This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Encrypted Network Traffic Classification Using Deep and Parallel Network-In-Network Models

ZHIYONG BU[1,2], BIN ZHOU[1,2], PENGYU CHENG[3], KECHENG ZHANG[1,2] and ZHENHUA LING[3]

[1]Key Laboratory of Wireless Sensor Network and Communications, Chinese Academy of Sciences, Shanghai, 200050, China
[2]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, 200050, China
[3]University of Science and Technology of China, Hefei, 230027, China

Corresponding author: Zhenhua Ling (e-mail: zhling@ustc.edu.cn).

ABSTRACT Network traffic classification aims to recognize different application or traffic types by analyzing received data packets. This paper presents a neural network model with deep and parallel network-in-network (NIN) structures for classifying encrypted network traffic. Comparing with standard convolutional neural networks (CNN), NIN adopts a micro network after each convolution layer to enhance local modeling. Besides, NIN utilizes a global average pooling instead of traditional fully connected layers before final classification, which reduces the number of model parameters significantly. In our proposed method, deep NIN models with multiple MLP convolutional layers are built to map fixed-length packet vectors towards application or traffic labels. Furthermore, a parallel decision strategy of building two sub-networks to process packet header and packet body separately is designed considering that they may carry different kinds of clues for classification. The results of our experiments on the "ISCX VPN-nonVPN" encrypted traffic dataset show that NIN models can achieve a better balance between classification accuracy and model complexity than conventional CNNs. The parallel decision strategy can further improve the accuracy of using single NIN model for encrypted network traffic classification. Finally, the test set F1 scores of 0.983 and 0.985 are achieved for traffic characterization and application identification respectively.

INDEX TERMS Network traffic classification, convolutional neural network, network-in-network, data packet.

I. INTRODUCTION

NETWORK traffic classification is the task of recognizing different application or traffic types by analyzing received data packets, which is important in modern communication networks [1]. Advanced network management tasks, such as guaranteeing network quality-of-service (QoS) and detecting network anomaly, relies on accurate traffic classification.

Existing methods of network traffic classification can be classified into three approaches, i.e., port-based approach, payload-based approach and machine learning approach. The port-based approach is the oldest and the simplest one [2], which extracts port numbers from the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) headers of packets to determine traffic categories. The payload-based approach, usually named deep packet inspection (DPI), analyzes the payload of packets using predefined patterns for different protocols [3]. Although these two approaches can achieve high accuracy of traffic classification in some scenarios, they suffer from the popularity of encrypted data in current communication networks. For example, the traffic of virtual private network (VPN) sessions significantly reduces the accuracy of the port-based approach. Secure transfer protocols, such as Hyper Text Transfer Protocol over Secure Socket Layer (HTTPS) and Secret File Transfer Protocol (SFTP), also increase the difficulty of recognizing application types using the payload-based approach. Therefore, the machine learning approach to traffic classification, especially to encrypted traffic classification, has attracted more and more research attentions recently. This approach considers that encrypted packets are not just sequences of totally random bits, but contains some inter-class discriminative patterns and intra-class similarities that can be captured by machine learning algorithms. This approach usually utilizes a public or self-made dataset, which contains network packets with accurate labels and is divided into two parts, a training

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

set and a test set. The former, training set, is used to train a statistical model, i.e., a classifier, which predicts the labels of the latter, test set, for performance evaluation. The conventional classifiers that have been investigated for traffic classification include k-nearest neighbor (k-NN) [4], C4.5 decision tree [5], support vector machine (SVM) [6], etc. Although these machine learning based methods can achieve better performance of encrypted traffic classification than port-based and payload-based approaches, they still have two deficiencies. First, these methods relied on manually designed features, such as flow duration, inter-arrival time, and so on. Such handcrafted feature selection constrained the robustness and generalization ability of these methods. Second, the machine learning models adopted by these methods were conventional ones with shallow structures, which limited the representation and prediction ability of these methods.

Since 2006, deep learning has emerged as a new area of machine learning research [7], [8]. Deep learning models, such as deep neural networks (DNNs), convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have be applied to various research areas, e.g., image classification [9], speech recognition [10], and natural language processing [11], and have achieved significant progresses. Comparing with conventional statistical classifiers, deep learning models are better at describing the complex and nonlinear mapping relationship from input features towards class labels. Besides, deep learning models are able to learn feature representations automatically from raw data, which alleviates the dependency on manually designed features and simplifies the pipeline of building classifiers. Therefore, such deep learning models have been introduced into machine learning based encrypted traffic classification recently [12]–[14]. Some of these studies adopted a traffic flow as the unit for classification. Wang [15] first proposed a traffic classification method based on stacked autoencoders (SAEs) for bidirectional flows (bi-flow) in 2015, which employed TCP payload as input data. An end-to-end encrypted traffic classification method was also proposed [16], in which one-dimensional (1D) CNNs were built based on raw traffic data. An automatic multitask learning system [17] was proposed for abnormal traffic detection, in which 2D-CNNs were used for bi-flow classification and the bi-flows contained packets with all layers. M. Lopez-Martin et al. [18] presented a method by stacking 2D-CNNs and RNNs, in which each packet was represented by its 6 fields and 20 packets were combined as a bi-flow. A hierarchical deep learning method was designed for malicious flow detection task [19], in which L7 layer data and manually-designed features were combined as model input. RNNs with long short-term memories (LSTMs) have also been applied to encrypted traffic classification and been compared with CNNs and SAEs under the framework of deep-full-range (DFR) [20]. Some other studies adopted a data packet as the unit for classification. A framework for encrypted traffic classification named Deep Packet was presented [21], in which CNNs achieved better performance than SAEs for classifying packets. In the DataNets method developed for SDN home gateway [22], CNNs outperformed multilayer perceptrons (MLPs) on the accuracy of packet classification. Some other issues with encrypted traffic classification, such as class imbalance [23] and multimodal learning [24], have also been studied by proposing deep-learning-based models.

