Encrypted Traffic Identification by Fusing Softmax Classifier with Its Angular Margin Variant

Lin YAN\textsuperscript{1}, Mingyong ZENG\textsuperscript{1,†a,}, Shuai REN\textsuperscript{1,}, Nonmembers, and Zhangkai LUO\textsuperscript{1†,}, Student Member

SUMMARY Encrypted traffic identification is to predict traffic types of encrypted traffic. A deep residual convolution network is proposed for this task. The Softmax classifier is fused with its angular variant, which sets an angular margin to achieve better discrimination. The proposed method improves representation learning and reaches excellent results on the public dataset.

key words: traffic identification, convolution neural network, residual network, softmax classifier

1. Introduction

Encrypted traffic identification has drawn increasing attention due to the widespread use of encryption techniques. It classifies unknown traffic to certain application types such as Chatting, Browsing, File transfer and Voice call. Identifying the traffic type is of vital significance to network security, surveillance and management. It is usually the first step towards later analysis, such as network planning, traffic flow control, malware detection and QoS (Quality of Service).

Traffic identification has been studied with four solutions [1]: The first two solutions are based on port and DPI (Deep Packet Inspection). They rely heavily on fixed well-known ports in the packet headers or certain signatures within the packet payloads. Therefore, they are not reliable for inferring encrypted traffic. The latter two are based on machine learning (ML) and can be further divided to traditional ML methods and deep learning (DL) methods. The difference lies in that traditional ML methods need to be provided with hand-crafted features and DL methods can learn both features and classifiers in an end-to-end manner. Both methods can deal with encrypted traffic and are current research focuses. The frequently extracted features for traditional ML methods include total packets, packet lengths and inter-arrival time in a flow [2]. A flow is determined by a 5-tuple (source IP and port, destination IP and port, transportation-layer protocol).

Our work belongs to deep learning methods which require no feature engineering. Many papers regard network traffic as images and apply CNNs (Convolution Neural Networks) to learn features and classifiers [1]. However, Wang et. al. found that one-dimensional CNN (1DCNN) performed better than 2DCNN in traffic classification [3]. They claimed the packets of a flow consist of 1D (1-dimensional) sequence of packet bytes rather than 2D images. Lotfollahi et. al. proposed a Deep-Packet method with CNN and SAE (Stacked Auto-Encoder) for encrypted application identification [4]. Both methods have achieved promising results. Besides, multiple LSTM models are utilized for short-term traffic flow prediction in [5].

The drawbacks in above DL methods are that their representation learning abilities in traffic identification are not fully explored. In this letter, we propose to tackle the traffic representation learning from two aspects. The first one is to adopt a deep residual CNN network as the backbone in replace of shallow networks in [3] and [4]. The second one is to adapt Softmax classifier with its angular variant and fuse them for joint learning. On the ISCX VPN-NonVPN dataset [2], the proposed method achieves much better results in traffic identification than previous methods. It reveals that deeper backbone network and the fused classifier can lead to better traffic identification performance.

2. Deep Residual CNN Backbone Network

The CNNs in [3] and [4] have only two convolution (Conv) layers for traffic identification. To investigate the performance gains brought by deeper network, we propose a deep residual network to replace shallow CNNs. The residual network (ResNet) has been extensively applied as an outstanding network in image classification and many related tasks [6]. As in 1DCNN [3], we regard the traffic bytes in a 5-tuple flow as 1D byte sequence rather than 2D image. Then we adapt the image based 2D ResNet in [6] to 1D ResNet for our sequence task. The input to our 1D ResNet is a normalized 784-byte sequence, which is extracted by truncating or zero-padding a traffic flow as in [3]. Each input byte is normalized to [0,1] by dividing 255.

The proposed 1D ResNet is shown in Fig. 1. Its backbone contains three building stages of several residual blocks with varying block parameters. The number of bottleneck filters in ResNet [6] is denoted \( n \) and the number of repeated blocks is denoted \( B \). We set \( n = 64, 128, 256 \) and \( B = 2, 3, 3 \) for the three stages. Within blocks in each stage,
the input and output sequence lengths remain the same except for the transition blocks. The two transition blocks are the first blocks in Stage-2 and Stage-3, where their input lengths are reduced to one forth for output, i.e. 784 → 196 and 196 → 49. The bottleneck residual version is adopted (refer to [6] for details). The bottleneck filters $n$ are doubled in adjacent stages, i.e. 64 → 128 → 256.

