# TransNet: Unseen Malware Variants Detection Using Deep Transfer Learning

Candong Rong[1,2], Gaopeng Gou[1,2], Mingxin Cui[1,2], Gang Xiong[1,2], Zhen Li[1,2], and Li Guo[1,2(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{rongcandong,gougaopeng,cuimingxin,xionggang,lizhen,guoli}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

**Abstract.** The ever-increasing amount and variety of malware on the Internet have presented significant challenges to the interconnected network community. The emergence of unseen malware variants has resulted in a different distribution of features and labels in the training and testing datasets. For widely used machine learning-based detection methods, the issue of dataset shift will render the trained model ineffective in the face of new data. However, it is a laborious and tedious undertaking whether relearning features to describe new data or collecting large amounts of labeled samples to retrain the classifiers. To address these problems, this paper proposes TransNet, a framework based on deep transfer learning for unseen malware variants detection. We first convert the raw traffic represented by sessions containing data from all layers of the OSI model into fixed-size RGB images through data preprocessing. Afterward, based on the ResNet-50 model pre-trained on the ImageNet, we replace Batch Normalization with Transferable Normalization as the normalization layer to construct our deep transfer learning model. In this way, our approach leverages deep learning to avoid the problem of traditional machine learning in relying on expert knowledge and uses transfer learning to address the issue of domain shift. We test the effectiveness of different methods with a thorough set of experiments. TransNet achieves 95.89% accuracy and 96.09% F-measure on two public datasets from the real-world environment, which is higher than comparative methods. Meantime, our method ranks first on all ten subtasks, showing that it can detect unseen malware variants with stable and excellent performance. Moreover, the distribution discrepancy computed by our method is much smaller than other approaches, which illustrates that our method successfully reduces the shift of data distributions.

**Keywords:** Deep transfer learning · Unseen malware variants detection · Network traffic classification.


N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 84–101, 2020.
https://doi.org/10.1007/978-3-030-63095-9_5

## 1 Introduction

Malicious software (malware), which is the tool for launching cyber attacks, poses great threats to military, government, and industrial production. Network traffic analysis plays an important role in addressing the issue of detecting malware [1,2] and is widely used in Intrusion Detection System (IDS) and advanced firewalls [3,4]. Over the years, including Advanced Persistent Threat (APT) activities and attacks using zero-day vulnerabilities, more and more cyber attacks have evaded detection by making use of the variation and obfuscation of malware [5]. Those variants have taken a toll on the enterprises, e.g., GandCrab that is a variant of ransomware has made 2 billion dollars in one and a half years [6]. Conventional signature-based detection methods are limited to known samples and patterns in the database, so they cannot handle the increasingly sophisticated and diverse malware variants [7]. Therefore, building a novel detection system in a rapidly changing network is essential to weaken the critical challenges that malware variants pose to detection systems.

In recent years, machine learning techniques have been applied to detect malware for improving the detection rate. On the one hand, data-driven supervised learning models rely on large quantities of labeled data, and their results depend on known samples in the training set [8,9]. Once the malicious samples change behavior, the detection rate of the trained classifiers will drop dramatically, making it impossible to detect malware variants effectively. As a result, it needs recollecting new data to retrain the detection model for accommodating these changes. However, labeled data are often costly to obtain in the real world, especially malicious network traffic, as it represents a tiny small percentage of traffic. On the other hand, unsupervised learning detection methods divide the samples into distinct clusters based on the similarity among them. They can detect new threats, but lots of false alerts limit their practical usefulness [10,11]. Similar to signature-based methods, traditional machine learning approaches fail to detect unseen threats. Consequently, it is necessary to find out an effective method for understanding and identifying malware variants.

The problem of changing malicious behavior between known malware samples and unseen variants can be solved by *domain adaptation (transfer learning)*. According to the concept of transfer learning, samples in the source domain are considered as already known and labeled, while samples in the target domain are unlabeled and different from those in the source domain. By transferring the knowledge learned from the source domain to the target domain, the task of detecting unseen malware variants is completed. Recently, several papers have focused on utilizing transfer learning to solve the issue of detecting malware variants. For example, Bartos et al. learned domain-invariant statistical feature representation computed from the network traffic to detect unseen malware variants [12]. Li et al. studied a novel method to identify malware variants based on adaptive regularization transfer learning [13]. Zhao et al. proposed a feature-based heterogeneous transfer learning approach HeTL [14] and a hierarchical transfer learning algorithm with clustering enhancement CeHTL [15] to detect unseen variants of attacks. However, the above methods are the combination

of traditional machine learning technology and transfer learning. Conventional machine learning techniques are limited in their ability to process natural data in their raw form, as these methods rely on considerable domain expertise to build a system that could detect patterns in the input [16]. Not only are these methods time-consuming and labor-intensive, but they are also not as effective. Meanwhile, with the deployment of TLS 1.3 and ESNI (Encrypted Server Name Indication), some features in the traditional machine learning methods are no longer applicable to the fully encrypted traffic analysis. On the contrary, instead of extracting manually designed features from traffic data, with the help of multiple processing layers, deep learning methods can learn the very complex function to automatically convert the raw input into representations at a higher and more abstract level. Hence, methods of combining deep learning with transfer learning are considered to achieve more satisfactory results.

In the field of network traffic classification, one type of commonly used transfer learning-based approach is to use the models pre-trained on the ImageNet [17] as feature extractors [18,19]. It freezes the convolutional layers of deep neural networks and adapts the last layer to the particular task. In this way, the structure and parameters of pre-trained models are transferred to new tasks, reducing the computational cost of training from scratch while taking advantage of the knowledge that pre-trained models bring. However, such methods do not further consider the variability of data distribution between datasets, so they still cannot effectively address the issue of detecting unseen malware variants.