Although these deep-learning-based methods have achieved significantly higher accuracy of encrypted traffic classification than conventional classifiers, such as k-NNs and decision trees, they still have some limitations. First, CNN was the dominant model structure in these studies. The convolution operations in CNNs are linear, which restricts the abstraction ability of CNN layers when learning latent representations from raw encrypted data. Second, no matter using flows or packets as the units for classification, existing models treated a packet as a whole or only utilized a specific part of packets as input data. However, different parts of packets, such as packet headers and packet bodies, may provide different kinds of clues for classification.

Therefore, this paper presents a neural network model with deep and parallel network-in-network (NIN) structures for encrypted traffic classification. The NIN model was initially proposed for image classification [25]. Comparing with CNN, NIN adopts a micro network after each convolution layer to enhance its local modeling and abstraction ability. Besides, NIN utilizes a global average pooling instead of traditional fully connected layers before final classification, which reduces the number of model parameters significantly. In this paper, deep NIN models with multiple convolutional layers are built for encrypted traffic classification, in which the micro network after each convolutional layer is an MLP. Furthermore, a parallel decision strategy is designed, which trains two sub-networks to process packet headers and packet bodies separately. The final classification result for a data packet is determined by fusing the decisions of both sub-networks. We evaluated the performance of our proposed methods on the "ISCX VPN-nonVPN" encrypted traffic dataset [5]. Experimental results show that NIN models can achieve a better balance between classification accuracy and model complexity than conventional CNNs. The parallel decision strategy can further improve the accuracy of single NIN models for encrypted network traffic classification. The best test set F1 scores of traffic characterization and application identification are 0.983 and 0.985 respectively.

The contributions of this paper are twofold. First, NINs are employed into the task of encrypted traffic classification for the first time, which performs better than conventional CNNs and the existing Deep Packet model [21] on the "ISCX VPN-nonVPN" dataset. Second, a parallel decision strategy is designed which further improves the classification accuracy and indicates that different packet segments may provide different kinds of clues and should be processed separately for traffic classification.

The rest of the paper is organized as follows. Section II introduces the details of our proposed methods. Section

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

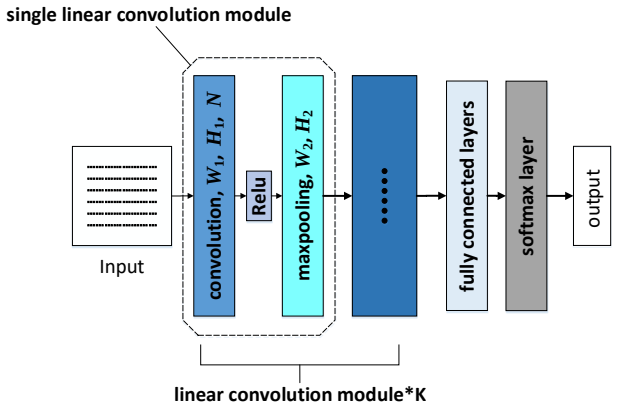

FIGURE 1: The structure of our CNN-based baseline model for encrypted traffic classification. The definitions of all symbols can be found in the last paragraph of Section II.A.

III presents the experimental results. Finally, the paper is concluded in Section IV.

II. METHODOLOGY

This paper focuses on the task of packet-level encrypted traffic classification, which means that a traffic or application type decision should be made for each input traffic packet. In this section, we will first introduce our CNN-based baseline model, and then presents our proposed NIN-based model together with the parallel decision strategy.

A. CNN-BASED ENCRYPTED TRAFFIC CLASSIFICATION

1D-CNNs (one-dimensional-CNNs) has been applied in natural language processing [26] and gained a great success. Since a traffic packet is a sequence of data bytes which is similar to a language sequence to some extent, we choose 1D-CNNs to build our CNN-based baseline model for encrypted network traffic classification. An illustration of our model's structure is presented in Figure 1. Here, the input is a sequence of packet bytes with fixed length. The output is a class label for either traffic characterization or application identification.

As shown in Figure 1, the input data is first processed by K linear convolution modules, each composed of a convolution layer and a pooling layer. Let $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ denote the input feature map of a convolution layer, where d and n are the length and channel number of $\boldsymbol{X}$. The output of the convolution layer is calculated as

$$\boldsymbol{F} = ReLU(\boldsymbol{U} * \boldsymbol{X},\ H_1), \tag{1}$$

where $ReLU(\cdot)$ is the Rectified Linear Unit activation function, $*$ stands for the operation of one-dimensional convolution and H_1 is the stride for downsampling. $\boldsymbol{U} \in \mathbb{R}^{W_1 \times n \times N}$ are the convolution filters of this layer that need to be estimated, where W_1 and N denote the kernel size and output channel number respectively. $\boldsymbol{F} \in \mathbb{R}^{d' \times N}$ is the output

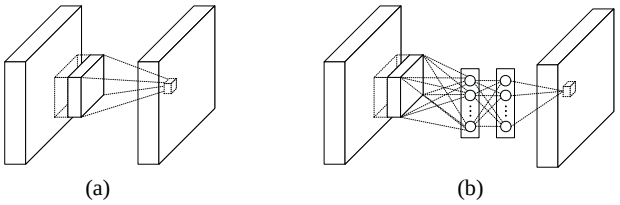

FIGURE 2: Comparison between (a) a traditional convolution in CNNs and (b) an MLP convolution in NINs [25].

of this convolution layer, and its length d' is determined by d, W_1, and H_1. To further reduce the length of $\boldsymbol{F}$, a max pooling operation is then performed in each linear convolution module as shown in Figure 1.

