Received February 22, 2019, accepted March 21, 2019, date of publication April 1, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2908225

Deep-Full-Range: A Deep Learning Based Network Encrypted Traffic Classification and Intrusion Detection Framework

YI ZENG[1,2], HUAXI GU[1], WENTING WEI[1], AND YANTAO GUO[2]
[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[2]Science and Technology on Communication Networks Laboratory, Shijiazhuang 050081, China

Corresponding author: Huaxi Gu (hxgu@xidian.edu.cn)

This work was supported in part by the National Science Foundation of China under Grant 61634004 and Grant 61472300, in part by the Fundamental Research Funds for the Central Universities under Grant JB180309 and Grant JB170107, in part by the Key Research and Development Plan of Shaanxi Province under Grant 2017ZDCXL-GY-05-01, in part by the Open Fund Project of the Science and Technology on Communication Networks Laboratory under Grant HHS19641X001, and in part by the China Postdoctoral Science Foundation under Grant 2018M633465.

ABSTRACT With the rapid evolution of network traffic diversity, the understanding of network traffic has become more pivotal and more formidable. Previously, traffic classification and intrusion detection require a burdensome analyzing of various traffic features and attack-related characteristics by experts, and even, private information might be required. However, due to the outdated features labeling and privacy protocols, the existing approaches may not fit with the characteristics of the changing network environment anymore. In this paper, we present a light-weight framework with the aid of deep learning for encrypted traffic classification and intrusion detection, termed as deep-full-range (DFR). Thanks to deep learning, DFR is able to learn from raw traffic without manual intervention and private information. In such a framework, our proposed algorithms are compared with other state-of-the-art methods using two public datasets. The experimental results show that our framework not only can outperform the state-of-the-art methods by averaging 13.49% on encrypted traffic classification's $F1$ score and by averaging 12.15% on intrusion detection's $F1$ score but also require much lesser storage resource requirement.

INDEX TERMS Encrypted traffic classification, network intrusion detection, deep learning, end-to-end.

I. INTRODUCTION

An accurate traffic classification has been the prerequisite for network basic tasks, say providing appropriate Quality-of-Service (QoS), Network Intrusion Detection (NID), etc. Yet traffic classification has been becoming more and more challenging due to the following fact. The first one is the great boom in traffic diversity and variety. Apart from that, more and more applications have started to apply security protocols, such as HTTPS, SSH, SSL, to encrypt internet traffic for the sake of user's privacy; The third fact is that some of the basic information, like flow volume or flow duration, is not easy to be acquired nowadays due to privacy protocols and laws. In a nutshell, how to achieve a high-quality traffic identification without private information has drawn significant attention in the concerning of cyber security and QoS.

Previous methods of traffic classification, like the Port Number Based method and the Data Packet Inspection (DPI) Based method [1], are not competent enough for modern traffic environment due to their disability toward encrypted traffic. The newest generation of traffic classification method is based on flow-statistics and Machine Learning (ML), which can cope with both encrypted and normal traffic, like the K-Nearest Neighbor (KNN) and the Decision Tree (DT) that used in Atli's work [2].

However, ML-Based approaches' performance highly depends on the human-engineered features and some private

The associate editor coordinating the review of this manuscript and approving it for publication was Irene Amerini.

traffic information, hence dramatically limits their accuracy and generalizability. Moreover, ML-Based classification methods usually require high storage and computational resource, which limited the implementation of it in resources constrained nodes, like vehicles [3], home gateways, and mobile phones. A real-time accurate network traffic classifier is the foundation of network management tasks and NID systems, therefore, a novel classification method is of urgent need.

Methods based on Deep Learning (DL) obviate the burdensome work of selecting features and acquiring private features information since it automatically extracts and selects features through training [4]. This characteristic has made DL-based methods a highly desirable approach for traffic classification. Another characteristic of DL-based methods is that DL has a higher learning capability in comparison with traditional ML methods like Random Forest, Support Vector Machine, and KNN [2]. Thus, they are expected to learn highly complicated patterns in order to gain a higher accuracy than previous methods with more functionality, like NID, critical link analysis, traffic classification, and so on, within just one framework. Take those two characteristics into mind, as an end-to-end approach, DL-based methods are capable of effectively learning relationships between raw traffic and corresponding output without the need of requiring heavy labor and private information.

In this paper, we present a novel network encrypted traffic classification and intrusion detection framework, named Deep-Full-Range (DFR), where three deep learning algorithms — Convolutional Neural Network (CNN) [5], Long Short-Term Memory (LSTM) [6], and Stacked Auto-Encoder (SAE) [7]— are employed. We intend to use CNN to learn features of the raw traffic from spatial range. LSTM is used to learn features from the time-related aspect. SAE is adopted to extract features from coding characteristics. All those three aspects are combined to gain a deep and full ranged understanding of the raw input. With a full-range concerned structure, DFR is capable of classifying encrypted traffic and malware traffic within one framework without human's help and private details. Since our framework will only save the model that most effective with the current traffic environment, compared with previous methods, a dramatic drop in the storage resource requirement is met.

