt transfer learning (TL) [20] to improve classification accuracy. Leveraging the knowledge from the relevant domains with abundant labeled samples, TL has a significant advantage in solving the scarcity problem of the labeled samples in the target domain. In [21], the TL for emotion recognition (TLER) model under Cloud–Edge–Client collaborations is proposed, providing an efficient solution to solve the acquisition and marking problems of electroencephalogram (EEG) signals. Pan and Yang [22] divided TL into three different types, i.e., inductive TL, transductive TL, and unsupervised TL. In the case of inductive TL, the target task is different from the source task. In the case of transduction TL, the source task and the target task are the same, and the source domain and the target domain are different. While in the case of unsupervised TL, the target task and the source task are different but related. For the data set used in this article, we adopt the transductive TL method. Furthermore, the knowledge transfer ConvLaddernet (KT-ConvLaddernet) method is proposed based on TL to improve the classification accuracy of fewer labeled samples. Specifically, we adopt convolutional autoencoder (CAE) [23] and random forest (RF) [24] for benchmarks.

Furthermore, in order to address the problem of the mismatch between the source domain and the target domain, we use domain adaptation (DA) ideas in MTC, as shown in Fig. 1. By combining DA with proposed KT-ConvLaddernet methods, we propose the KTDA-ConvLaddernet method. Finally, because the shallow network is relatively weak in feature extraction, we use the deep VGG network [25] as the feature extraction layer. Our experimental results show that the proposed method has high classification accuracy on the USTC-TFC2016 data set [26]. The main contributions of this article are summarized as follows.

1) A ConvLaddernet method of MTC is proposed to solve the problem of low classification accuracy in the case of few (less than 20% of total labeled samples) labeled samples.
2) An MTC method based on KT-ConvLaddernet is proposed to solve the problem of low classification accuracy in the case of fewer (less than 5% of total labeled samples) labeled samples.
3) We propose KTDA-CNN and KTDA-Convladdernet methods to solve the mismatch problem between the source domain and the target domain in the TL process.

II. Related Works

A. Traditional MTC Methods

Three traditional methods for MTC that have emerged [27]: 1) port-based method; 2) payload-based or deep packets inspection (DPI)-based method; and 3) statistical-based method. Li *et al.* [28] proposed to use the port table assigned by the Internet Assigned Number Authority (IANA) to solve the classification problem, but the accuracy was very low, only 30%. Sen *et al.* [29] proposed to check available documents and packets and identify the signature of applications to accurately track Peer-to-Peer (P2P) traffic. Shafiq *et al.* [30] used network traffic capture tools to capture data and used feature extraction tools to extract features from traffic data. Recently, deep learning has been extensively applied to the field of communication and signal recognition [31]–[33] and achieves good results. Deep learning is a powerful tool to directly obtain features from raw traffic data [15], [34], [35] and can solve the problem of low precision in manual-designed methods to a certain extent. Wu *et al.* proposed a privacy-preserving edge task assignment framework (PETA) in [9], using the Gaussian mixture model and group signatures to cluster, manage, and verify, achieving a balance between task matching accuracy and user privacy. Wang *et al.* [26] proposed the application of representation learning to network traffic data classification for the first time. In the 20-classes classification, this method achieved 99.17% of the classification effect, and the accuracy even reached 100% in the binary classification. Although the deep-learning method has achieved excellent results in MTC, the disadvantage of the deep-learning method is that it often requires a large amount of labeled data for training, which is inconsistent with the few sample situation, so it is difficult to directly apply in the actual MTC project.

B. SSL and Its Applications

SSL is a branch of machine learning, most of which is used for classification problems. SSL uses a small amount of labeled data and a large amount of unlabeled data for training at the same time, and the trained model can achieve an effect close to the model trained with a large amount of labeled data. SSL is well suited for situations where there is a lack of labeled data, such as computer-aided diagnosis, drug discovery, and part-of-speech tagging. In recent years, many SSL methods have emerged. Lee [19] proposed the pseudolabel method in which the network's prediction of unlabeled data was regarded as the label of unlabeled data. It

is simple to train the network with pseudolabels and reduce the overfitting of the network under the limited data with labels. However, the prediction capability is poor at the initial stage of the network, which leads to the low accuracy of pseudolabels. Rasmus *et al.* [36] proposed a semisupervised ladder network to solve the problem compartibility between supervised learning and unsupervised learning, and an end-to-end semisupervised deep model was developed. Tarvainen and Valpola [37] proposed the mean teacher method to obtain the teacher model through moving average of model parameters, and then the teacher model was used to construct high-quality target data, which finally achieved good results. In addition to the above SSL methods, some studies have used SSL in dynamic network scenarios. Qian *et al.* [38] proposed the deep reinforcement learning (DRL)-based online algorithm for the dynamic channel scenario to efficiently learn the near-optimal offloading solutions for the time-varying channel realizations.

C. DA and Its Applications

DA has been extensively studied in many fields, including computer vision applications [39], [40], natural language processing [41], image recognition [42]–[44], etc. In the field of image recognition, most of the data sets need manual feature description as model input, but the actual situation of marking features manually is very complex. The emergence of representation learning [45] reduces the dependence on manual features. Ghifary *et al.* [46] proposed a denoising self-coding pretraining algorithm. Then, the maximum-mean discrepancy (MMD) was used to measure the distance between two distributions in the regenerative Hilbert space, and the two-layer network was simultaneously trained. The neural network adopted was called DaNN. This method can effectively learn the domain-invariant representation. However, due to the shallow network and limited representation capability, it cannot effectively solve the DA problem. Tzeng *et al.* [47] proposed a new network called deep domain confusion (DDC) on the basis of the original AlexNet network to the network fc7 layer (a classifier in the previous layer) and by adding an adaptation layer. The idea of DDC is changing the shallow network to the deep network for training in order to get better results. Long *et al.* [48] proposed the deep adaptation networks (DANs) structure on the basis of DDC. The DDC network only adapts one classification layer, while DAN adapts multiple layers. DDC uses single-kernel MMD, but a single kernel may not be the optimal core. DAN uses a multi-kernel MMD (MK-MMD) to get a better result than the DDC network.

In summary, MTC has achieved many excellent results, especially the MTC method based on deep learning is able to achieve high performance. However, the existing deep-learning-based MTC methods require a large number of labeled samples for training, and the use of few sample training still needs to be explored. Therefore, based on the previous studies on SSL and TL-based DA, we focus on the use of few samples to train high-performance MTC networks.

III. System Model and Data Preprocessing

A. System Model

First, we preprocess our original software traffic data. For a group of labeled software traffic samples $\{x_r(n), y_r(n)|1 < n < N\}$, the CNN classification is expressed as

$$\hat{y} = z(\Theta, x_r, y_r) \tag{1}$$

where $z(\cdot)$ represents the feature extraction function, which is fitted or realized by CNN. Θ represents the network parameter; $\hat{y}$ is the classified prediction result of the network. y_r denotes the true labels. In the backward propagation optimization, its objective function can be expressed as

$$J = \sum_r L_r\big(y_r, \hat{y}_r\big) \tag{2}$$

where L_r is the loss function to measure the difference of true labels y_r and predicted labels $\hat{y}_r$.

In the real small samples environment, the classification method used by CNN is inefficient due to the lack of training data. Therefore, we use the SSL method to make full use of the large amount of unlabeled data that exist in the network $\{x_u(m)|N < m < M \ \ (N << M)\}$. The overall objective function of the network can be expressed as

$$J = \sum_r L_r\big(y_r, \hat{y}_r\big) + \lambda \sum_u L_u(y_u) \tag{3}$$

where L_u represents the loss function of the unsupervised part and y_u is the predicted value of x_u in the network.

