Deep Learning-Based Encrypted Traffic Classification: Novel Use of the Headers

Chun-Hua Lin **Advisor: Raylin Tso**

National Chengchi University

Introduction

Network traffic identification provides advantages on advanced network management such as anomaly detection, Quality-of-Service (QoS), etc. The usage of encryption schemes, e.g., TLS, VPN, has made such tasks difficult. We propose a deep learning-based traffic classification by obtaining the unencrypted data of individual packets, using a convolutional neural network (CNN) to extract the spatial features and identify the traffic type. Our approach achieved F_1 score of 0.95 with the UNB ISCX VPN-nonVPN dataset and can identify packets that were not in the training phase with a high success rate.

Keywords: Traffic classification, Convolutional Neural Networks

Methodology

1. Remove the L2 header and the payload of L4.
2. Pad each packet to 100 bytes and normalize them to be in range [0, 1].
3. Perform sampling to ensure the number of classes is fairly balanced.
4. Feed the preprocessed tensor into our CNN for classification.

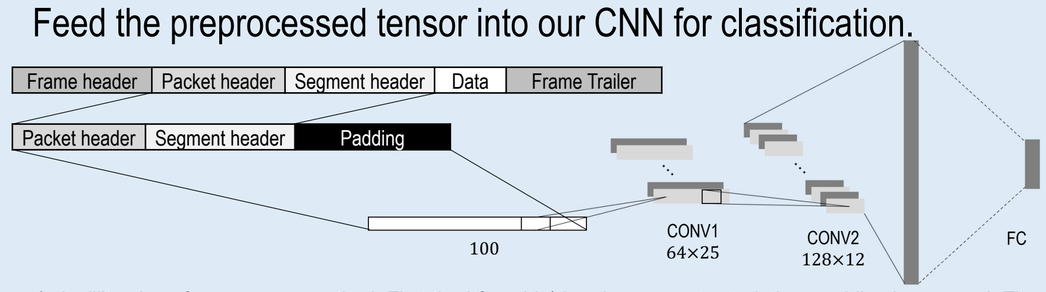

Figure 1. An illitrusion of our propose method. First the L3 and L4 headers are extracted, then padding is appened. Then the data is fed into our CNN with layer CONV1, CONV2 and FC.

Result

We implemented our CNN using Pytorch on Google Colab with Tesla P100-PCIe-16GB as our allocated GPU. Our model is trained and tested with two independent datasets extracted from the full dataset: 80% for training and 20% for testing. As shown in Figure 1., the CONV1 layer contains 64 kernel maps with size 4 and stride 4. This is because most features in the headers are multiples of 4. Then the CONV2 layer has 128 kernel maps with size 2 and stride 2 to further extract patterns of the feature maps. Finally, the feature maps are flattened into one dimension and fed into the FC layer consisting of 5 layers. We have hyperparameter: batch size 128, epochs 50, an initial learning rate of 0.001, and used Adam optimizer and Cross entropy as the loss function for training.

To evaluate the performance of our proposed model, we used Recall ($Rc = \frac{TP}{TP+FP}$), Precision ($Pr = \frac{TP}{TP+FN}$), and F_1 ($F_1 = \frac{2 \cdot Rc \cdot Pr}{Rc+Pr}$) Score metrics. TP, FP and TN respectively stands for True Positive, False Positive and False Negative. As shown in Figure 3., Convergence is attained after 5-10 epochs, and the trained model achieved an accuracy (precision) of 0.95, indicating our model was able to extract features from the testing set and distinguish each traffic. However, according to Figure 4. and Table 1., we see a slightly low performance with Chat and Email, we suspect this is due to the characteristic of two traffic has a certain level of similarity. Overall, the model was able to correctly identify each traffic.

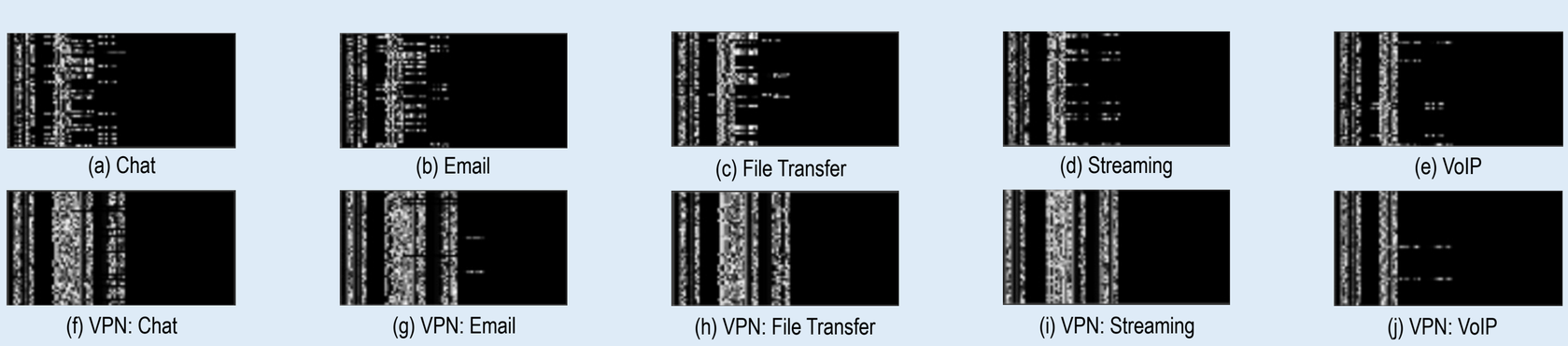

Figure 2. Visualization of the the ten traffic classes. Each image has size 50×100, representing 50 processed packets of size 100 bytes. The packets are appened for easier comprehension. White pixels represent any value between 1 and 255, while black pixels represent the value 0.