This paper's contributions can be summarized as follows: **1)** A light-weight classification and intrusion detection framework, named DFR, is proposed. The effectiveness of completing those two tasks is evaluated on two public datasets, significant improvement of effectiveness is met. **2)** DFR is based on three DL models which can acquire a deep and full-ranged understanding of the raw traffic. **3)** DFR is an end-to-end framework, does not require too much human's help and private information of the data.

The remainder of this paper is organized as follows. The details of DFR and the methodology of each part is illustrated in Section II. We evaluate the framework by comparing with other state-of-art methods on two public datasets in Section III. Finally, Section IV gives the concluding remark of this paper.

II. THE DFR FRAMEWORK

In this section, we will illustrate the details of our proposed network encrypted traffic classification and intrusion detection framework. DFR framework can be divided into two functional modules, as shown in Fig.1: *Preprocessing Procedure*, *DFR Procedure*.

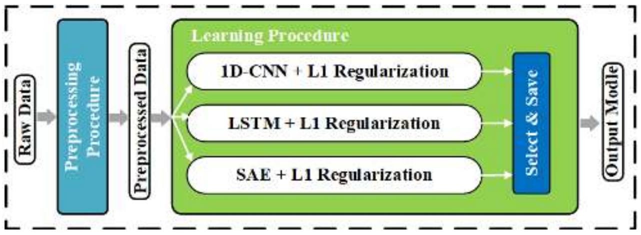

FIGURE 1. Overview of the DFR framework.

The notations and definitions of some of the parameters used in the DFR model are listed in Table 1.

TABLE 1. Notations & parameters used in the DFR.

Notation	Description
T	The total number of time units
J	The total number of generated traffic-graphs
RT^t	The raw traffic of time unit $t, t \in (1, T)$
PT_i^t	The i^{th} packet that captured from RT^t
$G(j)$	The j^{th} traffic-graph generated, $j \in (1, J)$
$Epoch$	The number of epochs to train
$Minbatch$	The batch size
LR	The learning rate of the optimizer
$KeepP$	The keep probability of the dropout process
N	The total classes to be classified or identified
$Lambda$	The lambda parameter of the L1 regularization
$EpochFin$	Epochs for the fine tune of SAE-Based DFRs
$LambdaFin$	Lambda in the fine tune of SAE-Based DFRs
G_{Train}	The training dataset
G_{Test}	The testing dataset

A. PREPROCESSING PROCEDURE

In this process, our presented framework will convert raw network traffic data into IDX format, which is the format that we used as the input of the following procedure. The reason why we preprocess the raw data has 3 following reasons. **1)** The raw data from the network is of different length, which is not an ideal input format for DL models. **2)** The raw data contains some information that might interfere with the result, say the port number or the MAC address. **3)** An unified format can pave the way for following works like adding more network applications, the maintenance of the procedure, and etc. Fig.2 illustrates the overview of the preprocessing procedure, which consists of 5 steps.

Package Generation is the step to split continues raw traffic data and save as PCAP files by a package-capture tool — Wireshark — [8]. The method that we proposed will

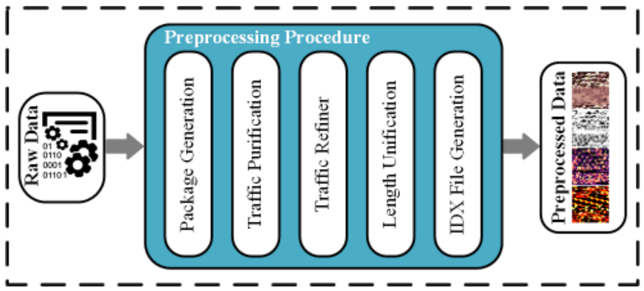

FIGURE 2. Overview of the preprocessing procedure.

capture the Bi-directional flow of all layers, which is the most ideal traffic representation for DL-Based traffic classification as evaluated in Wang's paper [9].

Traffic Purification is the step to eliminate the interferential data in traffic packages. This includes TCP or UDP headers and some Ethernet related data in the data link layer, for example, the MAC address, since they are no use for the following procedure and might even harm the result.

Traffic Refiner will remove duplicated files and empty files, as those files will only harm the learning ability of our framework.

Length Unification will trim files that longer than 900 bytes to 900 bytes, and add 0x00s at the end of files that shorter than 900 bytes to complement it to 900 bytes.

IDX Fill Generation is when we will convert those length-unified PCAP files into 30 bytes $\times$ 30bytes two-dimensional format IDX files. IDX format is a common file format in DL field [10], which is as well ideal for our framework. Those files are regarded as gray-scale images if mapped to [0,1]. In Fig.2, we can see the preprocessing result as gray-scale images.

Further details of the Preprocessing Procedure is presented in Alg. 1. After this procedure, a dataset full of traffic-graph is generated for future use.