In SSL, the purpose of the network's supervised part and the unlabeled part is not matched, hence the semisupervised method using (3) cannot achieve good performance. Inspired by Laddernet [36], we add the Gaussian noise to the coding layer of an encoder and set the noise transformation as g, then $\tilde{z}_e = g(z_e)$. Using $\tilde{z}_e$ and z_e to process labeled and unlabeled data, respectively. Then, in order to optimize each layer of the network, we calculate an L_u for each layer to optimize. For any lth layer in CNN, we modify the objective function as

$$J = \sum_r L_r(y_r, \tilde{z}_e(x_r)) + \lambda \sum_{l=1}^{L} \sum_{i=1}^{N^{(l)}} L_u\Big(\tilde{y}_r^{(l)}, z_e^{(l)}(x_u)\Big) \tag{4}$$

where L is the total number of network layers and $N^{(l)}$ denotes the amount of characteristic data at each layer. Inputting the encoder output $z_e^{(l)}$ into the lth layer decoder $z_d^{(l)}$ to get the decoder output as $\tilde{y}_r^{(l)}$. The system model is shown in Fig. 2.

In order to make full use of some existing software traffic data sets, we propose a transfer-SSL. First, the existing network traffic data set is preprocessed and then trained according to (2) to obtain the CNN z, which can be used to extract network traffic features. Then, we remove the classification layer of z and make it output characteristic data as follows:

$$f = z_f(\Theta_{\text{conv}}, x, y) \tag{5}$$

where f is the feature extracted by CNN. z_f represents the CNN without the header layer (i.e., the full connection layer

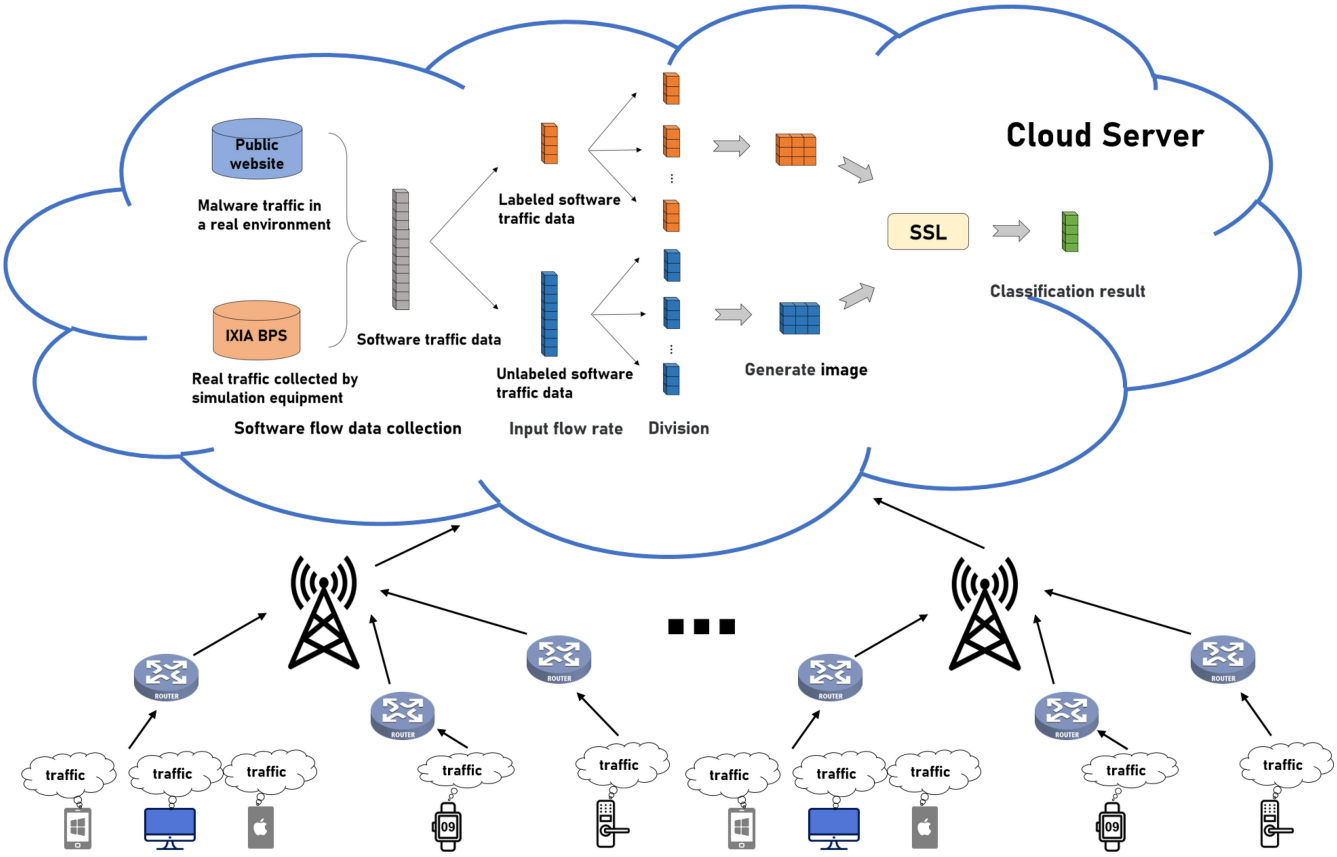

Fig. 2. System model. The various computers and mobile phones in the picture generate network traffic, which is collected by the router and sent to the base station. The base station sends the data to the cloud server, where the data are processed. The left-hand side of the cloud server shows the basic situation of the data set, where the orange part represents the labeled data, and the blue part represents the unlabeled data. The data is processed and converted into image formats, respectively, and the SSL method is used to obtain the classification results.

and the softmax). Θ_{conv} is the parameter of z_f. We use z_f to process unlabeled data obtained from the network and few labeled data. Then, the unlabeled feature f_u and labeled feature f_r are acquired. Taking feature f_u and feature f_r as input data, the classification is expressed as

$$\hat{y} = z_{\text{TL}}\left(\Theta'_{\text{conv}}, \Theta_{fc}, f_r, f_u, y\right) \tag{6}$$

where z_{TL} represents the transfered network, Θ_{fc} is the network parameters of the new classification layer, and Θ'_{conv} means that Θ_{conv} is frozen so that it can be updated. By optimizing z_{TL} according to (4), our classification objective can be achieved.

TL is sometimes faced with the problem that the distribution of the source domain and target domain is quite different. At this time, the simple use of the pretraining model to extract the features of the target domain cannot provide a good effect. The fine-tuning method is generally used to improve this problem. The fine-tuning method is to freeze only part or even none of the feature extraction layer of the pretraining model and retrain it with a small learning rate together with the new classification layer, so that the pretraining model can absorb some knowledge of the target domain. This approach has some impressive features, but sometimes results in performance degradation. In order to get better performance, we use DA to improve the above problems.

We use the feature-based DA method, which is an unsupervised adaptive method. The source domain samples and the target domain samples are adjusted to the same feature space by a mapping φ, so as to realize alignment. Hence, the objective function is formulated as

$$J_a = \min \frac{1}{N}\sum_{i=0}^{N} L_a\left(\varphi\left(x_i^s\right), \varphi\left(x_i^t\right)\right) \tag{7}$$

where N is the sample size, x_i^s and x_i^t are the samples of the source domain and the target domain, and L_a is the adaptive loss function. Here, L_a can use the KL divergence, H-divergence, or MMD, etc. The MMD algorithm is chosen in this article. The idea is to find a kernel function, map both the source domain and the target domain to the Hilbert space of the regenerated kernel, and take the difference of the two domain data after the mean value, respectively, in this space, and then take it as the distance loss. The MMD algorithm is given as

$$\text{MMD} = \frac{1}{N}\sum_{i=1}^{N}\left\|\varphi\left(x_i^s\right) - \varphi\left(x_i^t\right)\right\|_H \tag{8}$$

where H refers to the Hilbert space. φ mapping is a kernel function, usually a Gaussian kernel. We focus on the adaptation of the feature extraction layer after TL. Hence, the samples of our source domain and target domain are the

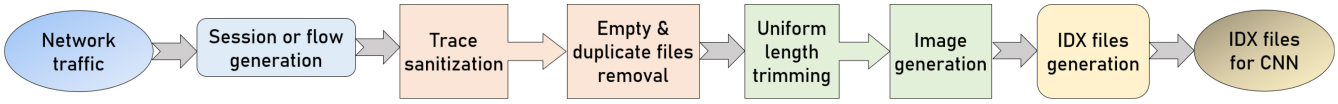

Fig. 3. Procedure of data preprocess. Perform a series of processing on the collected raw data to get the flow data that can be used directly.

