Network Traffic Classification using Deep Neural Networks

XYZ
CICS
University of Massachusetts Amherst
xyz@umass.edu

ABC
CICS
University of Massachusetts Amherst
abc@umass.edu

Abstract

Internet traffic classification serves as a base task in computer network management domain and is also highly useful in the field of cybersecurity and network resource management. With the introduction of encryption techniques in order to achieve confidentiality, traffic classification is getting increasingly difficult using the traditional methods. Therefore, the use Machine Learning models is getting common as they can learn from the available data and devise an algorithm on its own. In this paper I train and compare Deep Learning models, one uses one dimensional Convolutional Neural Network and the other uses two-dimensional Convolutional Neural Network. This paper builds on the end-to-end traffic encryption method introduced in [1]. I use ISCX VPN-nonVPN public dataset.

1. Introduction

Traffic classification is mapping the internet traffic to a specific class of application which could be later user in several cybersecurity and network management tasks. One of the main tasks in network management is the Quality of Service (QoS) and hence user experience improvement. Federated Learning is gaining popularity therefore solving privacy issues as well as having real-time traffic classification will improve user experience. Also, getting a precise and fast traffic classification method is the goal. Traffic classification can be as simple as genuine user traffic and malware traffic classification, to be of use in cyberspace. The main challenges for traffic classification are the variety in data traffic and the speed at which data is being generated. In addition, ever since encryption has become a regular practice the encrypted data classification is a difficult task in itself. For example, malware data also make use of the TLS along with the genuine data and bypass the firewall and Network Intrusion Detection System. And modern firewalls which have the capabilities to decrypt and filter data do not assure the privacy. So, traffic classification can be conveniently used in automated intrusion detection systems.

There are different encryption techniques based on different layers of OSI/ISO model. In application layer encryption applications use their own encryption technique. TLS/SSL works between application layer and transport layer. Typically, IPsec works on network layer. Using VPN and tunneling technology add to the difficulty because the encapsulation and encryption of payload follow a different set of protocols and is called protocol encapsulation. The application layer encrypted traffic classification is also called regular encrypted traffic classification [1]. The classification of data using tunneling technology is called protocol encapsulated traffic classification [1].

For traffic classification, the first step is to identify the encrypted traffic and then associate the traffic to the type of application instead of a specific application because of huge variety of applications, and the volume and variety of application data. The association of encrypted traffic to application type is called traffic characterization [1]. In this paper only identification and characterization steps are followed because of the above problems related to detailed encrypted traffic classification.

The most common traditional methods of traffic classification are based on port analysis and deep packet inspection (DPI). The issue with port analysis is the use of random ports nowadays as opposed to the assumption of the applications using common TCP and UDP ports. DPI, the more sophisticated classification techniques classify based on the protocols and information from specific data types from specific applications. One problem is, because of encryption the packet data is not interpretable to the classifier anymore and the other is that the classifier cannot always keep track of the type of application data with the increasing number of applications. This provoked the introduction of ML algorithms which can learn to associate data with classes without looking at the raw data but only the hand designed features and applying classifiers like Naïve Bayes and decision tree. This is step-by-step process which will give the local optimum at each step which is not necessarily the global optimum. So, end-to-end encryption classification, introduced by Wang et al. in [1], would be

ideal to follow as it gives global optimum.

In this paper, I train end-to-end models for traffic classification with one-dimensional convolutional neural network (1DCNN) and two-dimensional convolutional neural network (2DCNN). In both the models CNNs are the learning algorithms which take raw processed network data as input and learn from them, instead of taking the hand designed features as input. The output of the models is predicted labels. The models learn the non-linear relationships between the raw network traffic input and output classification labels.

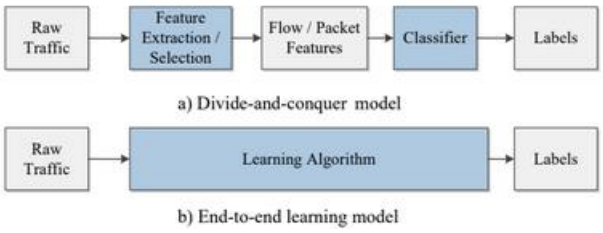

Figure 1. Machine Learning model approach versus end-to-end deep learning approach [1]

I am training a model which should be faster to train, and which can be used to provide real-time results but at the same time gives comparable evaluation metrics when compared with the model proposed by Wang et al. [1] and also with Tree-RNN model proposed by Ren et al. [6]. The network data types or packet formats and the datasets in the computer networks fields is constantly updating because of the introduction new technologies and the ever-increasing applications. And there is a need to accommodate those changes in the classifier models and if the training time can be reduced the model training could be done frequently with updated data. My study results in the model performing at acceptable level with faster training and testing speed on ISCX VPN-nonVPN (ISCX) traffic dataset [11].