The output of the K linear convolution modules is a two-dimensional tensor, which is then flattened into a one-dimensional vector and fed into two fully connected layers. In the end, a softmax layer is used for class label prediction.

There are some hyper-parameters in the linear convolution modules. For all convolution layers in these modules, W_1, H_1 and N represent the kernel size, stride and filter number respectively. Similarly, for all pooling layers in these modules, W_2 and H_2 represent the kernel size and stride.

B. NIN-BASED ENCRYPTED TRAFFIC CLASSIFICATION

The Network-In-Network (NIN) structure [25] was first proposed in 2013, and has been applied to many tasks, such as image recognition [27], object detection [28], and speech recognition [29]. Comparing with traditional CNN structure, it makes two modifications.

First, NIN adopts a micro network after each convolution layer to enhance its local modeling and abstraction ability. Figure 2 shows the comparison between a traditional convolution in CNNs and an MLP convolution in NINs which adopts a multilayer perceptron (MLP) as the instantiation of the micro network. For a traditional convolution, the computation on local receptive field can be seen as a single linear operation. Then the computation output is activated by nonlinear functions, e.g., Rectified Linear Unit (ReLU). In contrast, the MLP convolution owns an MLP-based micro network and there is a nonlinear activation after each layer in the micro network. Therefore, MLP convolution enhances model's capacity of nonlinear expressiveness and allows complicated interactions across channels.

Second, the NIN structure uses a global average pooling (GAP) layer to replace the fully connected layers after convolutions in conventional CNNs. The differences between these two structures are shown in Figure 3. In traditional CNNs for classification, the feature maps calculated by the last convolutional layer are flattened into a one-dimensional vector and then sent into fully connected layers. Finally, the output of the last fully connected layer is processed by a softmax layer. However, these fully connected layers contain a large amount of parameters and are prone to cause

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

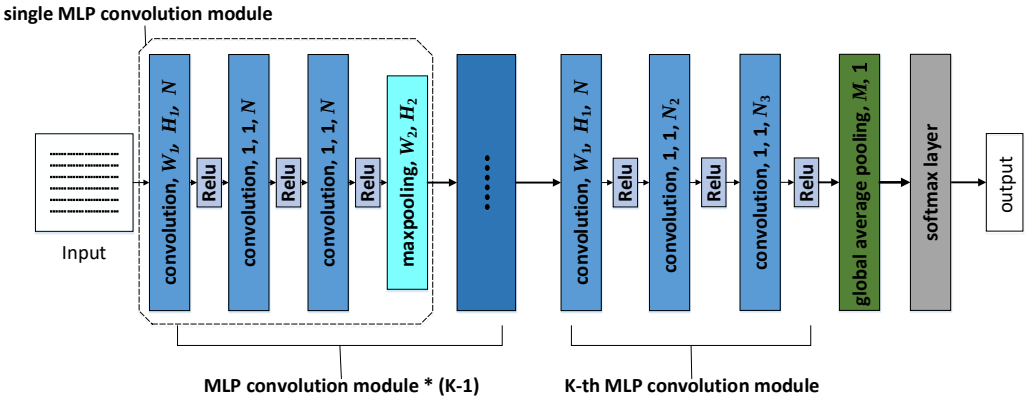

FIGURE 4: The structure of the NIN model used in our proposed method. The definitions of all symbols can be found in the last paragraph of Section II.B.

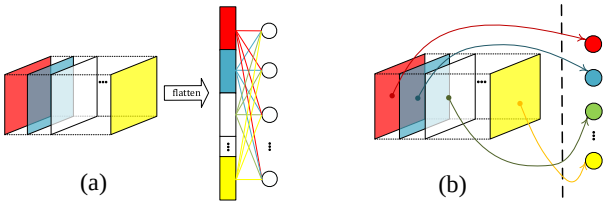

FIGURE 3: Comparison between (a) fully connected layers in CNNs and (b) the global average pooling in NINs.

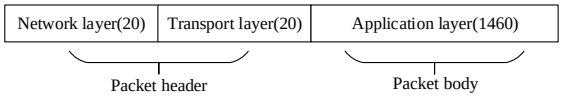

FIGURE 5: The structure of a data packet used in our implementation.

overfitting issues, resulting in unsatisfactory generalization ability of networks. In contrast, the GAP layer in NINs averages the feature maps given by the last convolution layer, and the output vector is directly fed into the softmax layer. The use of global average pooling can reduce model complexity and avoid model overfitting effectively [25].

In this paper, we introduce NINs into encrypted network traffic classification and the model structure is illustrated in Figure 4.

As shown in this figure, the NIN model contains K MLP convolution modules and the first $K-1$ ones have the same structure. Each MLP convolution module is composed of a linear convolution layer and two convolution layers with 1×1 convolution kernel. These two 1×1 convolution layers act as the MLP-based micro network mentioned above. Let $\boldsymbol{F} \in \mathbb{R}^{d'\times N}$ denote the output of the convolution layer as shown in Eq. (1). Then, the output of the micro network is calculated as

$$\boldsymbol{F'} = ReLU(ReLU(\boldsymbol{F}\cdot\boldsymbol{V}_1)\cdot\boldsymbol{V}_2), \tag{2}$$

where $\cdot$ stands for matrix multiplication and $\{\boldsymbol{V}_1, \boldsymbol{V}_2\} \in \mathbb{R}^{N\times N}$ are the parameters of the two 1×1 convolution layers, which enhances the nonlinear and cross-channel interactions in the model.