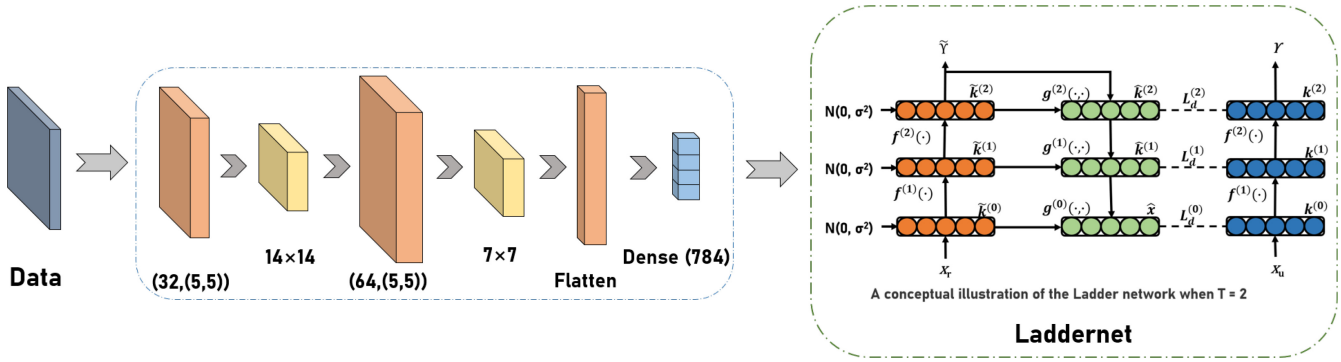

Fig. 4. Architecture of ConvLaddernet. A neural network structure is composed of a convolutional layer and a Laddernet. The input data size is 28×28, the number of channels in the first layer of convolution is 32, the size of the convolution kernel is 5×5, and it passes through the first layer of convolution. After the data size becomes 14×14, the second convolutional layer has 64 channels, the size of the convolution kernel is still 5×5, and the data of 7×7 size is obtained, and the final classification result is obtained after the flatten and Laddernet.

features output by different feature extraction layers. We train a new pretraining feature extraction layer z'_f, and then do adaptive training with the previously finished training feature extraction layer z'_f. Inputing few labeled samples x_r into the two data feature extraction layers, the output characteristics are f_a and f_{pre}, respectively. The total loss function after using MMD is formulated as

$$L = L_c(x, y) + \mu \text{MMD}\big(\varphi(f_{\text{pre}}), \varphi(f_a)\big) \tag{9}$$

where L_c is the classification loss of the supervised or semisupervised method, and μ is the adaptive loss weight. Using this loss function, training can make the trained model inherit the performance of the pretrained model to the maximum extent and adapt to the field of few labeled samples at the same time.

B. Data Preprocessing

The data set used in the experiment is USTC-TFC2016 [26]. The data set roughly consists of two parts, one part is the ten types of malware traffic collected by researchers from the real network environment, and the other part is the ten types of normal traffic collected by the professional network traffic simulation equipment IXIA BPS. After the software data are obtained from the application software traffic, the data are sliced, collated to prevent duplicate data, generated the image with an input size of 784 bytes, and then converted to the IDX format. The procedure of data preprocess is shown in Fig. 3.

IV. Our Proposed Methods

A. Proposed ConvLaddernet-Based SSL-MTC Method

1) Structure of ConvLaddernet: The structure of the ConvLaddernet-based SSL-MTC method is shown in Fig. 4. The ConvLaddernet method consists of the CNN and the Laddernet. The processed training data are fed into the convolutional layer for training. After data features are extracted, the features are inputted into the Laddernet for data classification and the classification results are finally obtained.

2) Basic Principle of Laddernet: Laddernet is an SSL method. When there are a large amount of unlabeled data x_u and a small amount of labeled data x_r, the mathematical form of Laddernet can be expressed as

$$\left\{\tilde{k}^{(0)}, \ldots, \tilde{k}^{(T-1)}\right\},\ \tilde{\gamma} = \text{Encoder}_{\text{noisy}}(x_r) \tag{10}$$

$$\left\{k^{(0)}, \ldots, k^{(T-1)}\right\},\ \gamma = \text{Encoder}_{\text{clean}}(x_u) \tag{11}$$

$$\left\{\hat{k}^{(0)}, \ldots, \hat{k}^{(T-1)}\right\} = \text{Decoder}\left(\tilde{k}^{(0)}, \ldots, \tilde{k}^{(T-1)}\right) \tag{12}$$

where $\text{Encoder}_{\text{noisy}}$ is the coding layer with Gaussian noise and $\text{Encoder}_{\text{clean}}$ represents the coding layer without noise (clean). Decoder denotes the decoding layer; k represents the output of each intermediate layer; $\tilde{k}$ is the intermediate layer output of the noise channel; and $\hat{k}$ refers to the corresponding output of the clean channel; T denotes the number of Laddernet's layer; and $\tilde{\gamma}$ and γ represent the final output with and without noise, respectively.

For the supervised learning methods, the cross-entropy of the true label and the predicted value is usually used to represent the loss. The Laddernet uses the noise channel $\tilde{\gamma}$ and true label y, as well as the output of each layer of $\text{Encoder}_{\text{clean}}$ and Decoder, to jointly construct the objective function, and its objective function is given as

$$J_{\text{Laddernet}} = -\sum_{n=1}^{N} \log P(\tilde{\gamma}(n) = y(n) | x_r(n)) + \lambda_l \sum_{n=1}^{M} \sum_{l=0}^{L-1} \left\| |k^{(l)}(n) - \hat{k}^{(l)}(n) \right\|^2 \tag{13}$$

where x is input data. M and N are the number of unlabeled data and the number of labeled data, respectively. λ_l represents the weight of the each decoding layer's loss function.

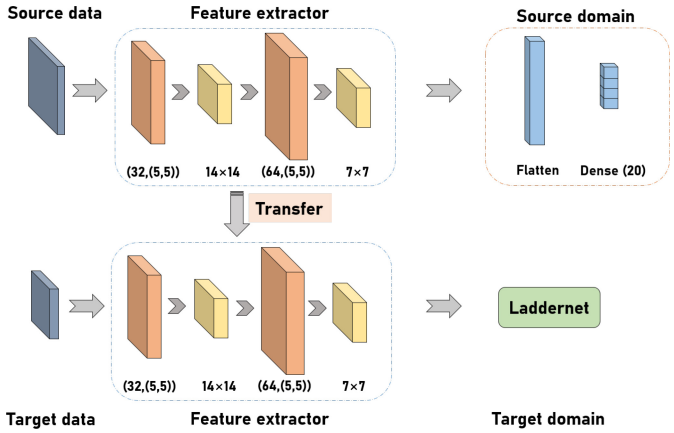

Fig. 5. Architecture of KT-ConvLaddernet. Using a transferred network structure, where the convolutional layer is the same as the ConvLaddernet method, a new classification layer is connected to it in the source domain, and the softmax activation function is used for 20 classifications. Transferred the trained feature extraction layer to the target domain and freeze it, then connect to Laddernet for classification.