The paper is organized as follows: Related work, Methodology and architecture description, evaluation, and comparison to the base model, and finally discussing the use and future work that could be done.

2. Related Work

Several earlier research focused on regular encrypted traffic. The feature choices for these models were either flow features or packet features. Flow features are selected from the data collected from routers and other devices as the traffic is sent from source to destination, these are the details and representative of traffic stream that flows following a protocol (e.g., number of bytes per seconds). Da Silva et al. [8] define the ways to identify and select flow features to improve the accuracy of a traffic classification model. Packet features are obtained from the actual data packet analysis (e.g., packet size). Wang et al. [12] used flow features with C4.5 decision tree as classifier to analyze P2P network. Coull et al. [13] used packet features and Naïve Bayes classifier to analyze iMessage.

The ongoing research in the field of network traffic classification focuses on deep learning. Lotfollahi et al. [7] developed a framework called Deep Packet which uses CNNs and Stacked Autoencoders (SAE) for traffic identification and characterization. In a recent work Ren et al. [6] have used Tree-RNN for network traffic classification on same ISCX public dataset with average precision and recall as 98.98% and 98.97% respectively.

In the end-to-end model I use I take preprocessed data in IDX format and train the model using IDX3 traffic and IDX1 labeled data as generated in [10]. I use Adam optimizer to converge training data to produce optimum results. I also use 5-fold cross-validation technique during training. The resulting hyperparameters from training method will be used in test phase.

3. Methodology

3.1. Dataset

The datasets used by most of deep learning approaches for traffic classification is not public dataset but self-collected over private network or company-based dataset which impacts the evaluation [1]. This is the case because, for earlier ML approaches the dataset needed features and not raw data. For later works ISCX dataset is the commonly used public dataset. For example, both the Tree-RNN and Deep Packet approaches use ISCX dataset. As mentioned by Heng et al. [9] in their paper, there is a lack of public datasets for network traffic management and intrusion detection system.

ISCX dataset that I use was published by Draper-Gil et al. [11] and I downloaded it from the link mentioned in [1]. This dataset has both regular encrypted traffic and protocol encapsulated traffic, each of 7 types, since a session was run using both VPN and non-VPN. This is a very diverse dataset with most frequently used applications' data. Although, I have only used 6 classes of each regular encryption and protocol encapsulated encryption. The 6 classes of encryption traffic are from email, chat, streaming, File Transfer, VoIP, P2P. So, there will be 12 class labels, 6 for VPN and other 6 for non-VPN.

The labelling I use is as mentioned in [1], since Wang et al. encountered a problem labelling browser data as VPN and non-VPN it was removed.

Content	Traffic Type
Email, Gmail	Email
	VPN-Email
ICQ, AIM, Skype, Facebook, Hangouts	Chat
	VPN-Chat
Vimeo, Youtube, Netflix, Spotify	Streaming
	VPN-Streaming
Skype, FTPS, SFTP	File Transfer
	VPN-File Transfer
Facebook, Skype, Hangout, Voipbuster	VoIP
	VPN-VoIP
uTorrent, Bittorrent	P2P
	VPN-P2P

Table 1. Labels of ISCX VPN-non VPN dataset

3.2. Data Preprocessing

The first step in end-to-end encryption classification is data preprocessing step. I obtained the preprocessed ISCX raw data traffic from the link mentioned in [1]. Wang et al. in their paper [1] have mentioned that the preprocessing of data is done using USTC-TL2016 tool developed by Wang et al. in [10]. It processes the data following four steps, traffic split, traffic clean, image generation and IDX conversion. IDX3 files have byte data of raw traffic.

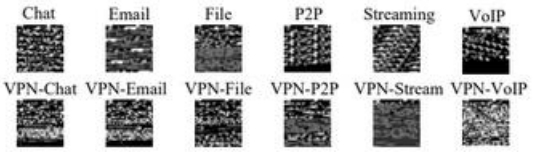

Figure 2. Preprocessed data for 12 classes

3.3. Representation and Preprocessed Results

The data representation choices that Wang et al. made in [1] is session and flow. Session is defined using five fields, source IP, destination IP, source port, destination port, and transport layer protocol used for that session. Flow is defined as traffic in one direction, source to destination. Session reviews the traffic from both the direction source to destination and destination to source.

Second condition of data representation is based on protocol layers. First is Layer 7 of OSI model or Layer 4 of TCP/IP or application layer protocol. Example, if the protocol is HTTP or HTTPS then the traffic is browser traffic, emails use SMTP. The other is all layers because many application layer protocols implement encryption methods and the required data is not accessible by the traffic classifier. For example, TLS is used by many application layer protocols including HTTPS and it encrypts the data starting from handshake.