In this paper, we propose TransNet, a framework based on deep transfer learning to detect unseen malware variants. In the procedure of data preprocessing, we first divide malicious traffic and benign traffic into sessions containing data from all layers of the OSI model, and then do traffic anonymization and remove duplicated files. Afterward, we convert the first 900 bytes of data from each session into fixed-size RGB images. Based on the ResNet-50 [20] model pre-trained on the ImageNet, we use Transferable Normalization [21] to replace Batch Normalization [22] as the normalization layer to form our deep transfer learning model. Finally, these RGB images are used as input of the model to generate the results. The experiments show that our framework TransNet can detect unseen malware variants with superior performance and smaller distribution discrepancy than comparative methods.

In conclusion, the main contributions of our work are briefly summarized as follows:

1. We present a framework based on deep transfer learning for detecting unseen malware variants. It combines deep learning with transfer learning to solve the problem of different data distribution between datasets. It leverages the powerful representation ability of deep neural networks and uses knowledge learned from other domains to make decisions about tasks in new domains. In this way, our method avoids the problem of relying on expert knowledge for feature extraction in traditional machine learning methods and dramatically reduces the cost of manually labeling large amounts of data, enabling it to respond to emerging malware in a relatively short time.

2. Our method is based on the ResNet-50 model pre-trained on the ImageNet, which not only utilizes well-established architecture and parameters but also avoids the computational cost of training from scratch. The decision that uses Transferable Normalization to replace Batch Normalization as the normalization layer has improved the transferability of deep neural networks without increasing additional overhead.
3. Our method achieves 95.89% accuracy and 96.09% F-measure on public datasets from the real-world environment, which is higher than comparison methods. Even though other methods yield good results on the particular subtasks, our method ranks first on all ten subtasks with stable and excellent performance, which shows the superiority of our method in detecting unseen malware variants. Furthermore, MMD (Maximum Mean Discrepancy) of our method is much smaller than other methods on four typical subtasks, which suggests that our method successfully reduces the shift of data distributions.

The rest of this paper is organized as follows. We summarize related work in Sect. 2 and review background knowledge in Sect. 3. In Sect. 4, we describe our deep transfer learning framework. We discuss the experiments in detail and analyze the results in Sect. 5. Finally, we conclude this paper in Sect. 6.

## 2 Related Work

The issue of detecting unseen malware variants has become a popular topic in recent years. A great effort has been devoted to applying different methods to detect the growing number of variants in the constantly changing network environment. In this section, we retrospect the most relevant work to our method in two parts.

### 2.1 Conventional Methods for Malware Detection

Traditionally, signature-based methods [7,23] find the threats through the predefined signatures extracted from known samples in the database, which means they cannot be effective against novel attacks. Recently, machine learning technology has played a major role in the task of malware detection. They are mainly divided into two parts, namely supervised models [8,9] and unsupervised methods [10,11]. The models based on supervised learning train data-driven classifiers to distinguish malware from the benign samples. New data that are different from the training set will cause the accuracy of the already trained models to decrease. Unsupervised methods divide the samples into different clusters according to the similarity of behavior. They can detect unseen threats, but the high false alarm rate is unbearable in practice. Conventional machine learning-based methods have the same drawback as signature-based methods in that they work poorly in the face of new threats. The increasing number of malware variants has grown up to be a bottleneck in the development of detection systems.

### 2.2 Transfer Learning for Unseen Malware Detection

Transfer Learning, a novel machine learning technology, aims to build effective learners that can leverage the knowledge of rich labeled data from the source domain and transfer it to the target domain short of labeled data for reducing the shift in data distributions across different domains [24]. Even if transfer learning approaches are widely used in the tasks of computer vision and natural language processing, few studies have utilized them to address the issue of unseen malware detection. In recent papers, Bartos et al. [12] proposed a domain-invariant representation to detect previously unseen malware and behavior changes. The optimization method represented network traffic through the combination of invariant histograms of feature values and feature differences. Li et al. [13] thought it was not enough for knowledge transfer if only considered minimizing the difference between marginal distribution of different domains. Compared to [12], they studied a novel approach to detect unseen variants based on adaptive regularization transfer learning, which could reduce the marginal distribution and conditional distribution between the source and target domains. Zhao et al. proposed HeTL [14] and CeHTL [15] to solve the problem of unknown network attacks detection, respectively. The former method could find optimized representations for both training and testing data by transforming them into a common latent space via the spectral transformation where the difference of distributions could be reduced. The method CeHTL was proposed to make up for the shortcoming that the performance of HeTL depended on manual presettings of hyper-parameters. It was a hierarchical transfer learning algorithm with clustering enhancement, which could cluster the source and target domains and compute the relevance between them.

## 3 Background

In this section, to better understand the research in this paper, we review the relevant background knowledge. Firstly, we list the definitions related to transfer learning. We then introduce the processing steps of Batch Normalization that has a strong ability to accelerate network training. Finally, we compare Transferable Normalization and Batch Normalization, showing the changes made by the former to improve the transferability of the normalization layer. Table 1 summarizes the notations used in this paper and their corresponding descriptions.

### 3.1 Definition

In this part, we introduce the definitions related to transfer learning, including the "domain", the "task", and domain adaptation.