Algorithm 1 Preprocessing Algorithm

```
Input:
  $RT^1, RT^2, ..., RT^t, ..., RT^T$
Output:
  Preprocessed Dataset $G(1, 2, ...j, ...J)$
 1: for each $t$ in $(1, T)$ do
 2:     Split the $RT^t$ into $PT_1^t, PT_2^t, ...PT_i^t, ...$
 3:     for each $i$ do
 4:         Purify $PT_i^t$ following 'Traffic Purification'
 5:     end for
 6:     remove duplicated packet from $PT_1^t, PT_2^t, ...PT_i^t, ...$
 7:     for each $i$ do
 8:         Cut the length of $PT_i^t$ to 900 bytes
 9:         Generate traffic-graph $G$
10:     end for
11: end for
```

B. DFR PROCEDURE

As shown in Fig.1, the DFR procedure is based on three DL models, which are CNN, LSTM, and SAE respectively. The reason why we adopted L1 regularization [11] in all the models is that L1 regularization is capable of punishing some weights to 0 that L2 cannot, which is helpful for the machine to learn which feature is irrelevant to the classification, hence a better result can be attained. We will explain more about selecting L1 regularization based on the experimental result in the following section.

1) 1D CNN-BASED DFR CLASSIFIER

We develop the first encrypted traffic classification and intrusion detection identifier based on one dimensional CNN. So far, CNN has been mainly applied in the computer vision respect, say image classification, object detection. CNN has a strong ability to learn the spatial properties of a graph pixel by pixel. We intended to use CNN to find features that help machine to classify traffic from spatial range. We picked 1D CNN over the popular 2D version because of the original traffic format is a sequential form organized by hierarchical structure rather an image. As shown in a relevant work based on CNN to classify traffic [9], it is proved that 1D CNN can attain a higher accuracy than 2D CNN.

The 1D CNN-Based DFR Classifier is illustrated in Fig.3. It consists of two convolutional layers, 2 Maxpooling layers, 2 *Local Response Normalization* (LRN) [5] layers, and a densely connected layer with Softmax classifier. In the first stage, we will reshape the input data as 1×900 shape, then we will discard height and focus on processing the 1D data. The classification process is defined as follows:

The first convolutional layer processes the input data with 32 filters, where each filter has a size of [25, 1]. Each filter moves 1 step after one convolution operation. The results of the convolutional layer are inputted to an activation function. We adopted ReLU [12] activation function in our 1D CNN-Based DFR Classifier. Then, the results are processed through max pooling. In each step, the max pooling processes a [3, 1] input as follows:

$$maxpooling\,[x1, x2, x3] = \max(x1, x2, x3) \tag{1}$$

And the stride of the maxpooling process is 3. At the end of the first convolutional layer, an LRN layer is added to punish those abnormal responses or out layers in order to attain a better generalization. Then the output will go through a second convolutional layer which is similar to the first one. The only difference between those two convolutional layers is that the second convolutional layer has 64 filers. Finally, the data will go through a densely connected layer. This layer is gained by applying dropout on a fully connected layer. Then the output label is attained by the softmax classifier at the end of 1D CNN-Based DFR Classifier. The softmax classifier is defined as follows:

$$\hat{y} = \frac{\exp(Out^j)}{\sum \exp(Out^i)} \tag{2}$$

where Out^j is the output of jth neuron in the densely connected layer. $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3, ..., \hat{y}_N\}$ is the complete set of

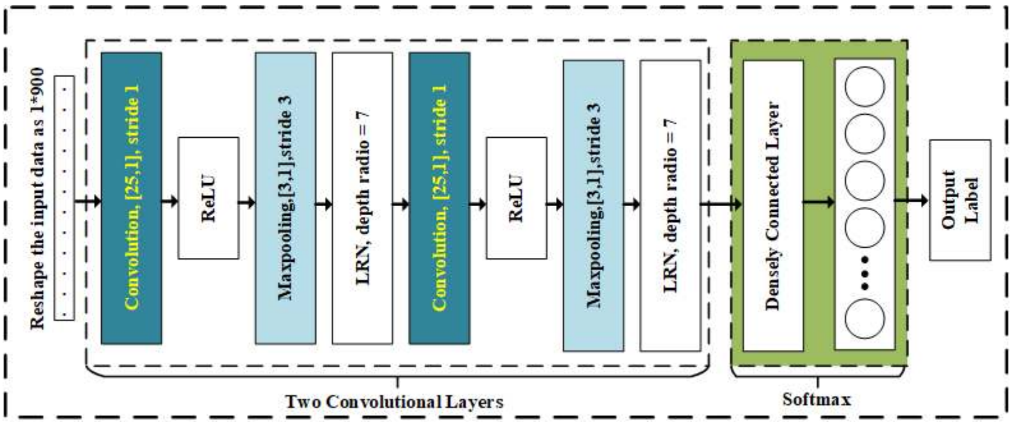
Reshape the input data as 1*900
Convolution, [25,1], stride 1
ReLU
Maxpooling,[3,1],stride 3
LRN, depth radio = 7
Convolution, [25,1], stride 1
ReLU
Maxpooling,[3,1],stride 3
LRN, depth radio = 7
Densely Connected Layer
Output Label
Two Convolutional Layers
Softmax

FIGURE 3. Overview of the 1D CNN-Based DFR Classifier.

classes, and N denotes the total number of classes. The output with the highest probability indicates the class of the input value. To be notified that we used Adam [13] optimizer in this DFR.