In the first $K-1$ MLP convolution modules, all convolution layers are activated by ReLU and have the same filter number N. Meanwhile, there is a max pooling layer at the end of each MLP convolution module with the kernel size W_2 and the stride H_2. The structure of the last MLP convolution module is slightly different from the previous ones. First, the three convolution layers within this module have different filter numbers (N, N_2, N_3) respectively, because a channel dimensionality reduction is necessary to meet the requirement of global average pooling. Here, N_3 is the number of traffic or application types for classification. Second, there is no max pooling in this module. The output of all K MLP convolution modules is a two-dimensional matrix with dimensions of $M\times N_3$, which refer to the size of each feature map and the number of feature maps respectively. Then for each feature map, a global average pooling with kernel size M is conducted. Finally, the resulted vector with size of $1\times N_3$ is sent to the softmax layer to complete the prediction.

C. PARALLEL DECISION USING NINS

As introduced above, this paper studies packet-level encrypted traffic classification. As shown in Figure 5, each data packet processed by neural networks is composed of three segments corresponding to the Network Layer, the Transport Layer and the Application Layer of TCP/IP model respectively. In our implementation, these three segments contain fixed 20, 20 and 1460 bytes respectively after padding or truncating during data pre-processing. The details of data pre-processing will be introduced in Section III.B. In previous deep learning based traffic classification methods, a data

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

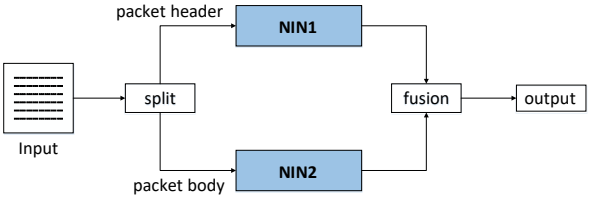

FIGURE 6: The diagram of parallel decision for traffic classification.

packet is usually treated as a whole and is sent into a single model for classification. However, considering that different packet segments may provide different kinds of clues for traffic classification, such as the port numbers at the Transport Layer and the data patterns at the Application Layer, we consider that it may be a better method if sub-networks can be build to process them separately.

Therefore, this paper designs a parallel decision strategy for neural network based traffic classification. Its diagram is shown in Figure 6. A data packet is first split into two parts, a packet header and a packet body, according to Figure 5. Then, these two parts are sent into two NIN models to calculate classification probabilities separately. At last, the two results are fused to obtain the final one. Let vectors $\boldsymbol{y}_1 = [y_{1,1}, \ldots y_{1,N_3}]^\top$ and $\boldsymbol{y}_2 = [y_{2,1}, \ldots y_{2,N_3}]^\top$ denote the softmax outputs of the two NIN models, where $y_{1,m}$ and $y_{2,m}$ stand for the probabilities that the packet should be classified as the m-the class calculated by the two NIN models. These two vectors are fused linearly as

$$\boldsymbol{y} = \alpha \boldsymbol{y}_1 + (1 - \alpha)\boldsymbol{y}_2, \tag{3}$$

where the vector $\boldsymbol{y}$ contains the final probability predicted for each category and α is a fusion weight.

At the training stage, all the data packets in the training set are first divided into headers and bodies, and then two NIN models are built accordingly. Then, we tune the weight α within the range of $(0, 1)$ and observe the performance of fused probabilities $\boldsymbol{y}$ on the validation set. The optimal α is determined when the fused prediction achieves the best performance on the validation set.

III. EXPERIMENTS

A. TASKS AND DATASETS

The "ISCX VPN-nonVPN" encrypted traffic dataset [5] was adopted in our experiments. This dataset consisted of pcap files corresponding to various network applications, and all files were captured by the authors in their daily life. When constructing the dataset, the captured packets were first discriminated according to the application category (such as Skype, Hangouts, etc.), and then classified according to the specific activities that the data packets in the application participated in (such as voice call, chat, etc.). Ultimately, the packets which belonged to a same application's activity were arranged into the same pcap file. This dataset also contained packets encrypted by virtual private networks (VPN).

TABLE 1: Sample numbers of different application and traffic types in the raw ISCX dataset.

Application	Number	Traffic	Number
AIM	5K	Chat	82K
Email	28K	Email	28K
Facebook	2502K	File Transfer	210K
FTPS	7872K	Streaming	1139K
Hangouts	3766K	P2P	70K
ICQ	7K	VoIP	5120K
Netflix	299K	VPN-Chat	50K
SFTP	418K	VPN-Email	13K
Skype	2872K	VPN-File Transfer	251K
Spotify	40K	VPN-Streaming	479K
Torrent	70K	VPN-P2P	269K
Voipbuster	842K	VPN-VoIP	753K
Vimeo	146K		
YouTube	251K		
SCP	448K		
Tor	202K		
Gmail	12K		

The specific numbers of packets belonging to different application and traffic types in the raw "ISCX VPN-nonVPN" dataset is summarized in Table 1. As shown in the left part of this table, the dataset contained a total of 17 types of applications. It is worth mentioning that the SCP, Tor and Gmail applications only had non-VPN packets, and the remaining 14 application types had both VPN and non-VPN packets. As shown in the right part of this table, the dataset contained a total of 12 traffic types, including 6 non-VPN ones and 6 VPN ones. We can also see there was a serious data imbalance problem among different application or traffic types in the raw dataset. In our experiments, two tasks were defined for application classification and traffic classification respectively. To minimize the impact of the data imbalance problem, two experimental datasets corresponding to the two tasks were constructed based on the raw "ISCX VPN-nonVPN" dataset.