**Definition 1 (Domain).** *A domain $\mathcal{D}$ consists of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathbf{x})$, i.e., $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$, where $\mathbf{x} \in \mathcal{X}$ [25].*

**Table 1.** Notations used in our paper and their corresponding descriptions

| Notation | Description | Notation | Description |
|---|---|---|---|
| $\mathcal{D}_s$ | the source domain | $\mathcal{D}_t$ | the target domain |
| $\mathcal{X}$ | feature space | $P$ | marginal probability distribution |
| $\mathcal{T}_s$ | the source task | $\mathcal{T}_t$ | the target task |
| $\mathcal{Y}$ | label space | $f$ | predictive function |
| $Q$ | conditional probability distribution | $\mathcal{B}$ | mini-batch |
| $k$ | channel | $n$ | size of mini-batch |
| $\mu$ | mean | $\sigma^2$ | variance |
| $\widehat{x}$ | normalized value | $\gamma$ | the scaling parameter |
| $\beta$ | the shift parameter | $d$ | the domain distance |
| $l$ | size of channel | $\alpha$ | distance-based probability |

In general, if the source domain $\mathcal{D}_s$ and the target domain $\mathcal{D}_t$ are different, then they may have different feature spaces or different marginal probability distributions, i.e., $\mathcal{X}_s \neq \mathcal{X}_t \vee P_s(\mathbf{x}_s) \neq P_t(\mathbf{x}_t)$ [26].

**Definition 2 (Task).** *Given a specific domain, $\mathcal{D} = \{\mathcal{X}, P(\mathbf{x})\}$, a task $\mathcal{T}$ is composed of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\mathbf{x})$, i.e., $\mathcal{T} = \{\mathcal{Y}, f(\mathbf{x})\}$. From the probabilistic viewpoint, $f(\mathbf{x}) = Q(y|\mathbf{x})$ can be considered as the conditional probability distribution, where $y \in \mathcal{Y}$, and $y$ is the corresponding label of an instance $\mathbf{x}$ [25].*

In general, if the source task $\mathcal{T}_s$ and the target task $\mathcal{T}_t$ are different, then they may have different label spaces or different conditional probability distributions, i.e., $\mathcal{Y}_s \neq \mathcal{Y}_t \vee Q_s(y_s|\mathbf{x}_s) \neq Q_t(y_t|\mathbf{x}_t)$ [26].

**Definition 3 (Domain Adaptation).** *Given a labeled source domain $\mathcal{D}_s = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and an unlabeled target domain $\mathcal{D}_t = \{\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{n+m}\}$, where $\mathbf{x}_i$ is an instance and $y_i$ is the corresponding label, the goal of domain adaptation is to utilize the knowledge learned from the source domain $\mathcal{D}_s$ and the source task $\mathcal{T}_s$ for generating a target predictive function $f_t : \mathbf{x}_t \mapsto y_t$ on the target domain $\mathcal{D}_t$ with low error rate, under the assumptions of $\mathcal{X}_s = \mathcal{X}_t$, $\mathcal{Y}_s = \mathcal{Y}_t$, $P_s(\mathbf{x}_s) \neq P_t(\mathbf{x}_t)$, and $Q_s(y_s|\mathbf{x}_s) \neq Q_t(y_t|\mathbf{x}_t)$ [13].*

### 3.2 Batch Normalization (BN)

Based on the known practice that the network training converges faster if its inputs are whitened [27], Batch Normalization [22] was designed to accelerate the training of deep neural networks. It transforms the inputs of each layer to have zero means and unit variances, and then scales and shifts them with a pair of learnable parameters to restore the representation power of the network. To reduce the expensive computation cost of fully whitening the inputs of each layer, it makes two necessary simplifications: normalize each scalar feature independently in each mini-batch. Specifically, in a mini-batch $\mathcal{B}$ of size $n$, for a layer

with $l$-dimensional input $\mathbf{x} = \left(x^{(1)} \ldots x^{(l)}\right)$, it focuses on a particular dimension (channel) $x^{(k)}$, which has $n$ values in this channel $k$. The first is that it calculates two statistics: the expectation $\mu^{(k)}$ and the variance $\sigma^{2(k)}$ for the channel $k$ using the training data in a mini-batch $\mathcal{B}$:

$$\mu^{(k)} = \frac{1}{n} \sum_{i=1}^{n} x_i^{(k)}, \tag{1}$$

$$\sigma^{2(k)} = \frac{1}{n} \sum_{i=1}^{n} \left(x_i^{(k)} - \mu^{(k)}\right)^2. \tag{2}$$

It then utilizes the statistics to normalize the inputs of every layer to have the mean of 0 and the variance of 1. The formula is shown as follows:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mu^{(k)}}{\sqrt{\sigma^{2(k)} + \epsilon}}, \tag{3}$$

where $\epsilon$ is a constant added to the mini-batch variance for numerical stability. The second step is to learn two trainable parameters for scaling and shifting the normalized value of the channel $k$ to represent the identity transform and preserve the network capacity:

$$y^{(k)} = \gamma \widehat{x}^{(k)} + \beta \equiv \mathrm{BN}_{\gamma,\beta}\left(x^{(k)}\right), \tag{4}$$

where $\mathrm{BN}_{\gamma,\beta}\left(x^{(k)}\right) : x^{(k)} \rightarrow y^{(k)}$ is the Batch Normalization Transform.

There is no doubt that Batch Normalization has made outstanding contributions in accelerating network training. However, without considering the difference between the training data and testing data, BN combines them into a single batch to feed as the input of the normalization layer. The representations of data that are following different distributions share the expectation and the variance, which is unsuitable due to the phenomenon known as dataset shift [28]. Additionally, all channels are given the same importance in BN, which is unreasonable in the case of domain adaptation since some channels are more transferable than the others. Therefore, the transferability of BN needs to be improved.

### 3.3 Transferable Normalization (TransNorm)

In order to overcome the aforesaid shortcomings of Batch Normalization when applied to domain adaptation, Transferable Normalization [21] (TransNorm) was proposed to improve the transferability of deep neural networks. To bridge different domains in domain adaptation, comparing with BN, TransNorm changes the shared mean and variance to domain-specific mean and variance, and assigns different weights to the channels according to the difference of transferability. In this part, we introduce the processing procedure of TransNorm in three steps: domain-specific mean and variance, domain-shared learnable parameters, and domain adaptive weights.