3) Training and Test Process of ConvLaddernet: Before training, the data set is divided into a training part and a validation part for cross-validation, and then a small sample part is separated as labeled data in the training part, and the rest of the data is used as unlabeled data. We use the Adam optimizer, the learning rate is set to 0.02, and the categorical cross-entropy function is selected as the loss function. Other parameters, including the maximum epoch and batch size, are set to 50 and 128, respectively. After training, we evaluate the effectiveness of our model using the test set.

B. Proposed KT-ConvLaddernet-Based SSL-MTC Method

1) Structure of KT-ConvLaddernet: The structure of the KT-ConvLaddernet-based SSL-MTC method is shown in Fig. 5. The main difference between ConvLaddernet and KT-ConvLaddernet is that KT-ConvLaddernet uses the TL method. First, inputting the source domain data to the pretrain network. After the completion of pretraining, the feature extractor is transferred to the target domain and frozen. Then, the target domain data is input, and the feature extractor is connected to the Laddernet for classification. The KT-ConvLaddernet-based MTC method process is shown in Algorithm 1.

2) Training and Testing Process of KT-ConvLaddernet: In this method, we first train the pretraining network using the pretraining set and then divide the data set according to Section IV-A3 to complete the transfer training. The maximum epoch during pretraining and transfer training is set to 50, and the rest of the experimental settings is consistent with Section IV-A3.

C. Proposed KTDA-Based SSL-MTC Methods

1) Basic Principle of DA: DA is a method that is widely used in TL to solve the problem of few samples. It is used when the source domain and the target domain have the same task, but different distribution, and the target domain has only few labeled samples compared with the source domain. Due to the different distributions of the source domain and the target domain, the classification boundary of the classifier trained directly in the source domain cannot sufficiently distinguish the samples in the target domain, and the few labeled samples in the target domain make it unable to fully train the target domain model, leading to a phenomenon also referred to as overfitting. As shown in Fig. 1, the idea of DA is not to use the source domain model to process the target domain data directly, but to align the sample features of the source domain and the target domain as far as possible when training the target domain model, to train a powerful target domain model. The method of DA can be divided into the following three: 1) feature-based adaptation; 2) instance-based adaptation; and 3) model parameter-based adaptation. Presently, the most widely used DA method is feature-based adaptation.

2) KTDA-CNN Method: The structure of the KTDA-CNN-based SSL-MTC method is shown in Fig. 6. The KTDA-CNN method consists of two convolution layers and two classification layers. The main difference between this method and KT-ConvLaddernet is that it no longer uses the direct transfer method, but uses the DA method to transfer the necessary knowledge between the pretrained feature extraction layer and the new network. First, we use source domain data to train a pretrained convolutional layer for feature extraction from source domain data. Due to the large gap between the data distribution of the source domain and the target domain, we do not directly perform TL, but use the target domain data to retrain a convolutional layer with the same structure. But because the target domain has only few samples, direct training will lead to serious overfitting, so we set the MMD

Algorithm 1: Training Process of the Proposed KT-ConvLaddernet-Based MTC Method

Input: pre-training samples H_{pre}; labeled re-training samples H'_r; unlabeled re-training samples H'_u; pre-training hyperparameters θ_{pre}; re-training hyperparameters θ_{re}; maximum epoches $E_{\max}$.
Output: predictive value $\hat{H}/\hat{H}'$.

[Training stage]:
Randomly initialize the parameters θ_{pre}
for $i = 1, 2, \ldots, E_{\max}$ **do**
  $\hat{H} \leftarrow CNN(H_{pre})$
  $Loss \leftarrow J_{pre}(H_{pre}, \hat{H})$
  Update θ_{pre} with $Loss$
end
Save θ_{pre}.
[Re-training stage]:
Randomly initialize the parameters θ_{re}
Load and **freeze** θ_{pre}
Remove header: $CNN' \longleftarrow (CNN, \theta_{pre})$
for $i = 1, 2, \ldots, E_{\max}$ **do**
  $f_r, f_u \leftarrow CNN'(H'_r, H'_u)$
  $\hat{H}' \leftarrow \text{ConvLaddernet}(f_r, f_u)$
  $Loss \leftarrow J_{re}(H'_r, \hat{H}')$
  Update θ_{re} with $Loss$
end
Save θ_{re}.

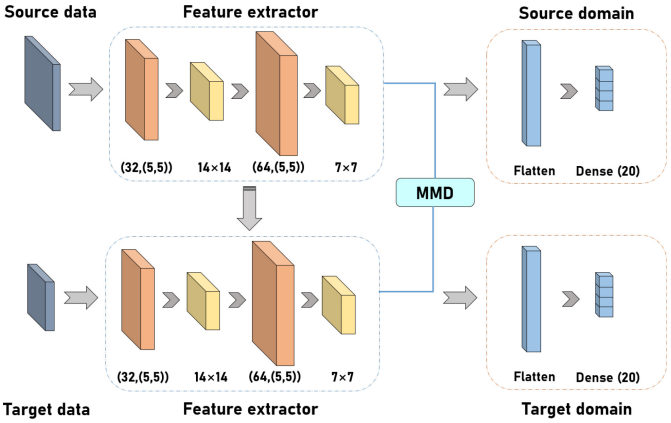

Fig. 6. Architecture of KTDA-CNN. The MMD distance is added after the ordinary convolutional layer, and the dense layer of 20 classifications is added after the trained feature extraction layer.

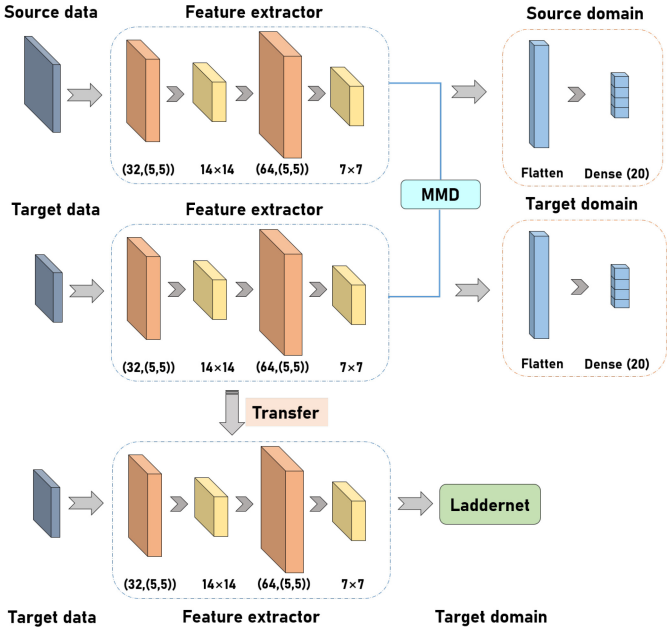

Fig. 7. Architecture of KTDA-ConvLaddernet. The feature extraction layer is trained by adding the MMD distance after the convolutional layer, and the dense layer is used for classification. Using the transfer method, the trained feature extraction layer is transferred to the target domain and frozen, followed by Laddernet for classification.

loss between the pretrained convolutional layer and the new training convolutional layer, so that the features of the new convolutional layer can be close to the pretrained convolutional layer. After the training in the target domain, it is classified at the classification layer.

The method first uses the pretraining data set in the source domain to pretrain the network. The data set is also divided into the training part and the cross-validation part. The training data in the target domain is a large number of unlabeled data and different percentages of labeled data, so as to complete the transfer training. The loss function (9) is given in Section III. We also need to set the DA loss weight μ. Too large μ will lead to overregularization, but too small will cause MMD to have no effect on the result. In the experiment, the regularization hyperparameter μ is set to 0.25. The rest of the experimental settings is consistent with Section IV-A3. After the training process, we use the test set to evaluate the model.