For the training process, hyperparameters are set as $\{Epoch, Minbatch, LR, KeepP, N, Lambda\}$, where the definition of those parameters are descripted in Table 1. Further details of training a 1D CNN-Based DFR is explained in Alg. 2.

Algorithm 2 1D CNN-Based DFR Training Algorithm

```
Input:
    $G_{Train}$,
    $\{Epoch, Minbatch, LR, KeepP, N, Lambda\}$
Output:
    1D CNN-Based DFR
1:  for each epoch in (1, Epoch) do
2:      for each batch of Minbatch data do
3:          for each G in batch do
4:              Reshape G to $1 \times 900$ form
5:              Compute convolution with 32 filters
6:              Compute the result through ReLU
7:              Max pooling referring Eq. 1
8:              Punish the result through the LRN layer
9:              Compute convolution with 64 filters
10:             Compute the result through ReLU
11:             Max pooling referring Eq. 1
12:             Punish the result through the LRN layer
13:             Run through a densely connected layer
14:             Output the result according to Eq. 2
15:             Update the weight & bias
16:         end for
17:     end for
18: end for
```

2) LSTM-BASED DFR CLASSIFIER

The second DFR is developed based on LSTM [6], a kind of network which is designed to deal with sequential-form-data. LSTM is one kind of *Recurrent Neural Network* (RNN) that can utilize the time-related information. Taken that a segment of traffic is built up byte by byte and package by package, which is time-related, traffic from similar classes must share some similarity in the time-related characteristics. We apply LSTM in DFR to help the machine to learn the time-related characteristics on its own. Previously, only a few works mentioned using RNN to classify network traffic. Torres *et al.* [14] transformed the traffic byte-data into character-data then LSTM is applied to learn the connections between characters. Yet, in our proposed LSTM-Based DFR Classifier, the input will be a graph. Fig.4 illustrates the overview of our proposed LSTM-Based DFR Classifier.

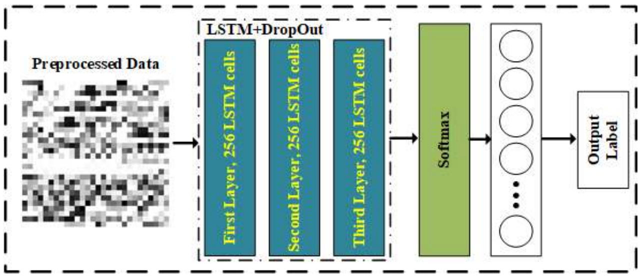
Preprocessed Data
LSTM+DropOut
First Layer, 256 LSTM cells
Second Layer, 256 LSTM cells
Third Layer, 256 LSTM cells
Softmax
Output Label

FIGURE 4. Overview of the LSTM-Based DFR classifier.

As shown in Fig.4, the LSTM-Based DFR Classifier is actually based on a three-layered LSTM model. For each layer, LSTM-Based DFR Classifier havs 256 LSTM cells. And we also applied dropout in every layer of the LSTM model in order to gain a better generalization. After the LSTM model learned the time-related characteristics, the data will go through a softmax classifier which is described in Equation (2). Finally, the result label can be checked at the end of the DFR. It is worth to mention that we used Adam optimizer in this DFR as well. Training a LSTM-Based DFR Classifier can use the same hyperparameters that defined in 1D CNN-Based DFR Classifier, which are set as $\{Epoch, Minbatch, LR, KeepP, N, Lambda\}$. The training of the LSTM-Based DFR Classifier is explained in Alg. 3.

Algorithm 3 LSTM-Based DFR Training Algorithm

```
Input:
  $G_{Train}$,
  $\{Epoch, Minbatch, LR, KeepP, N, Lambda\}$
Output:
  LSTM-Based DFR
1:  for each epoch in (1, Epoch) do
2:      for each batch of Minbatch data do
3:          for each G in batch do
4:              Compute G through the 3-layered LSTM
5:              Dropout according to KeepP
6:              Output the result according to Eq. 2
7:              Update the weight & bias
8:          end for
9:      end for
10: end for
```

3) SAE-BASED DFR CLASSIFIER

The last DFR's core is two SAEs. In general, autoencoder is a kind of semi-supervised learning method that used for automatic features extraction. SAE is one of the variations of autoencoder, which has the ability to scan through the data byte by byte to find the coding characteristics. Fig.5 demonstrates the overview of our SAE-Based DFR Classifier.