The object of Task A was application classification and the application types included the 14 ones in Table 1 which contained both VPN and non-VPN packets. For each type with sufficient samples, 5000 packet samples were randomly selected from the raw dataset to construct the experimental dataset of Task A, in which the ratio between non-VPN and VPN packets was 1:1, i.e., 2500 samples for each. For the two application types with insufficient samples (AIM and ICQ), their non-VPN/VPN packet numbers were 1522/1351 and 910/2500.

Task B aimed at traffic classification, that was, to classify the packets belonging to multiple applications but with the same service type into one traffic type. The mapping relationship between application types and traffic types is shown in Table 2 and more details can be found in [5]. For each of the 12 traffic types shown in Table 1, 12000 packet samples were randomly selected from the raw dataset to construct the experimental dataset of Task B.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

TABLE 2: The mapping relationship between application types and traffic types. Each traffic type in this table represents both VPN one and non-VPN one.

Traffic Type	Application Type
(VPN-)Chat	ICQ, AIM, Skype, Facebook, Hangouts
(VPN-)Email	Email, Gmail
(VPN-)File Transfer	Skype, FTPS, SFTP
(VPN-)Streaming	Vimeo, YouTube, Netflix, Spotify
(VPN-)P2P	Torrent
(VPN-)VoIP	Facebook, Skype, Hangouts, Voipbuster

TABLE 3: Hyper-parameters of the 5 linear convolution modules in CNN_{large}. The meanings of these hyper-parameters can be found in Figure 1.

Module	1	2	3	4	5
W_1	7	5	3	3	3
H_1	3	1	1	1	1
N	64	256	384	512	1024
W_2	3	3	3	3	3
H_2	3	3	3	3	3

For each experimental dataset, we randomly divided it into a training set, a validation set and a test set according to the proportions of 68%, 16% and 16%.

B. DATA-PREPROCESSING

All packet samples in the experimental datasets for both tasks needed to be pre-processed since the raw packets with variable lengths cannot be fed into neural networks directly. In this paper, we mainly followed the previous work [21] for data preprocessing, which included four steps, packet parsing, packet length unification, byte normalization and packet labeling.

Raw packet were first parsed by removing Ethernet Layer header, anonymizing IP addresses, unifying the length of Transport Layer header, and discarding the packets without Application Layer data. Here, unifying the length of Transport Layer header aimed to make the header lengths of all packets the same, and was implemented by truncating the TCP header to 20 bytes and zero-padding the UDP header to 20 bytes. The packet length unification step generated packets with fixed length by truncating or zero-padding the Application Layer data. In our implementation, the fixed length was set as 1500 bytes because the length of most data packets was less than or equal to 1500 bytes. Thus, the length of Application Layer data was 1460 bytes as shown in Figure 5. Then, each element, i.e., each byte, in data packets was divided by its maximum value 255 and normalized to the interval of [0, 1] in order to facilitate the training of neural networks. Finally, each packet was marked with an application type label and a traffic type label so that neural networks can be trained in a supervised way.

TABLE 4: Hyper-parameters of the 5 MLP convolution modules in NIN_{large}. The meanings of these hyper-parameters can be found in Figure 4.

Module	1	2	3	4	5
W_1	7	5	3	3	3
H_1	3	1	1	1	1
N	64	256	384	512	-
W_2	3	3	3	3	-
H_2	3	3	3	3	-

TABLE 5: Model complexities of built neural networks.

Model	Parameters	FLOPs
CNN_{large}	5.755×10^6	1.119×10^8
CNN_{small}	6.267×10^5	1.593×10^7
NIN_{large}	4.034×10^6	2.177×10^8
NIN_{small}	1.099×10^6	3.468×10^7
$CNN_{DeepPacket}$	1.218×10^7	1.095×10^8

C. MODEL CONSTRUCTION

For each task, two baseline CNN models, CNN_{large} and CNN_{small}, were first built following the introductions in Section II.A. In both models, the number of linear convolution modules was $K = 5$ and the neuron numbers of the two fully connected layers were 1024 and 64. Table 3 shows the hyper-parameters of the 5 linear convolution modules in CNN_{large}. The model structure of CNN_{small} were the same as CNN_{large} except that the filter numbers N of its 5 linear convolution modules were 32, 64, 128, 256 and 512, which were much fewer than those of CNN_{large}.

Similarly, for each task, two NIN models without parallel decision, NIN_{large} and NIN_{small}, were also built following the introductions in Section II.B. In both models, the number of MLP convolution modules was $K = 5$. Table 4 shows the hyper-parameters of the 5 MLP convolution modules in NIN_{large}. Most of them were the same as the ones in CNN_{large}. In the last MLP convolution module, the filter number N_2 were set as 512. The model structure of NIN_{small} were the same as NIN_{large} except that the filter numbers N of its 5 MLP convolution modules were 32, 64, 128, 256 and 512, and the filter number N_2 of the last MLP convolution module was 256, which were much fewer than those of NIN_{large}.

Considering that our data pre-processing procedure was basically consistent with the study of M. Lotfollah et al. [21], we chose to compare our proposed models with the state-of-the-art model in their paper [21] and named it $CNN_{DeepPacket}$. It consisted of two linear convolution layers and three fully connected layers. The filter number and the stride of both convolution layers were 200 and 5, and the kernel sizes were 5 and 4 for two layers respectively. The neuron numbers of three fully connected layers were 1024, 512 and 64.