3) Proposed KTDA-ConvLaddernet Method: The structure of the KTDA-ConvLaddernet-based SSL-MTC method is shown in Fig. 7. The KTDA-ConvLaddernet method added the DA algorithm on the basis of the original KT-ConvLaddernet method. Due to the high complexity of the ladder network, it is more difficult to optimize and train the ladder network than to train other simple classification networks. If the feature extraction layer is trained at the same time in this process, even if the MMD loss is added, the optimization process will be difficult to proceed smoothly, and the training will easily fall into a local optimal solution or cause gradient explosion. Therefore, different from the two-stage training of TL and KTDA-CNN, we design a three-stage training method. First, we follow the KTDA-CNN process to train a feature extraction layer adapted to the target domain, and then we directly use this new feature extraction layer for TL, i.e., directly freeze the new feature extraction layer and use it in the training process of the ladder network. After the feature extraction layer is well trained, the ladder network is trained in the target domain. In this way, the network is fully trained and the classification accuracy is improved. The KTDA-ConvLaddernet-based MTC method process is shown in Algorithm 2.

The training of KTDA-ConvLaddernet is divided into three steps. First, the feature of the source domain and the target domain is matched by adding the MMD measurement to obtain the extraction layer with close features. After the trained feature extraction layer is transferred to the target domain, the few labeled samples are inputted for classification, and the classification results are obtained through the ladder network. The parameter settings in the experiment are consistent with Section III.

D. Deep Layers MTC Methods

In the previous study, we adopted a relatively simple CNN network structure as the feature extraction layer. However, due to the shallow network and limited characterization capability, it is impossible to accurately solve the problem of classifying malware traffic. Therefore, in this article, the deep VGG network is proposed to replace the shallow CNN network to achieve higher classification accuracy. On the basis of the method proposed in this article, KTDA-VGG and KTDA-VGGLaddernet methods are also presented.

E. Benchmark Methods

1) CNN-Based Methods: We train a simple CNN [26] network to classify software traffic data. It consists of one feature extraction module with two convolutional layers, its filters are 32 and 64, respectively. Dropout is used to prevent overfitting in each convolutional layer. Then, features are compressed through two max-pooling layers. Finally, after the extracted features are flattened, a fully connected layer with activation function softmax is used to classify the data.

Algorithm 2: Training Process of the Proposed KTDA-ConvLaddernet-Based MTC Method

Input: pre-training samples H_{pre};
labeled re-training samples H_r;
unlabeled re-training samples H_u.
Parameters: pre-training hyperparameters θ_{pre};
parameters for adaptation θ_a,
re-training hyperparameters θ_{re};
maximum epoches $E_{\max}$.
Output: predictive value $\hat{H}/\hat{H}'$.

[Training stage]:
Randomly initialize the parameters θ_{pre}
for $i = 1, 2, \ldots, E_{\max}$ **do**
  $f_{pre} \leftarrow CNN_{pre}(H_{pre})$
  $\hat{H}_{pre} \leftarrow \text{Classifier}(f_{pre})$
  $Loss \leftarrow J_{pre}(H_{pre}, \hat{H}_{pre})$
  Update θ_{pre} with $Loss$
end
Save θ_{pre}.
[Domain adaptation stage]:
Randomly initialize the parameters θ_a
for $i = 1, 2, \ldots, E_{\max}$ **do**
  $f_{pre} \leftarrow CNN_{pre}(H_r)$
  $f_a \leftarrow CNN_a(H_r)$
  $Loss_{MMD} \leftarrow MMD(f_{pre}, f_a)$
  $\hat{H}_a \leftarrow \text{Classifier}(f_a)$
  $Loss_a \leftarrow J_a(H_a, \hat{H}_a)$
  $Loss \leftarrow Loss_a + \mu Loss_{MMD}$
  Update θ_a with $Loss$
end
Save θ_a.
[Re-training stage]:
Randomly initialize the parameters θ_{re}
Load and **freeze** θ_a
for $i = 1, 2, \ldots, E_{\max}$ **do**
  $f_r, f_u \leftarrow CNN_a(H_r, H_u)$
  $\hat{H}' \leftarrow \text{ConvLaddernet}(f_r, f_u)$
  $Loss \leftarrow J_{re}(H_r, \hat{H}')$
  Update θ_{re} with $Loss$
end
Save θ_{re}.

2) CAE With RF Classifier: The structure of the CAE-RF method is shown in Fig. 8. We first train an autoencoder with all unlabeled software traffic data. The structure of the autoencoder is

$$h = z_e(\Theta_e, x) \tag{14}$$

$$\hat{x} = z_d(\Theta_d, h) \tag{15}$$

where z_e and z_d are, respectively, the encoding and decoding parts of the CAE; Θ_e and Θ_d denote the network parameters; and h is the output of the encoder. It contains all the features of the raw data. Subsequently, we send the small sample data into the trained autoencoder to obtain the features h output by the encoder. Features h are used as labeled data to train the classifier. The software traffic encoded by the autoencoder has key features, which can improve the training performance of RF in the case of few labeled samples.

3) Joint Transferring and Classification Method: In order to verify the rationality of our KTDA-ConvLaddernet method, which is also named transferring then classification (TTC) method, we propose a joint transferring and classification (JTC) method. In the second stage of this method, the DA method is used to train the feature extraction network, and at the same time, the Laddernet for classification is trained with a few samples.

TABLE I
SIMULATION PARAMETERS

Parameter	Value	
Dataset	USTC-TFC2016	
Input data dimension	$28 \times 28 \times 1$	
The number of training samples	120,000	
The proportion of labeled samples	case I: {1%, 5%, 10%, 15%, 20%} case II: {1%, 2%, 3%, 4%, 5%}	
Training hyperparameters	Device	Geforce GTX 1080Ti
	Environment	Python 3.6.2
		Keras 2.2.4
	Optimizer	Adam
	Batch-size	128
	Learning rate	0.02
	DA loss weight	0.25
	Epoches	{50, 100}

V. SIMULATION RESULTS AND DISCUSSION

A. Experimental Setup

Our server uses Geforce GTX 1080ti GPU with 11-GB RAM for computing. The environment is the Keras 2.2.4 deep-learning framework and Python 3.6.2. We use the tools in sklearn 0.23.2 to evaluate our model, and the RF model used in the experiment is also called sklearn. The detailed simulation parameters are listed in Table I.

B. Experimental Results Based on ConvLaddernet and KT-ConvLaddernet

In this section, we first compare our proposed ConvLaddernet, KT-ConvLaddernet, and CAE-RF with CNN, KT-CNN, and RF in terms of accuracy in the case of $\{1\%, 5\%, 10\%, 15\%, 20\%\}$ with labeled samples, respectively, and then compare the simulation results in the case of $\{1\%, 2\%, 3\%, 4\%, 5\%\}$ fewer labeled samples of ConvLaddernet and KT-ConvLaddernet.

As shown in Table II, the methods with knowledge TL (KT-CNN and KT-ConvLaddernet) obtain better performance in the case of fewer samples (1%–5%), and methods of direct feature extraction (CNN and ConvLaddernet) achieve better results when the number of labeled samples gradually increased (5%–20%), e.g., when the proportion of labeled samples is 1%, the classification accuracy of the original CNN is 0.68% lower than that of the KT-CNN method, but it is 1.07% higher than the latter when the sample proportion is 20%. The reasons for this phenomenon are as follows. In

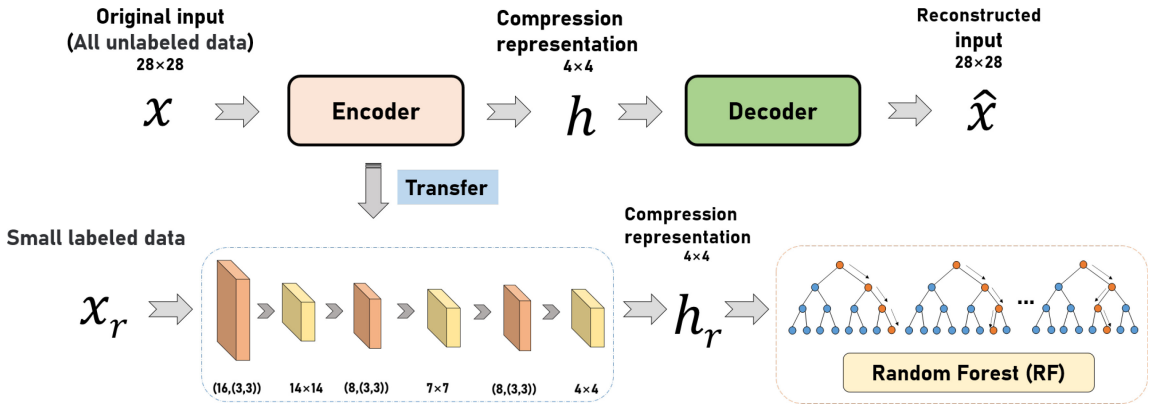

Fig. 8. Architecture of CAE-RF. A neural network structure composed of the convolutional layer, encoder, decoder, and RF.