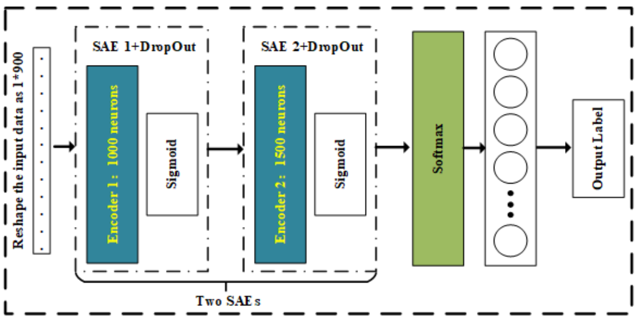

FIGURE 5. Overview of the SAE-Based DFR classifier.

The first step is reshaping the graph to have a form of 1×900, hence we can fully connect our input with the first encoder. Fig.6 shows the process of training those two SAEs. We can tell from Fig.6 that those two SAEs are trained separately. The first encoder has 1000 neurons which are densely connected with the 900 inputs and 900 outputs. The goal of training Encoder 1 is to gain an encoder which can generate 900 outputs that have the minimum variance with the 900 inputs. After the Encoder 1 is trained, we will stack it in the DFR and apply a sigmoid [15] activation function. Encoder 2 have 1500 neurons, which will be densely connected with the outputs from Encoder 1. Then we will train the Encoder 2 by reducing the variance between input 2 and output 2, which is similar to the training of Encoder 1. We will stack it in the DFR with a sigmoid activation function applied as well. Finally, the data will go through the softmax classifier which defined in Equation (2) and output the results. To be noticed that after those two SAEs stacked into the DFR, a fine tune training procedure will start in order to attain the final model.

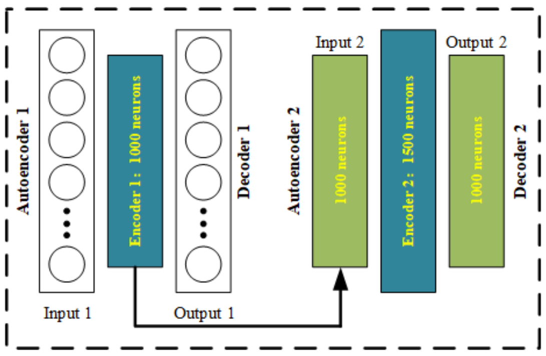

FIGURE 6. The training process of the two encoders.

Since the two training processes of SAEs are separated, some of the hyperparameters are different from the previous two DFRs. All the hyperparameters will be set as $\{Epoch, EpochFin, Lambda, LambdaFin\}$ and $Setsame = \{Minbatch, LR, KeepP, N\}$, where *Setsame* share the same definition with the previous two DFRs. *Epoch* is the number of epochs of the two-SAEs training process, which cannot be set as a very large number. A large *Epoch* in the two-SAEs training process can misguide the model to overfit with the training data. *EpochFin* is the number of epochs in the fine tune process, this number does not have the limitation as *Epoch*. *Lambda* is the lambda parameter of the L1 regularization in the two-SAEs training process, which need to be set as a number much bigger than the lambda parameter of the L1 regularization in the fine tune process, namely *LambdaFin*. The reason is that a small *Lambda* is not effective with the two SAEs training process. The training of a SAE-Based DFR is summarized in Alg. 4.

Algorithm 4 SAE-Based DFR Training Algorithm

```
Input:
  $G_{Train}$,
  $Setsame = \{Minbatch, LR, KeepP, N\}$,
  $\{Epoch, EpochFin, Lambda, LambdaFin\}$
Output:
  SAE-Based DFR
1:  for each one of the two encoders do
2:      for each epoch in (1, Epoch) do
3:          for each batch of Minbatch data do
4:              for each G in batch do
5:                  Compute the output of Autoencoder
6:                  Process Sigmoid
7:                  Dropout according to KeepP
8:                  Compute the cost between output and input
9:                  Update the weight & bias
10:             end for
11:         end for
12:     end for
13:     Current Encoder connects with the successive layer
14: end for
15: for each epoch in (1, EpochFin) do
16:     Output the result according to Eq. 2
17:     Update the weight & bias
18: end for
```

4) SELECT AND SAVE

After those three DFRs are trained with training data, we will exam those DFRs with testing data. The DFR that holds the highest accuracy is believed to be the model that best fits with the current traffic environment. Where accuracy is defined as follows:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{3}$$

where TP is *True Positive*, namely the number of correctly classified cases as a specific class; FP is *False Positive*, namely the number of misclassified cases that classified as that class; FN, *False Negative*, which is the number of cases that are supposed to be classified as that class, yet misclassified as other classes; TN, *True Negative*, which is the number of cases that correctly classified as not that specific class.

Hence, this model will be the model that we saved and conduct the following classification and identification tasks for this time units. DFR will be trained in an online fashion, which means DFR has the ability to update itself in order to fit with the traffic environment automatically. Alg. 5 explains other details in the processes of the online-fashioned DFR framework.