The model complexities of above five models in terms of parameter numbers and floating point operations (FLOPs) per packet are summarized in Table 5. We can see that all

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

TABLE 6: Overall experimental results of five neural networks on application classification (Task A) and traffic classification (Task B).

	NIN_{large}			CNN_{large}			NIN_{small}			CNN_{small}			$CNN_{DeepPacket}$		
	P_r	R_c	F_1	P_r	R_c	F_1	P_r	R_c	F_1	P_r	R_c	F_1	P_r	R_c	F_1
Task A	0.975	0.974	0.974	0.970	0.971	0.970	0.969	0.969	0.969	0.962	0.962	0.961	0.965	0.962	0.965
Task B	0.979	0.979	0.979	0.973	0.973	0.973	0.975	0.974	0.974	0.969	0.969	0.969	0.973	0.973	0.972

four proposed models had fewer parameter numbers than $CNN_{DeepPacket}$. NIN_{large} had fewer parameter numbers but higher computation costs than CNN_{large}. The reason is that the global average pooling in NINs reduced the parameter of the fully connected layers in CNNs while the micro networks in NINs increased the computation costs of CNNs. Anyway, the complexity of NIN_{small} was much lower than that of CNN_{large} in terms of both parameter numbers and FLOPs.

Finally, an NIN-based classification model with parallel decision, named $NIN_{parallel}$, was built for each task following the introductions in Section II.C. In this model, the NIN2 network in Figure 6 adopted the same hyper-parameters as NIN_{large} introduced above. The NIN1 network in Figure 6 consisted of two MLP convolution modules which were both configured as $W_1 = 3$, $H_1 = 2$, $N = 200$, $W_2 = 3$ and $H_2 = 2$. Besides, its first MLP convolution module had no max pooling layer. Instead, the tensor was flattened and fed into two fully connected layers with 1024 and 64 neurons. The fusion weight α was tuned according to the model performance on validation set and its optimal value was 0.52 for both tasks.

Our models were implemented with TensorFlow and run on a server with Ubuntu 14.04.3. An Nvidia Tesla K40m GPU card was used to accelerate our training and testing. All neural networks were trained for 300 epochs with a categorical cross entropy loss using the Adam optimizer. The learning rate was set to 0.001 at the beginning, and decreased by 98% after every epoch. In order to prevent overfitting, we used the dropout technique with a ratio of 0.9.

D. EVALUATION RESULTS

1) Evaluation metrics

In our experiments, Precision (P_r), Recall (R_c) and F_1 score (F_1) were used to measure model performance. These metrics are mathematically explained as

$$R_c = \frac{TP}{TP + FN}, \tag{4}$$

$$P_r = \frac{TP}{TP + FP}, \tag{5}$$

$$F_1 = \frac{2R_cP_r}{R_c + P_r}, \tag{6}$$

where TP, FP, and FN stand for the numbers of true positive, false positive and false negative samples respectively.

2) Comparison between CNN-based and NIN-based encrypted traffic classification and related work

Table 6 show the test set performance of NIN_{large}, CNN_{large}, NIN_{small}, CNN_{small} and $CNN_{DeepPacket}$ models on application classification and traffic classification tasks respectively. Here, the P_r, R_c and F_1 values were averaged across all classes. Since F_1 score is the harmonic mean of Precision and Recall, we chose to use the average F_1 score as the main metric for analyzing the performance of different models. From these results, we can see that our proposed CNN models achieved similar performance with $CNN_{DeepPacket}$. While, our proposed NIN models obtained better results than $CNN_{DeepPacket}$ although the NIN_{small} model had much lower model complexities than $CNN_{DeepPacket}$ as shown in Table 5. Then, we will focus on the comparison between our proposed NIN and CNN models. Two main conclusions can be drawn as follows.

First, when CNN and NIN models had similar model complexity, the NIN model achieved better results of packet classification than the CNN model. We can see that NIN_{large} outperformed CNN_{large} in terms of average F_1 score on both tasks (0.974 vs. 0.970 on Task A and 0.979 vs. 0.973 on Task B). NIN_{small} outperformed CNN_{small} on both tasks as well (0.969 vs. 0.961 on Task A and 0.974 vs. 0.969 on Task B). It is worth mentioning that the basic model structures of the two NIN models were the same as their CNN counterparts as mentioned in Section III.C. That is to say, the unique model structures of NINs (i.e., MLP convolution module and global average pooling) contributed to obtaining the better performance of encrypted traffic classification than CNNs.

Second, comparing the average F_1 scores of NIN_{small} and CNN_{large}, it is obvious that although the model complexity of NIN_{small} was much lower than that of CNN_{large}, they achieved similar classification performance (0.970 vs. 0.969 on Task A and 0.973 vs. 0.974 on Task B). These results demonstrate that NIN models can achieve a better balance between traffic classification accuracy and model complexity than conventional CNNs.

Furthermore, in order to compare the performance of different models in detail, Figure 7 and 8 show the class-specific F_1 scores of four models on both tasks. From these two figures, some findings can be observed as follows.