In each BN layer, the inputs coming from the source and target domains are combined into a single batch to calculate the mean and variance. However, due to dataset shift, there exists a significant difference in data distribution across two domains, leading to having different basic statistics between the source and target domains. Instead of straightforwardly sharing the mean and the variance, TransNorm separately calculates domain-specific statistics of each channel: the source statistics $\mu_s$, $\sigma_s^2$ and the target statistics $\mu_t$, $\sigma_t^2$ as follows (let us focus on the particular channel $k$ and omit $k$ for clarity):

$$\mu_s = \frac{1}{n_s}\sum_{i=1}^{n_s} x_i, \quad \sigma_s^2 = \frac{1}{n_s}\sum_{i=1}^{n_s}\left(x_i - \mu_s\right)^2, \tag{5}$$

$$\mu_t = \frac{1}{n_t}\sum_{j=1}^{n_t} x_j, \quad \sigma_t^2 = \frac{1}{n_t}\sum_{j=1}^{n_t}\left(x_j - \mu_t\right)^2, \tag{6}$$

where $n_s$ and $n_t$ correspond to the mini-batch size in the source and target domains, respectively. After that, the means and the variances are used to normalize the inputs of each normalization layer independently for each domain:

$$\widehat{x}_s = \frac{x_s - \mu_s}{\sqrt{\sigma_s^2 + \epsilon}}, \quad \widehat{x}_t = \frac{x_t - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}}, \tag{7}$$

where $\epsilon$ is a constant added to the mini-batch variance for numerical stability. In this way, the representations from the source and target domains are separately normalized to have the mean of 0 and the variance of 1.

With consideration of simply normalizing each channel of a layer may change what the layer can represent, BN learns a pair of parameters $\gamma$ and $\beta$ to scale and shift the normalized values for ensuring the transformation inserted in the network can preserve the representation ability of the network. TransNorm retains the capability of this part of BN, and scales and shifts the normalized values of each normalization layer with domain-shared parameters for different domains:

$$w_s = \gamma\widehat{x}_s + \beta, \quad w_t = \gamma\widehat{x}_t + \beta. \tag{8}$$

Both domain-specific basic statistics and domain-shared learnable parameters treat all channels equally. However, in the context of different channels extract different aspects from the input, the information existing in some aspects is more transferable than the others, resulting in differences in the transferability of different channels. Therefore, in the field of domain adaptation, we need to consider the difference in channel transferability further. To accomplish this, TransNorm reuses the basic statistics from the source and target domains: $\mu_s$, $\sigma_s^2$ and $\mu_t$, $\sigma_t^2$ to select the channels that are more transferable. For each channel $k$, TransNorm calculates the distance across domains $d^{(k)}$ as follows:

$$d^{(k)} = \left| \frac{\mu_s^{(k)}}{\sqrt{\sigma_s^2(k) + \epsilon}} - \frac{\mu_t^{(k)}}{\sqrt{\sigma_t^2(k) + \epsilon}} \right|, \quad k = 1, 2, \ldots, l, \tag{9}$$

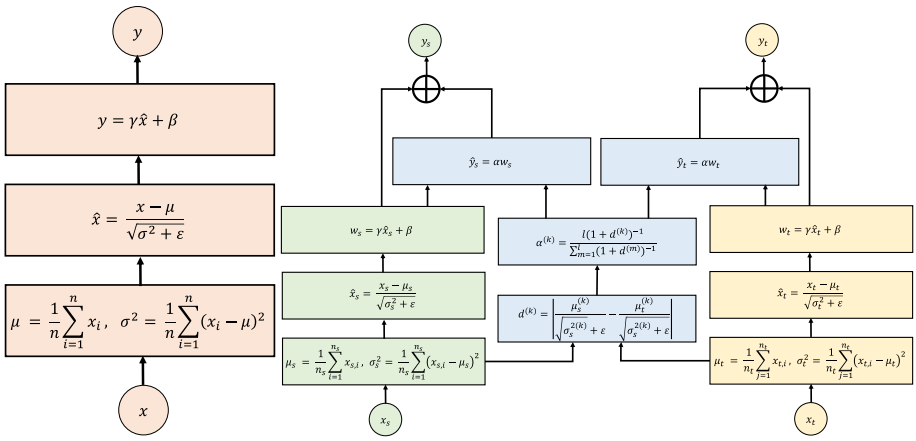
where $l$ is the size of channels within each TransNorm layer. For enhancing the understandability and applicability of the method, TransNorm utilizes Student t-distribution [29] to convert the channel-wise distances to probabilities. It assigns different weights to the channels according to the transferability calculated by the distance-based probability $\alpha^{(k)}$:

$$\alpha^{(k)} = \frac{l\left(1+d^{(k)}\right)^{-1}}{\sum_{m=1}^{l}\left(1+d^{(m)}\right)^{-1}}, \quad k = 1, 2, \ldots, l, \tag{10}$$

where $l$ is the number of channels. In doing so, the channels that are more transferable are assigned with higher weights, improving the transferability of the normalization layer for domain adaptation. Finally, to avoid overly penalizing the channels that are less transferable, TransNorm introduces the residual mechanism to combine the normalized values of Eq. (8) with the transferability-weighted one to generate the output of each normalization layer:

$$y_s = (1+\alpha)w_s, \quad y_t = (1+\alpha)w_t. \tag{11}$$

We visually show the differences between Batch Normalization and Transferable Normalization in Fig. 1. Based on BN, the changes made by TransNorm to improve the transferability are reflected in the following two aspects: the first is that TransNorm normalizes the representations from the source and target domains separately, instead of mixing the inputs across domains into a single batch for normalizing like BN. The second is that TransNorm reuses the basic statistics from different domains to calculate the transferability for each channel to select those more transferable channels, making sure the similar patterns are shareable across domains.