Algorithm 5 Online-Fashioned DFR Algorithm

```
Input:
  $RT^1, RT^2, ..., RT^t, ..., RT^T$
Output:
  The DFR model that best fits with the current traffic environment
 1: for each $t$ in $(1, T)$ do
 2:     Apply Alg. 1 to gain $G(1, 2, ...j, ...J)$
 3:     Randomly separate $G(1, 2, ...j, ...J)$ according to 9:1;
 4:     Use $G_{Train}$ to train the three DFRs;
 5:     Use $G_{Test}$ to select the highest-accuracy DFR;
 6:     Run the current-using DFR on the same $G_{Test}$;
 7:     if the current-using DFR's accuracy is smaller then
 8:         Save the new DFR;
 9:         Transmit the new DFR to other units
10:     end if
11: end for
```

III. EVALUATION

In this section, we will evaluate the DFR framework from three aspects, which are the efficiency of encrypted traffic classification, the efficiency of intrusion detection, and the storage resource requirement.

A. EXPERIMENTAL SETTINGS

We will first describe the two selected datasets that we used. Then, the setup of the experiment is revealed with details of hyperparameters in the training process. Finally, we will explain the metrics that we used.

1) DATASETS FOR EVALUATION

Currently, we have not yet found an available public dataset that has both encrypted traffic and malware traffic, hence we decide to evaluate the DFR framework using two selected public datasets, ISCX VPN-nonVPN traffic dataset [16] and ISCX 2012 IDS dataset [17] respectively.

The first selected dataset is regenerated from ISCX VPN-nonVPN traffic dataset [16] in order to evaluate the effectiveness of DFR on encrypted traffic classification. ISCX VPN-nonVPN dataset originally has 7 types of regular encrypted traffic and 7 types of protocol encapsulated traffic. Since we mainly focus on evaluating the efficiency on encrypted traffic classification, we will select and label data from those 7 types of regular encrypted traffic, which are Web Browsing, Email, Chat, Streaming, File Transfer, VoIP, and P2P. To be noticed that all other six types of encrypted traffic are related to Web Browsing, hence we abandoned this class of encrypted traffic referring to Wang's work [9]. Moreover, some of the classes in the original dataset have much more instances than others, which might misguide the model in the training process. Hence we cut them short to gain a balanced dataset. The structure of our selected dataset is shown in Table 2. After normalization applied, each class of traffic have a quantity around 10000 cases. This dataset will be automatically divided into the training dataset and the testing dataset according to 9:1. As a result, the training dataset will have 52916 cases, and the testing dataset will have 5880 cases.

TABLE 2. Structure of the selected dataset 1.

Class Name	Protocols	Quantity	Percentage (%)
Email	SSL & HTTPS	9417	16.01
Chat	HTTPS	10000	17.01
Streaming	HTTPS	10000	17.01
File Transfer	HTTPS	9729	16.55
VoIP	HTTPS	10000	17.01
P2P	HTTPS	9650	16.41
TOTAL	ALL	58796	100

TABLE 3. Structure of the selected dataset 2.

Class Name	Quantity	Percentage (%)
Normal	848306	95.32
Brute Force SSH	6964	0.78
DDoS	21121	2.37
HttpDoS	3482	0.39
Infiltrating Transfer	10044	1.13
TOTAL	889917	100

The second dataset is regenerated from ISCX 2012 IDS dataset [17] in order to evaluate the effectiveness of DFR on intrusion detection. This dataset contains the network traffic of seven days, which can be divided into 5 classes, Normal, Brute Force SSH, DDoS, HttpDoS, and Infiltrating respectively. We select the data that only from days that have malware traffic for simplification. Since malware traffic is expected to have a relatively smaller scale compared to normal traffic in real life, no normalization approach on the dataset is applied. Table 3 demonstrates the structure of the selected dataset. Finally, this selected dataset is divided into the training dataset and the testing dataset according to 9:1 for each class.

2) EXPERIMENT SETUP

Tensorflow is used as the experiment ML software framework. We run the evaluation on Ubuntu 18.04 64 bit OS. The processor is an 8 cores Intel I7-7700K CPU with 32 GB of memory. Two chips of Nvidia GeForce GTX 1080 Ti is used as the GPU accelerator. The hyperparameters for the DFR training process is shown in Table 4.

TABLE 4. **Settings of hyperparameters.**

	1D CNN-Based	LSTM-Based	SAE-based
$Epoch$	160000	160000	400
$Minibatch$	200	200	200
LR	0.0001	0.0001	0.001
$KeepP$	0.5	0.5	0.5
$Lambda$	0.0003	0.00001	0.01
$EpochFin$	×	×	150000
$LambdaFin$	×	×	0.00002

3) EVALUATION METRICS

We use the Accuracy that defined in Equation (3) to select the optimum DFR. We evaluate and compare the performance of the selected DFR with state-of-art methods using three metrics. Namely, Precision, Recall, and F1 score, which are defined as follows:

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

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

$$F1\_score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6}$$

where the definition of TP, FP, and FN can be viewed at the section II-B.4.