First, the application types AIM and ICQ achieved the lowest F_1 score on task A. The best F_1 scores of the remaining categories were all higher than 0.94. This is due to the insufficient samples of these two application types in the raw ISCX dataset. As introduced in Section III.A, these two

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

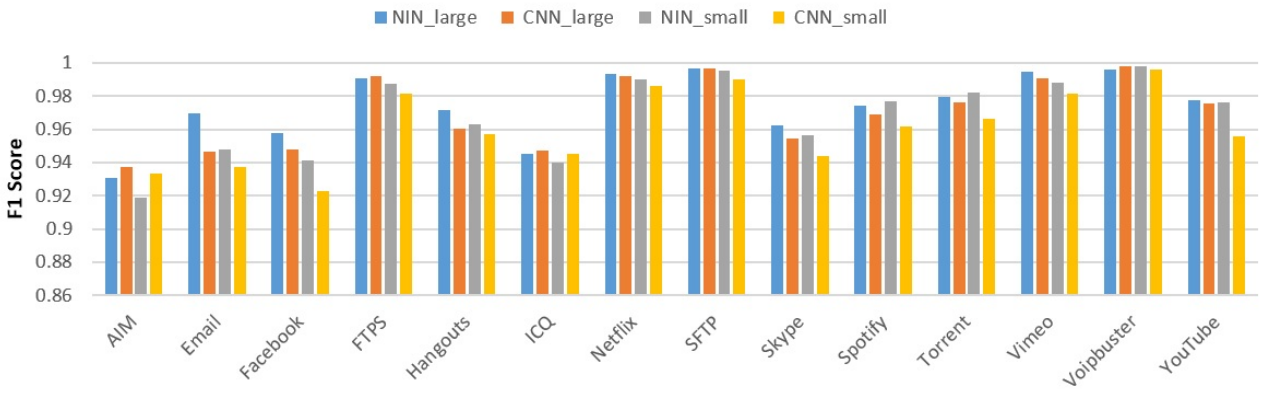

FIGURE 7: The class-specific F_1 scores of four models on application classification (Task A).

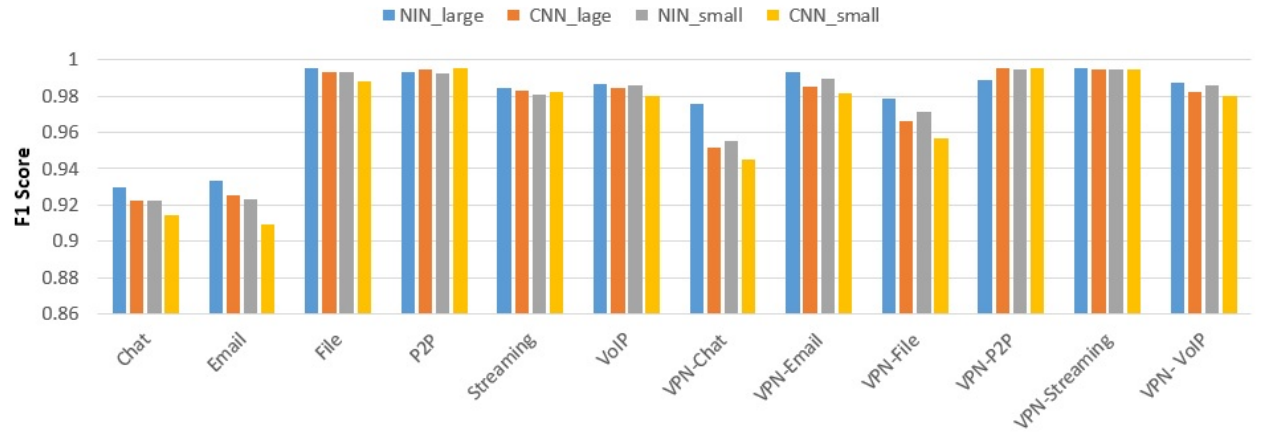

FIGURE 8: The class-specific F_1 scores of four models on traffic classification (Task B).

TABLE 7: Average F_1 scores and complexities of different models on application classification (Task A).

Model	Input Data	Input Length	Validation Set F1	Test Set F1	Parameters	FLOPs
NIN1 in $NIN_{parappel}$	Packet header	40	0.983	0.985	2.033×10^5	9.431×10^6
NIN2 in $NIN_{parappel}$	Packet body	1460	0.591	0.585	4.034×10^6	2.144×10^8
NIN_{large}	Whole packet	1500	0.973	0.974	4.034×10^6	2.177×10^8
$NIN_{parappel}$	Whole packet	1500	0.983	0.985	4.237×10^6	2.238×10^8

TABLE 8: Average F_1 scores and complexities of different models on traffic classification (Task B).

Model	Input Data	Input Length	Validation Set F1	Test Set F	Parameters	FLOPs
NIN1 in $NIN_{parappel}$	Packet header	40	0.983	0.982	2.033×10^5	9.431×10^6
NIN2 in $NIN_{parappel}$	Packet body	1460	0.681	0.673	4.034×10^6	2.144×10^8
NIN_{large}	Whole packet	1500	0.980	0.979	4.034×10^6	2.177×10^8
$NIN_{parappel}$	Whole packet	1500	0.984	0.983	4.237×10^6	2.238×10^8

types had only 2873 and 3410 samples respectively in our experimental dataset, while this number was 5000 for other application types. On task B, the traffic types Chat and Email obtained the lowest F_1 score. We consider that there were two reasons. a) There were five applications corresponding to the traffic type Chat as shown in Table 2, which increased the difficult of distinguishing the traffic type Chat from others. b) The application type Email included in the traffic type Email also had a relatively low accuracy on task A as shown in Figure 7. It may be attributed to that the data itself did not contain strong patterns for classification that can be captured by neural networks.