The detailed residual block is plotted in the right part of Fig. 1. For a multi-channel sequence input $x = (x^1, \ldots, x^n) \in \mathbb{R}^{dcn}$, $d$ denotes the input sequence length and $m$ is the number of input channels (i.e. #output filters of the previous layer). The $3 \times 3$ Conv in [6] is adapted to $9 \times 1$ Conv for our sequence task. Batch Normalization (BN) and ReLU activation are also used as in [6]. The first two Conv layers of the residual branch have $n$ filters (i.e. bottleneck) while the third Conv has $4n$ filters. $m$ equals to $4n$ within the same stage. The right-side $1 \times 1$ Conv is optional, which is inserted when $m \neq 4n$. For transition blocks (yellow color), the first two $1 \times 1$ Conv layers both adopt stride $s = 4$ to downsize the input length to its one fourth. The output is $x^{(1)} = (x^{(1)}_1, \ldots, x^{(1)}_{49})$ for transition blocks and $x^{(1)} = (x^{(1)}_1, \ldots, x^{(1)}_{2048})$ for other blocks.

On top of the residual backbone are a GAP (Global Average Pooling) layer and a Fc (Fully Connected) layer. The GAP layer pools the $49 \times 1024$ tensor to a 1024-dimensional vector by calculating average value from 49 responses for each channel. The Fc layer is used to reduce the feature dimension and is feed to our proposed classifier in the next section. The proposed 1D ResNet in Fig. 1 has over 20 Conv layers and has abundant network capacity for representation learning.

### 3. Softmax Classifier Fusing Angular Margin

#### 3.1 The Softmax Classifier

Instead of directly adopting Softmax classifier, we fuse it with its angular variant to construct a better classifier. The fused classifier is illustrated in Fig. 2.

For simplicity, we consider only one training sample since multiple samples can be easily extended. The 256-D Fc vector in Fig. 1 is again denoted $x \in \mathbb{R}^d$ as the embedding feature for an input sequence. A linear projection layer with weight matrix $W_s \in \mathbb{R}^{dcn}$ is used to project $x$ to $c$ class logits. We set $d = 256$ and $c = 12$ in our experiments. The bias terms are omitted and the unnormalized logit vector is written as $h_s = W_s^T x$. Then it is normalized by a Softmax layer to $[0,1]$ as:

$$
\hat{y}_s = e^h_s / \sum_{j=1}^c e^{W_s^j x}
$$

where $h_{s,j}$ is the $j$-th logit in $h_s$ and $W_{s,j}$ is the $j$-th column vector in $W_s$. $\hat{y}_s$ is the normalized class probability vector. The cross entropy loss can be calculated with prediction vector $\hat{y}_s$ and one-hot label vector $y$. Suppose $y_t = 1$, the loss is $L_s = -\log \hat{y}_{s,t}$, where $\hat{y}_{s,t}$ is the $t$-th probability in $\hat{y}_s$ for predicting the true label.

Based on findings in ArcFace [7], Softmax can be reformulated to an angular version. The vector $x$ is $l_2$ normalized to have unit size, $x' = x / \|x\|$. The projection layer has $h_a = W_a^T x'$ and each column vector $W_{a,j}$ is learned to also have unit length, $\|W_{a,j}\| = 1$. Then $h_{a,j} = W_{a,j}^T x' = \|W_{a,j}\| \times \|x'\| \cos \theta_j = \cos \theta_j$. $\theta_j$ is the angle between $x'$ and $W_{a,j}$. It can be computed as $\theta_j = \arccos h_{a,j}$. An angular margin $\Delta \theta$ is added to the angle $\theta_j$ for true label position. A scale $s$ is multiplied to form $h'_a$.