B. EXPERIMENT RESULT

We will first present the evaluation of the encrypted traffic classification efficiency. The reason why L1 regularization is adopted is explained with experimental results. Then, the intrusion detection efficiency is evaluated. Finally, a comparison of storage resource requirement is presented.

1) ENCRYPTED TRAFFIC CLASSIFICATION EFFICIENCY

The accuracy of three trained DFR on the testing dataset of dataset 1 is shown in Table 5. We also evaluate the effectiveness of applying L1 regularization in this experiment.

TABLE 5. **Accuracy of DFRs on encrypted traffic.**

DFR Name	Accuracy (%)
1D CNN-Based with L1 regularization	99.85
1D CNN-Based with L2 regularization	95.84
LSTM-Based with L1 regularization	99.22
LSTM-Based with L2 regularization	97.33
SAE-Based with L1 regularization	98.74
SAE-Based with L2 regularization	94.54

As we can tell from Table 5, DL models can attain an averaging 3.37 percents higher accuracy after replacing L2 regularization with L1 regularization. This has proved our analysis of L1 and L2 regularization in section II-B, hence we will only apply L1 regularization in our DFR framework. The 1D CDD-Based, attained the highest accuracy, 99.85%, therefore we will save this model as the optimum DFR classifier for the current network traffic environment.

We also compare the performance of our DFR framework with two state-of-art methods on the same dataset. The first method is from the work [16], C4.5 DT. This is a ML-Based classification method that requires human-engineered features. The second method is the 1D-CNN classification method that proposed in the work [9], which is a similar model as we used in DFR, yet no regularization approach and LRN applied in their 1D-CNN.

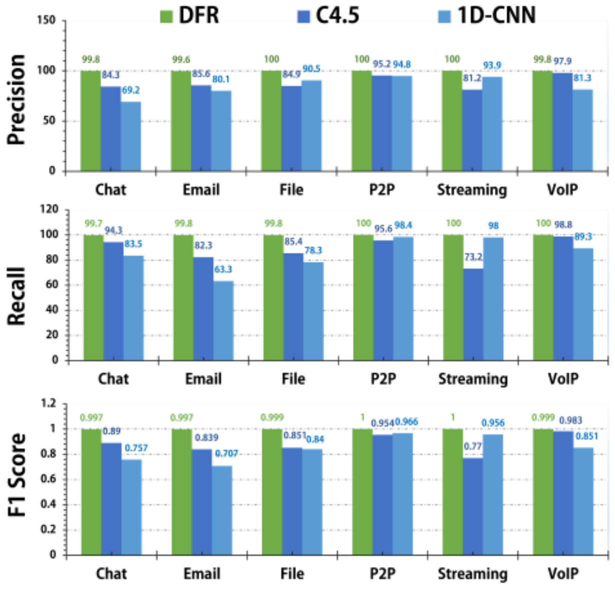

FIGURE 7. **Comparison of the classification efficiency.**

As shown in Fig.7, we can tell the DFR framework can attain a much accurate and robust classification result than ML-Based C4.5 method and DL-Based 1D-CNN method. The averaging F1 score of the DFR framework is 0.9987, which is kind of impressive comparing to the averaging F1 score of the C4.5 DT and 1D-CNN. The averaging improvement of precision compared to C4.5 is 11.6 percents, to 1D-CNN is 14.9 percents. All these data are supporting the fact that the DFR framework is capable of classifying encrypted traffic accurately.

2) INTRUSION DETECTION EFFICIENCY

As for the dataset 2, the LSTM-Based DFR Classifier attained an accuracy of 99.41 %, which is the highest of the three DFR classifiers. We then compared the DFR framework with two state-of-art ML-Based methods that demonstrated in Atli's work [2], namely DT, and KNN respectively.

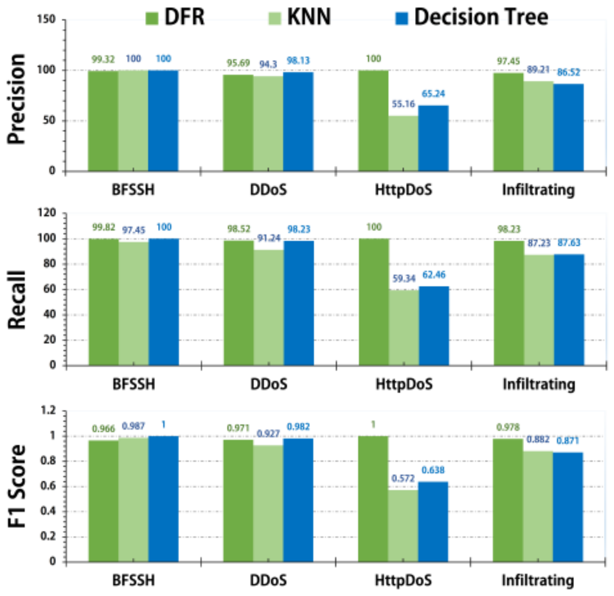

FIGURE 8. **Comparison of the intrusion detection efficiency.**

Those are two kinds of ML-Based methods for intrusion detection.