Second, NIN_{large} and NIN_{small} models achieved better performance than their CNN counterparts on most application and traffic categories. As shown in Figure 7, NIN_{large} outperformed CNN_{large} and meanwhile NIN_{small} out-

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

performed CNN_{small} for 10 out of 14 application categories. Similarly, as shown in Figure 8, NIN_{large} outperformed CNN_{large} and meanwhile NIN_{small} outperformed CNN_{small} for 8 out of 12 traffic types. Besides, the performances of NIN_{small} and CNN_{large} models were relatively close on most categories as well. These conclusions are consistent with the ones drawn from Table 6.

Third, we can see from Figure 8 that the F_1 scores of VPN traffic types were comparable with those of non-VPN ones. This demonstrates the robustness of deep learning based traffic classification methods that can handle the data packets encrypted by VPN effectively.

3) Performance of parallel decision using NINs

Table 7 and 8 show the average F_1 scores of $NIN_{parallel}$ models on both tasks. In addition, the results of using the NIN1 or NIN2 network in $NIN_{parallel}$ and the results of the NIN_{large} model introduced above are also listed for comparison.

Comparing the performance of NIN1 and NIN2 in both tables, it can be found that using only packet header can obtain significantly better performance than using only packet body for traffic classification. That is to say, for the packet-level traffic classification task, the 40-byte packet header contains more information useful for classification than the 1460-byte packet body. Comparing the performance of NIN1 and NIN_{large}, we can see that using only packet header was better than using the whole packet, although the former employed a much smaller model structure than the latter. This confirms our assumption that packet header and packet body contain different kinds of information for classification and it is difficult to model the concatenation of them using a single network.

In each $NIN_{parallel}$ model, NIN1 contained much fewer parameters than NIN2. Thus, the model complexity of $NIN_{parallel}$ was comparable with that of NIN_{large} as shown in Table 7 and 8. Furthermore, $NIN_{parallel}$ achieved better F_1 scores than NIN_{large} on both tasks, which demonstrates the effectiveness of our proposed parallel decision strategy. In addition, we counted the number of validation set samples that were correctly classified by one of NIN1 and NIN2 but failed to be correctly classified by the other. For task A, there were 4266 samples that were correctly classified by NIN1 but misclassified by NIN2, and there were only 38 samples that were correctly classified by NIN2 but misclassified by NIN1. For task B, these two numbers were 7429 and 169 respectively. Thus, we can clearly see that the performance of $NIN_{parallel}$ was actually dominated by its NIN1 part. In other words, the task of classifying encrypted packets is still challenging if only Application Layer data is available.

IV. CONCLUSION

This paper has presented a method of building deep and parallel network-in-network (NIN) models for encrypted network traffic classification. This method aims at mapping fixed-length data packets towards the labels of application or traffic categories. Based on deep NIN networks with multiple MLP convolutional modules, a parallel decision strategy is designed which builds two sub-networks for processing packet header and packet body separately. Experimental results on the "ISCX VPN-nonVPN" encrypted traffic dataset show that NIN models achieved better performance than CNNs. Besides, the parallel decision strategy further improved the accuracy of using single NIN model for traffic classification. To boost the performance of encrypted traffic classification using only Application Layer data will be a task of our future work.

REFERENCES

[1] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," IEEE network, vol. 26, no. 1, pp. 35–40, 2012.

[2] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in International Workshop on Passive and Active Network Measurement. Springer, 2005, pp. 41–54.

[3] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1135–1156, 2013.

[4] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, and M. E. Karsligil, "Application identification via network traffic classification," in 2017 International Conference on Computing, Networking and Communications (ICNC). IEEE, 2017, pp. 843–848.

[5] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related," in Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), 2016, pp. 407–414.

[6] R. Yuan, Z. Li, X. Guan, and L. Xu, "An svm-based machine learning method for accurate internet traffic classification," Information Systems Frontiers, vol. 12, no. 2, pp. 149–156, 2010.

[7] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural Computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[8] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, no. 5786, pp. 504–507, 2006.

[9] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in NIPS, 2012, pp. 1097–1105.

[10] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Process. Magazine, vol. 29, no. 6, pp. 82–97, 2012.

[11] L. Deng and Y. Liu, Deep learning in natural language processing. Springer, 2018.

[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," IEEE Transactions on Network and Service Management, vol. 16, no. 2, pp. 445–458, 2019.

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

[14] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," IEEE Access, vol. 7, pp. 54 024–54 033, 2019.

[15] Z. Wang, "The applications of deep learning on traffic identification," BlackHat USA, vol. 24, no. 11, pp. 1–10, 2015.

[16] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in 2017 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, 2017, pp. 43–48.

[17] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, "Automatic multi-task learning system for abnormal network traffic detection," International Journal of Emerging Technologies in Learning (iJET), vol. 13, no. 04, pp. 4–20, 2018.

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3010637, IEEE Access

[19] Y.-C. Chen, Y.-J. Li, A. Tseng, and T. Lin, "Deep learning for malicious flow detection," in 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC). IEEE, 2017, pp. 1–7.

[20] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "Deep-full-range: A deep learning based network encrypted traffic classification and intrusion detection framework," IEEE Access, vol. 7, pp. 45 182–45 190, 2019.

[21] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," Soft Computing, vol. 24, no. 3, pp. 1999–2012, 2018.

[22] 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. 55 380–55 391, 2018.

[23] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using cgan," arXiv preprint arXiv:1911.12046, 2019.

[24] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," Computer Networks, vol. 165, p. 106944, 2019.

[25] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv preprint arXiv:1312.4400, 2013.

[26] C. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts," in Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, 2014, pp. 69–78.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[28] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 3367–3375.

[29] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2017, pp. 4845–4849.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.