TABLE II
ACCURACY COMPARISONS OF THE PERCENTAGE OF LABELED SAMPLES WITH DIFFERENT METHODS

Methods \ Percentage	1%	5%	10%	15%	20%
CNN	0.8141	0.9260	0.9567	0.9613	0.9763
KT-CNN	0.8209	0.9250	0.9381	0.9669	0.9656
ConvLaddernet	0.9086	0.9771	0.9826	0.9832	0.9855
KT-ConvLaddernet	0.9326	0.9733	0.9741	0.9746	0.9786
RF	0.7439	0.8300	0.8467	0.8540	0.8605
CAE-RF	0.8790	0.9198	0.9238	0.9280	0.9378
KT-VGG	0.9134	0.9677	0.9677	0.9715	0.9738
KT-VGGLaddernet	0.9669	0.9870	0.9876	0.9887	0.9919
KTDA-CNN	0.9502	0.9775	0.9826	0.9829	0.9880
KTDA-VGG	0.9756	0.9908	0.9929	0.9938	0.9971
KTDA-ConvLaddernet	0.9582	0.9810	0.9812	0.9835	0.9859
KTDA-VGGLaddernet	0.9764	0.9946	0.9962	0.9964	0.9968

the case of fewer samples, the knowledge learned from the source domain data is helpful to the data training in the target domain. However, when the training data of the target domain increases, the features extracted by the feature extraction layer of the target domain are more consistent with the training data and have better classification performance, while the features extracted in the source domain are no longer suitable for the classification of the target domain. Therefore, the performance of TL will be worse in this case.

According to the results in Table II, we carry out an additional experiment for the case when the proportion of labeled samples is 1%–5% in Fig. 9. When the proportion of labeled samples is smaller (less than 3.8%), the performance of KT-ConvLaddernet is better than ConvLaddernet. When the proportion of labeled samples is greater than 3.8%, the Convladdernet method has a good result. It is also proved that knowledge TL is more suitable for fewer samples. Then, we compare our methods with the benchmark methods. CAE-RF has a higher classification accuracy than RF, but its classification accuracy is not as good as using the CNN network directly. The original CNN network uses a fully connected layer for classification, after using Laddernet for classification, the classification accuracy increases significantly. In the case of a sample proportion of 1%, the classification accuracy of using Laddernet is 9.45% higher than that of the original

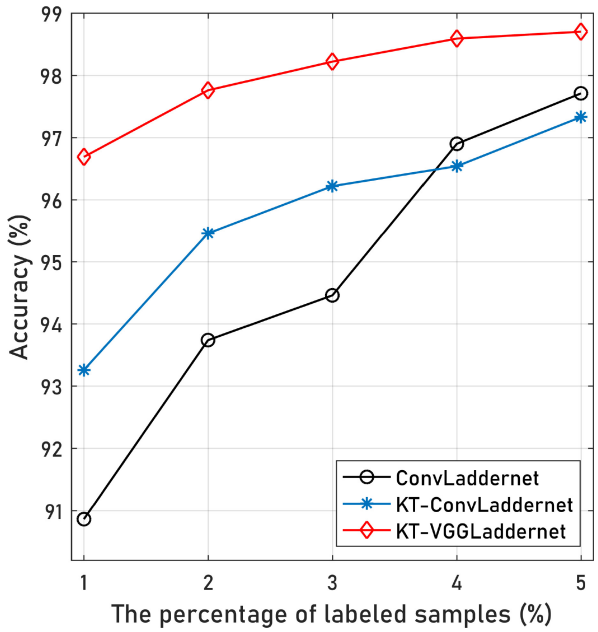

Fig. 9. Accuracy of the proposed methods with fewer labeled samples.

CNN, which proves the effect of ConvLaddernet in the case of fewer samples.

C. Experimental Results Based on KTDA Methods

In this section, we first compare the performance of the KTDA method with the previously proposed ConvLaddernet and KT-ConvLaddernet methods in the cases of few samples (5%–20%) and fewer samples (1%–5%), and experiments are also carried out for the deep VGG network. Subsequently, we compare the performance of KTDA-ConvLaddernet with the comparison method JTC. As shown in Table II, the KTDA-CNN method has higher classification accuracy than KT-CNN in the case of few samples and fewer samples. The highest case of few samples and fewer samples is 12.93% (sample proportion of 1%) and 4.45% (sample proportion of 10%). The classification accuracy of the KTDA-ConvLaddernet method in the cases of few samples and fewer samples is also higher than that of the KT-ConvLaddernet and ConvLaddernet methods. At a sample proportion of 1%, KTDA-ConvLaddernet

TABLE III
ACCURACY OF THE PERCENTAGE OF LABELED SAMPLES WITH DIFFERENT STRUCTURES

Percentage	KTDA-ConvLaddernet	JTC-ConvLaddernet	KTDA-VGGLaddernet	JTC-VGGLaddernet
1%	0.9582	0.7866	0.9764	0.6644
2%	0.9691	0.8537	0.9857	0.6329
3%	0.9725	0.8420	0.9848	0.6505
4%	0.9749	0.8699	0.9901	0.6347
5%	0.9810	0.8752	0.9946	0.7142
10%	0.9812	0.8548	0.9962	0.6878
15%	0.9835	0.8682	0.9964	0.6755
20%	0.9859	0.8754	0.9968	0.6592

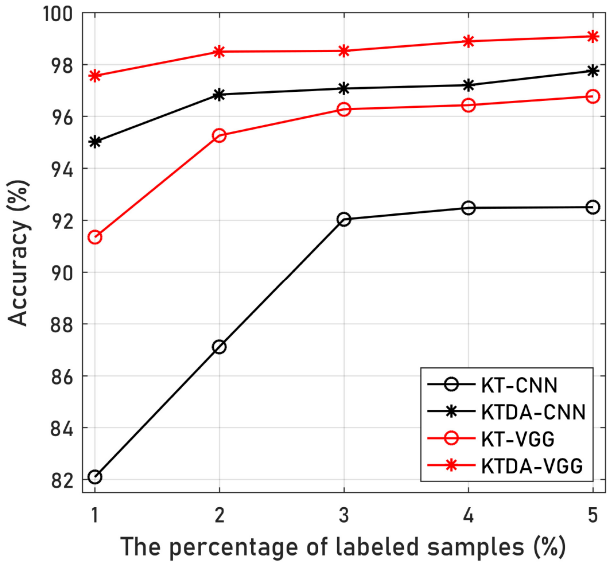

Fig. 10. Accuracy of the percentage of labeled samples with KTDA methods.