(a) Batch Normalization (b) Transferable Normalization

**Fig. 1.** The processing procedure of Batch Normalization is shown in (a). The processing procedure of Transferable Normalization is presented in (b).

## 4 Design of Framework

In this section, we discuss the design of our framework based on deep transfer learning for unseen malware variants detection. The framework is composed of two crucial components: data preprocessing and deep transfer learning model. As the input of the data preprocessing module, the raw traffic of malware variants and benign applications is converted to images in RGB format. We use Transferable Normalization to replace Batch Normalization in the ResNet-50 model pre-trained on the ImageNet for forming a classification model based on deep transfer learning. The model learns the representations of different types of traffic from the RGB images generated in the previous step, and then produces the final classification results. The architecture of our framework is shown in Fig. 2.

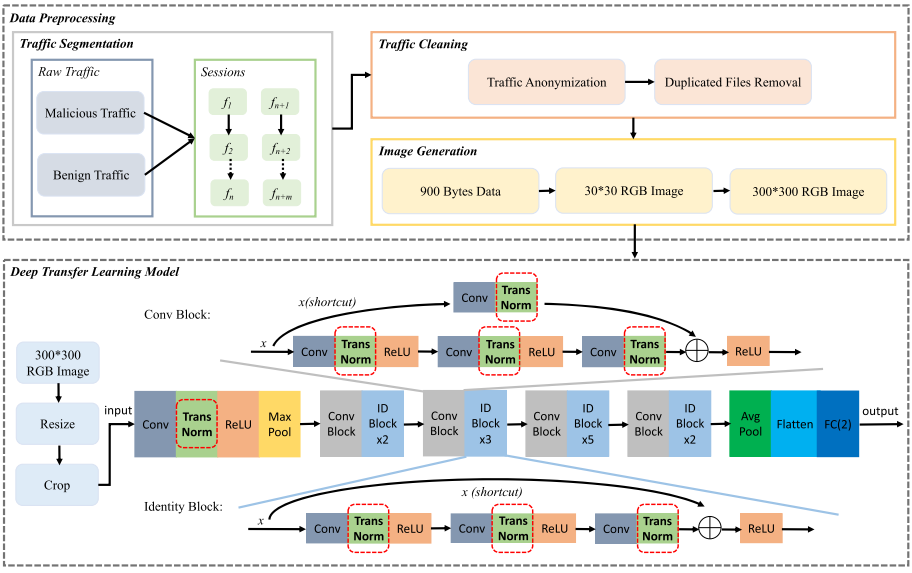



**Fig. 2.** The architecture of our framework TransNet.

### 4.1 Data Preprocessing

In the initial stage of data preprocessing, we first need to determine the way of representation for traffic. The flow and the session are commonly used units for traffic splitting in the field of network traffic classification. A flow is a set of packets with the same 5-tuple. The 5-tuple is composed of source IP, source port, destination IP, destination port, and transport layer protocol. Moreover, a session consists of bidirectional flows between two communication hosts. Generally speaking, bidirectional flows contain more interactive information than unidirectional flows. Meanwhile, the representations generated by the data from

all layers of the OSI model have more comprehensive traffic features than those from the application layer. Thus, we select the session as the unit for traffic splitting and reserve the data coming from all layers of the OSI model.

As shown in Fig. 2, to convert raw traffic into RGB images with a size of 300 * 300, we need to go through three steps: traffic segmentation, traffic cleaning, and image generation. In the first step, we split raw traffic to large numbers of sessions containing data from all layers of the OSI model. The input and output data formats of this step are packet capture files (pcap). In the stage of traffic cleaning, we first conduct traffic anonymization, that is, randomize MAC addresses in the data link layer and IP addresses in the IP layer. We then remove duplicate files generated by packets with the same content for avoiding biases in the training of deep learning models. In the final step, we first extract the first 900 bytes of data from each session and convert them into RGB images of size 30 * 30. Each byte of original data represents a pixel. If the size of a session is larger than 900 bytes, it is adjusted to 900 bytes. If the size of a session is smaller than 900 bytes, the 0×00 is added in the end to complement it to 900 bytes. There are two ways to expand on why the first 900 bytes of a session are used. The first is to meet the input size of the deep learning model and to make the method more lightweight for processing the raw data. The second is that during the initial phase of the session, the communicating parties exchange important information, such as communication parameters, which contains important features that we need to focus on, while the application data transmitted afterward is less important by comparison. A similar operation occurs in the papers [18,30] that are related to malware classification. This choice applies to both the traffic of malware variants and the traffic of benign applications. To meet the input data size of deep transfer learning model to be used next, we copy 100 copies of each image of size 30 * 30 and arrange them separately into images of size 300 * 300.

### 4.2 Deep Transfer Learning Model

Deep neural networks allow computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction, and these methods have dramatically improved the state-of-the-art in a wide range of tasks, such as image classification, object detection, and other challenging fields [16]. In recent years, many researchers have proposed numerous deep learning models and made unremitting efforts to improve the performance of the methods. Taking the classic image classification task, ILSVRC (ImageNet Large Scale Visual Recognition Challenge), as an example, different deep neural networks were trained on the ImageNet [17] to reduce the classification error rate, e.g., AlexNet [31], VGG [32], and ResNet [20]. Impressed by the powerful representation capability of deep neural networks, the researchers focusing on network traffic classification utilize them to promote the improvement of performance. In general, the more complex structure the model has, the better performance the model can achieve on a dataset. Meantime, the models pretrained on large datasets have configured structure and parameters, avoiding

the cost caused by training from scratch. In our method, after making a trade off between the performance of models and the computational cost of training, we choose the ResNet-50 model pre-trained on the ImageNet as our basic deep learning model.