As shown in Fig.8, DFR is able to maintain a robust and high detection performance on real-life malware traffic. The averaging improvement on the F1 score is 0.137 and 0.106 comparing to those two methods. It is worth to mention that, the proposed framework is able to identify the HttpDoS malware traffic at a 1 out of 1 F1 score, which has major improvement compared with DT and KNN. All these data demonstrate the capability of an accurate intrusion detection.

3) STORAGE REQUIREMENT COMPARISON

Finally, we compare the storage resource requirement of the aforementioned methods. An 1D CDD-Based DFR Classifier's trained file is 1534KB, LSTM-Based DFR Classifier requires 186KB to store, and the SAE-Based DFR Classifier's trained file is 12732KB. Comparing to the averaging file size of KNN model and DT model, which are 67548KB and 34247KB, a dramatic drop in the storage requirement is met.

IV. CONCLUSION

In this paper, we presented a novel DL-Based framework which is capable of classifying encrypted traffic and detecting malware traffic. Compared to state-of-art ML-Based methods, our DFR framework does not require the heavy work of selecting features and private featured details. Moreover, based on the results over two public datasets, it is proved that DFR can attain a much more robust and accurate performance on both encrypted traffic classification and intrusion detection than state-of-art methods with a less storage resource requirement.

REFERENCES

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

[2] B. G. Atli, Y. Miche, A. Kalliola, I. Oliver, S. Holtmanns, and A. Lendasse, "Anomaly-based intrusion detection using extreme learning machine and aggregation of network traffic statistics in probability space," *Cogn. Comput.*, vol. 10, no. 5, pp. 848–863, 2018.

[3] Y. Zeng, M. Qiu, Z. Ming, and M. Liu, "Senior2Local: A machine learning based intrusion detection method for vanets," in *Proc. Int. Conf. Smart Comput. Commun.*, 2018, pp. 417–426.

[4] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018.

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

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[8] A. Orebaugh, G. Ramirez, and J. Beale, *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Amsterdam, The Netherlands: Elsevier, 2006.

[9] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Inform. (ISI)*, Jul. 2017, pp. 43–48.

[10] M. Fatahi, "Mnist handwritten digits," Tech. Rep., 2014. [Online]. Available: https://www.researchgate.net/publication/273124795_MNIST_handwritten_digits_Description_and_using

[11] M. Y. Park and T. Hastie, "L_1-regularization path algorithm for generalized linear models," *J. Roy. Stat. Soc., B (Stat. Methodol.)*, vol. 69, no. 4, pp. 659–677, 2007.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[13] D. P. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

[14] P. Torres, C. Catania, S. Garcia, and C. G. Garino, "An analysis of recurrent neural networks for botnet detection behavior," in *Proc. IEEE Biennial Congr. Argentina (ARGENCON)*, Jun. 2016, pp. 1–6.

[15] X. Yin, J. Goudriaan, E. A. Lantinga, J. Vos, and H. J. Spiertz, "A flexible sigmoid function of determinate growth," *Ann. Botany*, vol. 91, no. 3, pp. 361–371, 2003.

[16] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, 2016, pp. 407–414.

[17] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

YI ZENG is currently pursuing the bachelor's degree, majoring in electronic and information engineering, with Xidian University. He has been with The State Key Laboratory of Integrated Service Networks (ISN), Xidian University, since 2017. His main research interests include machine learning, cyber systems' security issues, network traffic recognition, and network virtualization.

HUAXI GU received the B.E. degree and the M.E. and Ph.D. degrees in telecommunication engineering and telecommunication and information systems from Xidian University, Xidian, China, in 2000, 2003, and 2005, respectively, where he is currently a Full Professor with The State Key Laboratory of Integrated Service Networks (ISN), Telecommunication Department. He has authored or coauthored more than 100 publications in refereed journals and conferences. His current interests include interconnection networks, networks-on-chip, and optical inter-chip communication. He is a Reviewer for the IEEE Transactions on Computer, the IEEE Transactions on Dependable and Secure Computing, the IEEE Systems Journal, IEEE Communication Letters, *Information Sciences*, the *Journal of Supercomputing*, the *Journal of System Architecture*, the *Journal of Parallel and Distributed Computing*, and *Microprocessors and Microsystems*.

WENTING WEI received the M.E. degree from the School of Telecommunications Engineering, Xidian University, in 2014. She is currently pursuing the Ph.D. degree in telecommunication and information systems with The State Key Laboratory of Integrated Service Networks (ISN), Xidian University. Her main research interests include data center networks, optical interconnected networks, and network virtualization.

YANTAO GUO received the M.S. degree in electrical engineering from Hebei Technology University, Tianjin, China, in 1990, and the Ph.D. degree in system and network from Xi'an University, Xi'an, China, in 2008. He is currently the Deputy Director of the National Key Laboratory on Communication Networks, The 54th Research Institute, CETC. His current research interests include networks and communication system designing.