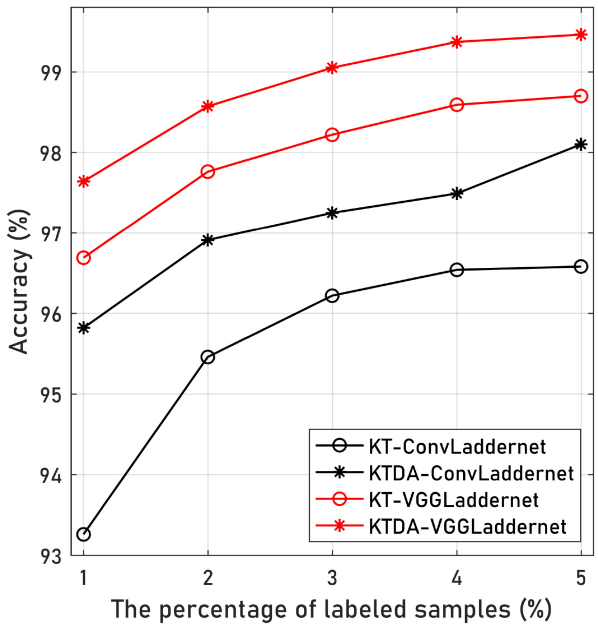

Fig. 11. Accuracy of the percentage of labeled samples with KTDA and Laddernet methods.

is 2.56% higher than KT-ConvLaddernet. At a sample proportion of 15%, KTDA-ConvLaddernet is 0.89% higher than KT-ConvLaddernet. In summary, the KTDA method has significantly improved performance compared to the KT-ConvLaddernet and ConvLaddernet methods, and its performance is significantly improved when the sample size is fewer.

Figs. 10 and 11 show the effect of using the shallow network and deep VGG network on classification accuracy. It can be seen that in the various methods proposed, the use of the deep VGG network can improve the classification performance of the model. With a fewer sample percentage, this improvement is more obvious, e.g., with a labeled sample proportion of 1%, the deep feature extraction network can increase the performance of KT by about 4%–8%, and the performance of KTDA can also increase by about 1%–2%. Finally, we compare the performance of KTDA-ConvLaddernet and the comparison method JTC-ConvLaddernet in the cases of few samples and fewer samples, as shown in Table III. It can be observed that the classification accuracy of the JTC-ConvLaddernet method is much lower than that of the KTDA-ConvLaddernet method with the same setting, regardless of the percentage of labeled samples. In addition, the performance of JTC-ConvLaddernet when using a deep network is lower than when using a shallow network. This is because the complexity of the two-stage network is too high, which makes the training unable to effectively converge, and replacing it with a deep network further increases the complexity of the network, which will further damage the training effect.

VI. CONCLUSION

In this article, we first proposed two MTC methods based on ConvLaddernet and KT-ConvLaddernet, respectively, for the applications of secure IIoT. The two cases are based on few labeled samples and fewer labeled samples, respectively. The ConvLaddernet-based MTC method achieves better performance with few labeled samples. While the classification performance of the KT-ConvLaddernet-based MTC method is better in the case of fewer labeled samples. Moreover, the CAE-RF method was proposed as a comparison. The experimental results show that in the case of fewer labeled samples, extracting sample features before training can achieve higher classification accuracy. To further improve the performance, we combined the DA method with

the proposed KT-ConvLaddernet method to improve the mismatch problem between the source domain and the target domain in the TL process and proposed an MTC method based on KTDA-ConvLaddernet. Based on both few and fewer samples, experimental results showed that the classification accuracy of the proposed method is better than ConvLaddernet and KT-ConvLaddernet methods. After replacing the shallow CNN network with a deeper VGG network, the classification performance of the proposed model can be further improved for the secure IIoT.

The models used in this article are common models in the field of image recognition, and generally have a high degree of complexity. Therefore, when performing some specific tasks, the required accuracy and computing power should be measured, and the model should be simplified appropriately. For example, the complexity of using VGG as a feature extraction layer is higher than that of using simple CNN, and the complexity of KTDA-ConvLaddernet is also higher than that of KTDA-CNN, although it has higher accuracy than the latter. In addition, the TL-based methods we proposed all require pretraining models, which are sometimes difficult to implement in practical applications. In future work, we will further explore the MTC in a few sample scenarios and optimize our method to solve the above problems.

REFERENCES

[1] J. Ning, Y. Wang, J. Yang, H. Gacanin, and S. Ci, "A novel malware traffic classification method using semi-supervised learning," in *Proc. IEEE 94th Veh. Technol. Conf. (VTC-Fall)*, Sep. 2021, pp. 1–5.

[2] Z. Su *et al.*, "Secure and efficient federated learning for smart grid with edge-cloud collaboration," *IEEE Trans. Ind. Informat.*, vol. 18, no. 2, pp. 1333–1344, Feb. 2022, doi: 10.1109/TII.2021.3095506.

[3] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network toward 6G: Machine-learning approaches," *Proc. IEEE*, vol. 108, no. 2, pp. 292–307, Feb. 2020.

[4] W. Hou, H. Wen, N. Zhang, J. Wu, W. Lei, and R. Zhao, "Incentive-driven task allocation for collaborative edge computing in Industrial Internet of Things," *IEEE Internet Things J.*, early access, May 31, 2021, doi: 10.1109/JIOT.2021.3085143.

[5] M. Wang, Y. Lin, Q. Tian, and G. Si, "Transfer learning promotes 6G wireless communications: Recent advances and future challenges," *IEEE Trans. Rel.*, vol. 70, no. 2, pp. 790–807, Jun. 2021.

[6] N. Zhang *et al.*, "Physical-layer authentication for Internet of Things via WFRFT-based Gaussian tag embedding," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9001–9010, Sep. 2020.

[7] G. Gui, M. Liu, F. Tang, N. Kato, and F. Adachi, "6G: Opening new horizons for integration of comfort, security, and intelligence," *IEEE Wireless Commun. Mag.*, vol. 27, no. 5, pp. 126–132, Oct. 2020.

[8] R. Coulter, Q.-L. Han, L. Pan, J. Zhang, and Y. Xiang, "Data-driven cyber security in perspective—Intelligent traffic analysis," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3081–3093, Jul. 2020.

[9] D. Wu, Z. Yang, B. Yang, R. Wang, and P. Zhang, "From centralized management to edge collaboration: A privacy-preserving task assignment framework for mobile crowdsensing," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4579–4589, Mar. 2021.

[10] D. Wu, J. Yan, H. Wang, and R. Wang, "Multiattack intrusion detection algorithm for edge-assisted Internet of Things," in *Proc. IEEE Int. Conf. Ind. Internet (ICII)*, Orlando, FL, USA, Nov. 2019, pp. 1–5.

[11] J. Yan, D. Wu, and R. Wang, "Socially aware trust framework for multimedia delivery in D2D cooperative communication," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 625–635, Mar. 2019.

[12] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in IoT edge devices," *IEEE Internet Things J.*, early access, Jul. 28, 2021, doi: 10.1109/JIOT.2021.3100755.

[13] S. I. Popoola, B. Adebisi, M. Hammoudeh, G. Gui, and H. Gacanin, "Hybrid deep learning for botnet attack detection in the Internet-of-Things networks," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4944–4956, Mar. 2021.

[14] D. Wu, H. Shi, H. Wang, R. Wang, and H. Fang, "A feature-based learning system for Internet of Things applications," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1928–1937, Apr. 2019.

[15] Y. Lin, Y. Tu, Z. Dou, L. Chen, and S. Mao, "Contour stella image and deep learning for signal recognition in the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 34–46, Mar. 2021.

[16] Y. Wang *et al.*, "Automatic modulation classification for MIMO systems via deep learning and zero-forcing equalization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5688–5692, May 2020.

[17] H. Valpola, "From neural PCA to deep unsupervised learning," 2014, *arXiv:1411.7783*.

[18] Y. Lu *et al.*, "Semi-supervised machine learning aided anomaly detection method in cellular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8459–8467, Aug. 2020.