The purpose of combining deep learning models with transfer learning is to leverage the deep neural networks for cross-domain representation learning. We delve into the architecture design of deep neural networks and use Transferable Normalization (TransNorm) successfully applied in computer vision for reference. TransNorm, which is a more transferable normalization technique, fully considers the difference in data distributions between the source and target domains. Meanwhile, it will not bring additional computational cost and change other network modules. As marked with red boxes in Fig. 2, based on the ResNet-50 model pre-trained on the ImageNet, we propose our deep transfer learning model by leveraging Transferable Normalization as the normalization layer to replace existing Batch Normalization. Our framework can bridge the source and target domains, which will also be confirmed in the experiments. As shown in Fig. 2, the RGB images after being resized and cropped are used as the input of our deep transfer learning model to produce the final results.

## 5 Evaluation

In this section, we evaluate our method on the real-world datasets. Firstly, we introduce the components of two public datasets used in our experiments and describe the experimental setup. We then construct a series of comparison experiments with different approaches and discuss the corresponding results. The results support the advantage of our deep transfer learning-based method in detecting unseen malware variants with superior performance over comparative methods.

### 5.1 Dataset and Experimental Setup

In our experiments, we are based on pcap files to carry on the study of malware variants detection from the perspective of network traffic. We use two public datasets for evaluating different methods, where the malicious traffic comes from Malware Capture Facility Project (MCFP) [33], and the benign traffic is composed of data from USTC-TFC2016 [30]. The MCFP is a research project of CTU (Czech Technical University) aiming to collect different kinds of malicious traffic from the real-world network environment. We choose malicious traffic generated by ten types of malware variants from it, including traffic of common categories of malware such as trojan, botnet, and virus. The benign traffic in the USTC-TFC2016 dataset consists of the traffic generated by different classes of normal applications, e.g., data transfer tools, instant messaging applications, email communication systems, databases, and games. As mentioned in Sect. 4.1, we utilize sessions to describe the raw traffic in the aforementioned public datasets. After the data preprocessing, we convert the first 900 bytes of each session into RGB

images with a size of 300*300. The RGB images are the input of the deep transfer learning model, and their number is equal to the number of sessions. Table 2 illustrates a general view of datasets used in our experiments.

**Table 2.** A general view of datasets used in our experiments

| Malicious Traffic | Category | Sessions | Benign Traffic | Category | Sessions |
|---|---|---|---|---|---|
| MCFP | Cridex | 7373 | USTC-TFC2016 | BitTorrent | 7517 |
| | Geodo | 9213 | | Facetime | 6000 |
| | Htbot | 5730 | | FTP | 15000 |
| | Miuref | 6066 | | Gmail | 8629 |
| | Neris | 7603 | | MySQL | 15000 |
| | Nsis-ay | 5462 | | Outlook | 7524 |
| | Shifu | 8671 | | Skype | 6321 |
| | Tinba | 7654 | | SMB | 9733 |
| | Virut | 7448 | | Weibo | 9988 |
| | Zeus | 9873 | | WorldOfWarcraft | 7883 |

In terms of the experimental setup, the training and testing data contain malicious traffic generated by different malware variants, in other words, for each subtask, we select nine malware variants as a part of training data and another malware variant to be part of testing data. For example, we choose malware variants other than Cridex as part of training data, and Cridex as part of testing data. The purpose of this experimental design is to simulate the scenario of detecting unseen threats created by malware developers for evading current detection methods and systems. Meanwhile, the benign traffic in the training and testing data is also different.

### 5.2 Basic Result

According to the experimental setup, the evaluation process of each method is divided into ten subtasks. The purpose of every subtask is to utilize nine malware variants to discover another novel variant. As mentioned in the introduction, some features in the methods of combining traditional machine learning technology and transfer learning are no longer suitable for the analysis of fully encrypted traffic. Therefore, in the comparison experiments, to ensure the comparability of methods, we select three pre-trained models commonly used in the studies related to network traffic classification, such as AlexNet, VGG19_bn, and ResNet-50, as the comparative methods. All methods are implemented based on PyTorch. The methods using pre-trained models are model-based transfer learning methods. Table 3 and Table 4 show the accuracy and F-measure of different approaches in the task of detecting unseen malware variants, respectively.

First of all, our method TransNet has achieved the best results among the approaches, with the accuracy of 95.89% and the F-measure of 96.09%. Afterward, the comparison of the average value of indicators of three pre-trained

**Table 3.** Accuracy (%) on our dataset for unseen malware variants detection

| Method | Subtask | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cridex | Geodo | Htbot | Miuref | Neris | Nsis-ay | Shifu | Tinba | Virut | Zeus | Avg |
| AlexNet | 71.30 | 82.10 | 71.22 | 80.23 | 75.54 | 63.47 | 74.40 | 74.26 | 71.11 | 66.95 | 73.06 |
| VGG19_bn | 87.75 | 85.64 | 88.72 | 77.02 | 79.21 | 71.38 | 83.66 | 91.56 | 87.39 | 93.36 | 84.57 |
| ResNet-50 | 90.49 | 90.38 | 90.70 | 91.62 | 83.51 | 65.39 | 92.32 | 92.16 | 79.84 | 85.94 | 86.24 |
| **TransNet** | **96.35** | **90.90** | **95.50** | **95.63** | **97.58** | **89.86** | **99.68** | **96.11** | **99.99** | **97.29** | **95.89** |