For $j \neq t$, $h'_{a,j} = s \times h_{a,j} = s \cos \theta_j$. For $j = t$, it is written as:

$$
h'_{a,t} = s \cos \theta_t + \Delta \theta = s \cos (\arccos h_{a,t} + \Delta \theta)
$$

Then $h'_a$ is normalized by a Softmax layer to get the prediction $\hat{y}_a$ as in Eq. (1). As validated in ArcFace, the angular margin based classifier forces the network to learn more discriminative features and it can better deal with ambiguous class boundaries. Similarly, the cross entropy loss with angular margin variant is thus $L_a = -\log \hat{y}_{a,t}$, where $\hat{y}_{a,t}$ is the $t$-th probability in $\hat{y}_a$ for the true label. The fused cross entropy loss is the summation of above two losses. The total loss for a batch of samples is averaged on all samples’s losses.

$$
L = L_s + L_a = -\log \hat{y}_{s,t} - \log \hat{y}_{a,t}
$$
4. Experiments

4.1 Experimental Conditions

In this section, the proposed method is compared with other existing methods on a public dataset. Since our method is built on 1D residual network and the adapted Softmax, we denote 1D residual network as 1DResnet and denote Angular Margin based Softmax as AMS for convenience. Our method is denoted ResAMS to reflect both components. In order to validate our designing choices, we replace AMS with conventional Softmax and denote this variant method as 1D-ResNet. Similarly, we replace 1DResnet with 1DCNN and denote this variant method as 1DCNN+AMS. The other compared methods are 1DCNN and 2DCNN [3].

As in 1DCNN, we adopt the same experimental setting on the public ISCX VPN-NonVPN dataset [2]. This dataset contains both VPN encrypted traffic and regularly encrypted traffic (Non-VPN). Both VPN and Non-VPN traffic contains 6 major types, i.e. Chat, Email, File Transfer, P2P, Streaming and VoIP, resulting in 12 traffic classes. The authors in 1DCNN experimented with different inputs, such as bytes from bi-directional sessions or uni-directional flows, bytes from all network layers or only from the application layer. Among these settings, we choose the hardest one, which means the bytes of a flow are extracted and concatenated from all packets’ application layer in this flow.

The bytes of a flow are truncated or zero-padded to be a 784-byte vector. All 784 bytes are normalized to [0,1] by dividing 255 and are feed to our network. The exact training and testing splits in 1DCNN are adopted. Total 38723 traffic flow samples are extracted from raw packets of this dataset. We use 10% (i.e. 3872) flows for testing and the rest 90% for training. The task of each method is to classify a flow to be one of the 12 traffic classes, solely based on its application data without packet header information.

All the experiments are conducted on a server with 40 CPUs (Intel Xeon E5-2640 V4@2.4GHZ, total 256GB host memory) and 8 GPUs (Nvidia Tesla P100, 16GB memory each GPU). Each model is trained on a single GPU until convergence. 2DCNN and 1DCNN are trained with SGD optimizer for 40000 steps by using public Tensorflow code from [3]. Our proposed method is implemented with Keras package and Tensorflow backend. The Adadelta optimizer is used with learning rate 1.0 and batch size 64. We set $\Delta\theta = \Delta\theta = 10^2 = 10 \times 2\pi/360$ after fine-tuning the margin $\Delta\theta$ in the fused classifier. In testing, prediction probabilities from the fused two classifiers are summed to give a final prediction.

4.2 Experimental Results

We compare the performance metrics and computation costs for all the compared methods. For performance, the results are listed in Table 1. The evaluation metrics include overall accuracy (Acc), mean precision, mean recall and mean F1 score. The overall accuracy reflects the percentage of correct predictions on all 3872 testing flows. In evaluation, the precision, recall and F1 score can be computed for each of 12 classes. By averaging over all 12 classes, we can compute the mean precision, mean recall and mean F1 score.