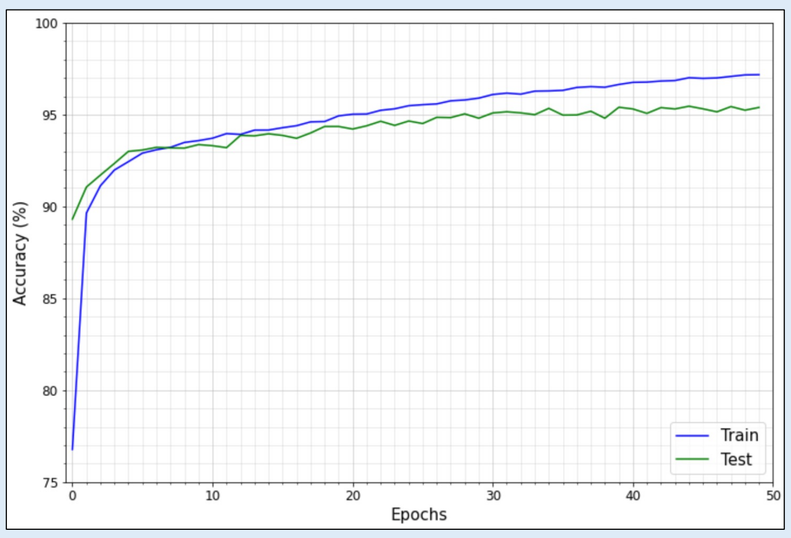

Figure 3. The training and testing accuracy curve as a function of the number of epochs of or CNN. Convergence is attained after 5-10 epochs and the final accuracy reached 0.95.

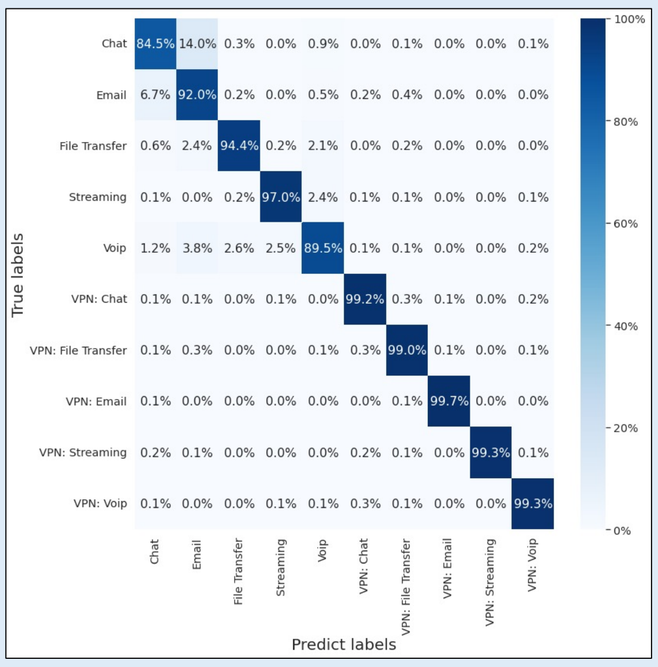

Figure 4. Confusion matric on traffic classification task for the proposed CNN over 10 classes. The rows of the confusion matric correspond to the actual class and the columns correspond to the predicted class.

TABLE I
Traffic classification result, including Recall, Precision and F_1 score.

Traffic	Rc	Pr	F_1
Chat	0.90	0.84	0.87
Email	0.81	0.92	0.86
File Transfer	0.97	0.94	0.95
Streaming	0.97	0.97	0.97
VoIP	0.94	0.90	0.92
VPN: Chat	0.99	0.99	0.99
VPN: Email	0.99	0.99	0.99
VPN: File Transfer	0.99	0.99	0.99
VPN: Streaming	0.99	0.99	0.99
VPN: VoIP	0.99	0.99	0.99
Average	0.96	0.95	0.95

Conclusion

In this poster, we presented a novel approach to interpreting the traffic classification task. We introduce a framework that emits the encrypted pseudo-random-like data and extracts useful features for further inference with CNN.

- With only the headers data, the accuracy reached 95%.
- Our model finished training within 5 minutes over 50 epochs with a minor set back on the accuracy of approximately 2% compared to SOTA result.

Future Work

- The CNN architecture optimizing and examination of other possible architectures and hyperparameters may improve results.
- Online classification was not tested in this research: QoS provisioning and routing.
- Further investigation of which part of feature selection of the header can be conducted. There may exist a more concise data representation of the full packet, which inherits all the benefits of our proposed method.

Acknowledgment

This research benefited from the consultation from Zi-Yuan Liu.