**Table 4.** F-measure (%) on our dataset for unseen malware variants detection

| Method | Subtask | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cridex | Geodo | Htbot | Miuref | Neris | Nsis-ay | Shifu | Tinba | Virut | Zeus | Avg |
| AlexNet | 68.63 | 78.40 | 71.97 | 80.71 | 74.51 | 63.50 | 67.91 | 69.82 | 68.95 | 60.64 | 70.50 |
| VGG19_bn | 87.84 | 85.73 | 89.08 | 72.10 | 78.06 | 64.09 | 83.73 | 91.68 | 87.41 | 92.85 | 83.26 |
| ResNet-50 | 90.64 | 90.27 | 91.29 | 92.19 | 83.35 | 57.48 | 92.13 | 92.26 | 76.88 | 85.98 | 85.25 |
| **TransNet** | **96.49** | **90.35** | **96.10** | **96.13** | **97.66** | **90.96** | **99.68** | **96.19** | **99.99** | **97.34** | **96.09** |

models, AlexNet, VGG19_bn, and ResNet-50, supports the experienced judgment that the increase in model complexity brings the improvement of performance. Although some comparison methods have achieved good results on particular subtasks, e.g., VGG19_bn has the accuracy of 93.36% when used to detect Zeus, but our method ranks first on all subtasks with stable and excellent performance. Our method is achieved by replacing Batch Normalization in the ResNet-50 pre-trained model with Transferable Normalization. The results of these two methods shown in the tables illustrate that TransNorm improves the transferability of deep neural networks. Specifically, TransNorm calculates the basic statistical information of different domains separately and assigns different weights to channels according to the transferability of them. The results show that our method is capable of detecting unseen malware variants at a superior performance. It can relieve the challenge caused by continually increasing malware in the real-world environment through reducing time cost and labor cost of manually labeling data.

### 5.3 Further Analysis

MMD (Maximum Mean Discrepancy) [34] is proposed as a criterion for comparing different distributions based on the Reproducing Kernel Hilbert Space (RKHS). It is a measure of the cross-domain discrepancy of distributions, which is commonly used in domain adaptation. According to the results in Table 3, among the ten subtasks, our method has the most significant performance gap with the second-ranked method on Neris and Nsis-ay, while our method has the smallest performance gap with the second-ranked method on Geodo and Tinba. Therefore, we compute MMD on the above four subtasks with features of AlexNet,VGG19_bn, ResNet-50, and TransNet. Figure 3 shows the results of MMD calculated by different methods on each subtask.

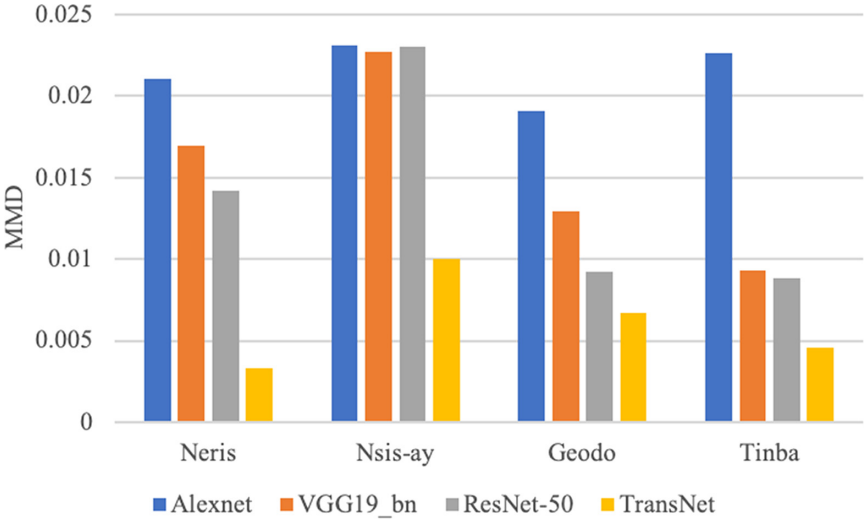



**Fig. 3.** The results of MMD calculated by different methods on four subtasks.

As shown in Fig. 3, we observe that MMD using TransNet features (our method) is much smaller than MMD using features of other methods, which suggests that TransNet features can close the cross-domain gap more effectively. Meanwhile, the MMD values of TransNet features in Neris and Tinba are smaller than those in Nsis-ay and Geodo, which explains better accuracy of TransNet on the subtasks of detecting Neris and Tinba. Furthermore, compared to the difference of MMD computed by TransNet and other methods on Geodo and Tinba, the MMD values of TransNet on Neris and Nsis-ay differ more significantly from those of other methods, which exactly corresponds to the degree of difference in performance between our method and the second-ranked method in the above four subtasks.

## 6 Conclusion

In this paper, we present a framework using deep transfer learning, namely TransNet, for unseen malware variants detection. We use data preprocessing to convert the raw traffic into RGB images as the input of our deep transfer learning model. Faced with the problem of identifying unseen malware variants, we delve into the architecture of deep neural networks and find that the operation of the normalization layer needs to take into account the variability of data distribution between different datasets. Consequently, to construct our deep transfer learning model, we replace the Batch Normalization in the ResNet-50 model pre-trained on the ImageNet with Transferable Normalization to improve the transferability of deep neural networks. We test different methods by simulating scenarios where the testing set contains different malware variants from the training set.