From Table 1, it is evident that the proposed ResAMS performs best on all four metrics. ResAMS achieves 5.2% higher accuracy and 9% higher F1 score than 1DCNN. Take 1DCNN and 1D-ResNet for comparison, 1DCNN achieves 4.1% lower accuracy and 8.2% lower F1 score than the deeper 1D-ResNet. It means network depth and capacity matter a lot for traffic identification. Besides, the proposed 1D residual structure can facilitate better network learning than plain CNN structure. When applying the adapted AMS classifier, the F1 score of 1DCNN+AMS improves from 76.7% of 1DCNN to 81.5%. Both deep residual network and the fused classifier of ResAMS can help to improve its performance.

The computation time of all the compared methods on the same server are listed in Table 2. The second to fifth columns present the total training time, the total testing time on 3872 testing flows, the average time per testing flow and the estimated flows per second for each method.

From Table 2, ResAMS costs about 3 times longer training time and 2 times longer testing time than 1DCNN. This is not surprising since ResAMS has over 20 Conv layers while 1DCNN has only 2 Conv layers. 1DCNN and ResAMS can classify about 2564 and 893 flows in every second. If a flow contains 10MB data on average, then ResAMS can support about 70Gbps link (i.e. $893 \times 10 \div 1024 \times 8 = 69.8$). It means the inference speed of ResAMS is often sufficient for most practical scenarios. ResAMS improves F1 score from 76.7% of 1DCNN to 85.7%, so its higher performance makes it a better choice than 1DCNN under such cases. However, for scenarios with over 100Gbps link, we recommend the 1DCNN+AMS variant since it has higher performance and comparable speed compared with 1DCNN. Though ResAMS focus on improving classification performance, we stress that all these compared methods are efficient because they extract at most 784 bytes from each flow.

| Methods   | training | testing | time/flow | flows/s |
|-----------|----------|---------|-----------|---------|
| 2DCNN     | 5.8min   | 1.62s   | 0.42ms    | 2381    |
| 1DCNN     | 5.1min   | 1.51s   | 0.39ms    | 2564    |
| 1DCNN+AMS | 5.5min   | 1.56s   | 0.40ms    | 2500    |
| 1D-ResNet | 19.6min  | 4.25s   | 1.10ms    | 909     |
| ResAMS    | 20.2min  | 4.33s   | 1.12ms    | 893     |

Table 1: Performance metrics of the compared methods.

| Methods   | Acc  | Precision | Recall | F1   |
|-----------|------|-----------|--------|------|
| 2DCNN     | 80.3%| 78.1%     | 75.2%  | 75.8%|
| 1DCNN     | 81.0%| 79.8%     | 75.4%  | 76.7%|
| 1DCNN+AMS | 82.3%| 81.8%     | 81.2%  | 81.5%|
| 1D-ResNet | 85.1%| 85.6%     | 84.7%  | 84.9%|
| ResAMS    | 86.2%| 86.1%     | 85.2%  | 85.7%|

Table 2: Computation time of the compared methods.
5. Conclusion

A deep residual network with fused classifier is proposed for encrypted traffic identification. Compared with shallow networks, the deep layers and large capacity can facilitate better representation learning. The angular margin based classifier is fused to better train the deep network. The proposed method has outperformed other methods greatly. Next, we will apply the proposed methods to more related security tasks.

References

[1] S. Rezaei and X. Liu, “Deep learning for encrypted traffic classification: an overview,” IEEE Commun. Mag., vol.57, no.5, pp.76–81, May 2019.

[2] G. Draper-Gil, A.H. Lashkari, M.S.I. Mohammad, and A.A. Ghorbani, “Characterization of encrypted and VPN traffic using time-related features,” 2nd Int. Conf. Information Systems Security and Privacy (ICISSP), pp.407–414, 2016.

[3] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” IEEE Int. Conf. Intelligence and Security Informatics (ISI), pp.43–48, 2017.

[4] 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, 2020.

[5] Z. Xue and Y. Xue, “Multi long-short term memory models for short term traffic flow prediction,” IEICE Trans. Inf. & Syst., vol.E101-D, no.12, pp.3272–3275, Dec. 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.770–778, 2016.

[7] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: additive angular margin loss for deep face recognition,” IEEE Conf. Comput. Vis. Pattern Recognit., pp.4690–4699, 2019.