[19] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 2013, pp. 1–6.

[20] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Seattle, WA, USA, Dec. 2018, pp. 1367–1376.

[21] D. Wu, X. Han, Z. Yang, and R. Wang, "Exploiting transfer learning for emotion recognition under cloud-edge-client collaborations," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 479–490, Feb. 2021.

[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[23] X. Zhao, X. Han, W. Su, and Z. Yan, "Time series prediction method based on convolutional autoencoder and LSTM," in *Proc. Chin. Autom. Congr. (CAC)*, Hangzhou, China, Nov. 2019, pp. 5790–5793.

[24] X. Chen, X. Li, P. Chen, Y. Liu, S. Sun, and J. Liu, "Research on android application detection based on static permission and random forest," in *Proc. Int. Conf. Control Robot. Cybern. (CRC)*, Wuhan, China, Oct. 2020, pp. 181–184.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1097–1105.

[26] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[27] E. Biersack, C. Callegari, and M. Matijasevic, *Data Traffic Monitoring and Analysis*. Berlin, Germany: Springer, 2013.

[28] W. Li, M. Canini, A. W. Moore, and R. Bolla, "Efficient application identification and the temporal and spatial stability of classification schema," *Comput. Netw.*, vol. 53, no. 6, pp. 790–809, Apr. 2009.

[29] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of P2P traffic using application signatures," in *Proc. 13th Int. Conf. World Wide Web*, New York, NY, USA, May 2004, pp. 512–521.

[30] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *Proc. IEEE Int. Conf. Comput. Commun. (ICCC)*, Chengdu, China, Oct. 2016, pp. 2451–2455.

[31] J. Wang, G. Gui, T. Ohtsuki, B. Adebisi, H. Gacanin, and H. Sari, "Compressive sampled CSI feedback method based on deep learning for FDD massive MIMO systems," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5873–5885, Sep. 2021, doi: 10.1109/TCOMM.2021.3086525.

[32] Y. Wang, G. Gui, H. Gacanin, T. Ohtsuki, O. A. Dobre, and H. V. Poor, "An efficient specific emitter identification method based on complex-valued neural networks and network compression," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2305–2317, Aug. 2021.

[33] R. Vijayanand, D. Devaraj, and B. Kannapiran, "A novel deep learning based intrusion detection system for smart meter communication network," in *Proc. IEEE Int. Conf. Intell. Tech. Control Optim. Signal Process. (INCOS)*, Virudhunagar, India, Apr. 2019, pp. 1–3.

[34] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Ghent, Belgium, 2020, pp. 444–449.

[35] R. Zhao *et al.*, "An efficient intrusion detection method based on dynamic autoencoder," *IEEE Wireless Commun. Lett.*, vol. 10, no. 8, pp. 1707–1711, Aug. 2021.

[36] A. Rasmus, H. Berglund, M. Honkala, H. Valpola, and T. Raiko,"Semi-Supervised learning with ladder networks," 2015, *arXiv:1507.02672*.

[37] A. Tarvainen, and H. Valpola,“ Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results,” Apr. 2018, *arXiv:1703.01780*.
[38] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin,“NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for Industrial Internet of Things,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.
[39] K. S. Neethu and D. Varghese, “An incremental semi-supervised approach for visual domain adaptation,” in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Chennai, India, Apr. 2017, pp. 1343–1346.
[40] B. Gong, Y. Shi, F. Sha, and K. Grauman, “Geodesic flow kernel for unsupervised domain adaptation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Providence, RI, USA, Jun. 2012, pp. 2066–2073.
[41] H. Daume III, “Frustratingly easy domain adaptation,” Jul. 2009, *arXiv:0907.1815*.
[42] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, “Adapting visual cateogry models to new domains,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Crete, Greece, Sep. 2010, pp. 213–226.
[43] F. Perronnin, J. Sanchez, and Y. Liu, “Large-scale image categorization with explicit data embedding,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, San Francisco, CA, USA, Jun. 2010, pp. 2297–2304.
[44] H. Bay, T. Tuytelaars, and L. V. Gool, “Speeded-up robust features (SURF),” *Comput.Vis. Image Understand.*, vol. 110, no. 3, pp. 346–359, Jun. 2008.
[45] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010.
[46] M. Ghifary, W. B. Kleijn, and M. Zhang, “Domain adaptive neural networks for object recognition,” in *Proc. Pac. Rim Int. Conf. Artif. Intell. (PRICAI)*, Gold Coast, QLD, Australia, Dec. 2014, pp. 898–904.
[47] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: maximizing for domain invariance,” Dec. 2014, *arXiv:1412.3474*.
[48] M. Long, Y. Cao, J. Wang, and M. I. Jordan, “Learning transferable features with deep adaptation networks,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jul. 2015, pp. 97–105.

Jinhui Ning (Graduate Student Member, IEEE) received the B.E. degree in communication engineering from Shijiazhuang University, Shijiazhuang, China, in 2020. She is currently pursuing the master's degree in communication and information engineering with Nanjing University of Posts and Telecommunications, Nanjing, China.

Her research interest includes learning for secure Industrial Internet of Things.

Guan Gui (Senior Member, IEEE) received the Ph.D. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2012.

Since 2015, he has been a Professor with Nanjing University of Posts and Telecommunications, Nanjing, China. His recent research interests include artificial intelligence, deep learning, nonorthogonal multiple access, wireless power transfer, and physical-layer security.

Yu Wang (Graduate Student Member, IEEE) received the B.S. degree in communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2018, where he is currently pursuing the Ph.D. degree.

He has published more than 20 IEEE journal/conference papers. His research interests include deep learning, optimization, and its application in wireless communications.

Mr. Wang received several best paper awards, i.e., ICEICT 2019, CSPS 2019, and CSPS 2018.

Jie Yang (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in communication engineering from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2003, 2006, and 2018, respectively.

She is currently an Assistant Professor with Nanjing University of Posts and Telecommunications and also with the Shaanxi Key Laboratory of Information Communication Network and Security, Xi'an University of Posts and Telecommunications, Xi'an, Shaanxi, China. Her research interests include deep learning and its applications in intelligent wireless communication.

Bamidele Adebisi (Senior Member, IEEE) received the bachelor's degree in electrical engineering from Ahmadu Bello University, Zaria, Nigeria, in 1999, and the master's degree in advanced mobile communication engineering and the Ph.D. degree in communication systems from Lancaster University, Lancaster, U.K., in 2003 and 2009, respectively.

In 2012, he joined Manchester Metropolitan University, Manchester, U.K., where he is currently a Full Professor (Chair) of Intelligent Infrastructure Systems.

Song Ci (Senior Member, IEEE) received the Ph.D. degree from the Department of Electrical Engineering, University of Nebraska–Lincoln, Lincoln, NE, USA, in 2002.

His current research interests include large-scale dynamic complex system modeling and optimization as well as its applications in the areas of the Internet and energy Internet, especially in energy digitization and digital battery energy storage.

Haris Gacanin (Fellow, IEEE) received the M.Sc. and Ph.D. degrees from Tohoku University, Sendai, Japan, in 2005 and 2008, respectively.

He is the Head of the Chair for Distributed Signal Processing and the Co-Director of the Institute for Communication Technologies and Embedded Systems, RWTH Aachen University, Aachen, Germany. His professional interests are related to broad areas of digital signal processing and artificial intelligence with applications in wireless communications.

Fumiyuki Adachi (Life Fellow, IEEE) received the B.S. and Dr.Eng. degrees in electrical engineering from Tohoku University, Sendai, Japan, in 1973 and 1984, respectively.

In January 2000, he joined Tohoku University, where he was a Professor with the Department of Communications Engineering, Graduate School of Engineering until he retired in March 2016. He is currently a specially appointed Professor with the Research Organization of Electrical Communication, Tohoku University, and is continuing his research interest in the area of wireless signal processing and wireless networking.