Our method achieves 95.89% accuracy and 96.09% F-measure on the public datasets from the real-world environment, which is higher than comparative methods. Meanwhile, our method ranks first on all ten subtasks with stable and excellent performance. The above two points show that our method can detect unseen malware variants. Furthermore, MMD of TransNet is much smaller than MMD of other approaches, which validates that our method successfully reduces the shift of distributions through more transferable representations. When new traffic emerges, instead of re-collecting large amounts of labeled data or learning novel representations, our framework achieves labeling and detecting malicious traffic generated by unseen malware variants with the help of the original trained model and analysis of the distribution of different datasets. In future work, we will conduct a more extensive analysis to discover more malware variants, such as malicious Windows PE (Portable Executable) files and malicious Android APK (Android application package) files. Meanwhile, we will explore the issue of unbalanced data between malicious traffic and benign traffic to improve the possibility of applying our method to the real-world environment.

**Acknowledgments.** This work is supported by The National Natural Science Foundation of China (No.61702501) and The National Key Research and Development Program of China (No.2016QY05X1000, No.2018YFB1800200).

## References

1. Anderson, B., McGrew, D.: Identifying encrypted malware traffic with contextual flow data. Proc. ACM Workshop Artif. Intell. Secur. **2016**, 35–46 (2016)
2. Wang, T.S., Lin, H.T., Cheng, W.T., et al.: DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis. Comput. Secur. **64**, 1–15 (2017)
3. Kovanen, T., David, G., Hämäläinen, T.: Survey: intrusion detection systems in encrypted traffic. In: Galinina, O., Balandin, S., Koucheryavy, Y. (eds.) NEW2AN/ruSMART -2016. LNCS, vol. 9870, pp. 281–293. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46301-8_23
4. Wang, W., Sheng, Y., Wang, J., et al.: HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2017)
5. Zhao, S., Ma, X., Zou, W., Bai, B.: DeepCG: classifying metamorphic malware through deep learning of call graphs. In: Chen, S., Choo, K.-K.R., Fu, X., Lou, W., Mohaisen, A. (eds.) SecureComm 2019. LNICST, vol. 304, pp. 171–190. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-37228-6_9
6. Tencent Security: https://s.tencent.com/research/report/790.html. August 2019
7. Kumar, V., Sangwan, O.P.: Signature based intrusion detection system using SNORT. Int. J. Comput. Appl. Inf. Technol. **1**(3), 35–41 (2012)
8. Timcenko, V., Gajin, S.: Ensemble classifiers for supervised anomaly based network intrusion detection. In: 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 13–19. IEEE (2017)
9. AlAhmadi, B.A., Martinovic, I.: Malclassifier: alware family classification using network flow sequence behaviour. In: 2018 APWG Symposium on Electronic Crime Research (eCrime), pp. 1–13. IEEE (2018)

10. Kohout, J, Pevný, T.: Unsupervised detection of malware in persistent web traffic. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1757–1761. IEEE (2015)
11. Alom, M.Z., Taha, T.M.: Network intrusion detection for cyber security using unsupervised deep learning approaches. In: 2017 IEEE National Aerospace and Electronics Conference (NAECON), pp. 63–69. IEEE (2017)
12. Bartos, K, Sofka, M, Franc, V.: Optimized invariant representation of network traffic for detecting unseen malware variants. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 807–822 (2016)
13. Li, H., Chen, Z., Spolaor, R., et al.: DART: detecting unseen malware variants using adaptation regularization transfer learning. In: ICC 2019–2019 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2019)
14. Zhao, J., Shetty, S., Pan, J.W.: Feature-based transfer learning for network security. In: MILCOM 2017–2017 IEEE Military Communications Conference (MILCOM), pp. 17–22. IEEE (2017)
15. Zhao, J., Shetty, S., Pan, J.W., et al.: Transfer learning for detecting unknown network attacks. EURASIP J. Inf. Secur. **2019**(1), 1 (2019)
16. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
17. Deng, J., Dong, W., Socher, R., et al.: Imagenet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
18. Rezende, E., Ruppert, G., Carvalho, T., et al.: Malicious software classification using transfer learning of resnet-50 deep neural network. In: 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1011–1014. IEEE (2017)
19. Rezende, E., Ruppert, G., Carvalho, T., Theophilo, A., Ramos, F., Geus, P.: Malicious software classification using VGG16 deep neural network's bottleneck features. In: Latifi, S. (ed.) Information Technology - New Generations. AISC, vol. 738, pp. 51–59. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77028-4_9
20. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
21. Wang, X., Jin, Y., Long, M., et al.: Transferable normalization: towards improving transferability of deep neural networks. In: Advances in Neural Information Processing Systems, pp. 1951–1961 (2019)
22. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
23. Hubballi, N., Suryanarayanan, V.: False alarm minimization techniques in signature-based intrusion detection systems: a survey. Comput. Commun. **49**, 1–17 (2014)
24. Long, M., Zhu, H., Wang, J., et al.: Deep transfer learning with joint adaptation networks. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2208–2217. JMLR. org (2017)
25. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2009)
26. Long, M., Wang, J., Ding, G., et al.: Adaptation regularization: a general framework for transfer learning. IEEE Trans. Knowl. Data Eng. **26**(5), 1076–1089 (2013)

27. Wiesler, S., Ney, H.: A convergence analysis of log-linear training. In: Advances in Neural Information Processing Systems, pp. 657–665 (2011)
28. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., et al.: Dataset Shift in Machine Learning. The MIT Press, Cambridge (2009)
29. Student: The probable error of a mean. Biometrika **6**(1), 1–25 (1908)
30. Wang, W., Zhu, M., Zeng, X., et al.: Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN), pp. 712–717. IEEE (2017)
31. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105 (2012)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Stratosphere: Stratosphere Laboratory Datasets. (2015). Retrieved March 13, 2020, fromhttps://www.stratosphereips.org/datasets-overview
34. Borgwardt, K.M., Gretton, A., Rasch, M.J., et al.: Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics **22**(14), e49–e57 (2006)