So, there are four combinations for data representation Flow+ALL, Flow+L7, Session+ALL, Session+L7.

3.4. Model

CNNs are most commonly used in computer vision domain but have recently produced significantly good results in NLP and has also found use in Federated Learning. According to LeCun et al. [4] CNNs can find use in more than computer vision, for example, in document reading and speech recognition as well. 1DCNN models are best suitable for working with series or sequences of data e.g., language. 2DCNN have found the best use in computer vision. CNNs are also useful for classification purposes and here I needed to classify the network traffic data, so I am using CNNs. Wang et al. in [10] have also used CNN for malware classification.

Although it would not be right to compare CNNs and RNNs for training time as the training time for two models depend on multiple factors but just by design analysis CNNs are faster to train than RNNs as the RNNs process data sequentially whereas CNNs process data parallelly. For this reason, Bradbury et al. proposed Quasi-RNN in [5] using CNNs in which CNNs try to imitate RNNs but can be faster than RNNs.

In my case network traffic is continuous byte stream of data i.e., sequential data so 1DCNN will be suitable. The structure of raw data for network traffic, bytes forming packets and packets forming sessions can be compared to the sequence of characters forming words and sequence of words forming sentences in NLP. I also chose to run 2DCNN to understand the performance difference even though [1] have mentioned that 1DCNN performs better. Detailed architecture of 1DCNN is explained below.

Let $x_i \in R$ be the d-dimensional *ith* byte of raw traffic input of a session or a flow. The entire data for a session or a flow is $N \times d$ long vector, since the bytes are concatenated. Here N is number of bytes in a session or flow.

$$x_{0:N} = x_0 \dots \dots . x_{N-1} \quad (1)$$

To perform a convolution operation on this data a filter $W \in R$ is required. Each convolution operation is over the p window on x. The output of each convolution and then activation function applied after convolution is c_i.

$$c_i = f(W \cdot x_{i:i+p} + b) \quad (2)$$

Here f is Swish activation function introduced in [3] and b is bias, $b \in R$. As mentioned by Ramchandran et al. [3] Swish activation function enhances performance of a deep

model.

$$f(x) = x \cdot \sigma(x) \tag{3}$$

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{4}$$

After activation, max pooling 1D operation is applied over the feature map and maximum value is taken as the next feature. Multiple convolution layers and multiple pooling layers are used to extract the representative features, which are then passed to a fully connected layer. A dropout layer is added after this fully connected layer with p=0.2. The final fully connected layer outputs the scores for each class using Additive Margin Softmax Classifier (AM-Softmax) which tends to perform better and faster as compared to other variations of softmax classifier, as mentioned by F.Wang et al. [2]. AM-Softmax uses cosine similarity to find the scores for each class with the correct class.

Since dot product of two vectors is their magnitude multiplied by cosine of angle between them. $\cos\theta$ can be obtained by equation (5), here W is the weights and f is the feature vector from previous layer. This W in (5) is different from previous W in (2) of convolution layer which is filter. Weights and features are normalized, between [0,1].

$$cos\theta_{y_i} = \frac{W_{y_i}^T f_i}{\|W_{y_i}\| \|f_i\|} \tag{5}$$

Mentioned below is the equation for AM-Softmax Loss. Here m and s are hyperparameters and I have used the values m=0.35, s=30.

$$\mathcal{L}_{AMS} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s\cdot(cos\theta_{y_i}-m)}}{e^{s\cdot(cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y_i}^{c} e^{s\cdot cos\theta_j}}$$
$$= -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s\cdot(W_{y_i}^T f_i-m)}}{e^{s\cdot(W_{y_i}^T f_i-m)}+\sum_{j=1,j\neq y_i}^{c} e^{sW_j^T f_i}}. \tag{6}$$

Layer	Operation	Input	Filter	Stride	Pad	Output
1	Conv+ReLU	784*1	25*1	1	Same	784*32
2	1d max pool	784*32	3*1	3	Same	262*32
3	Conv+ReLU	262*32	25*1	1	Same	262*64
4	1d max pool	262*64	3*1	3	Same	88*64
5	Full connect	88*64	--	--	None	1024
6	Full connect	1024	--	--	None	2/6/12

Table 2. Parameter count of 1DCNN model

2DCNN model works the same but the kernel W slides in 2-dimension instead of one dimension.

3.5. Experimental Comparison

I compare the results of the model, on ISCX dataset, in terms of accuracy, precision, recall and time taken to train and test the model with [1][6].

I have used Keras API on TensorFlow framework to run the model on Google Colab using GPU Nvidia K80. 5-fold cross validation is used for training the model. 10% data was used as test data with a mini-batch size o 50. Loss function is a custom loss function which takes negative log of the probabilities output by AM-Softmax classifier function. Optimizer used is built-in TensorFlow Adam with learning rate 0.001 for 2 class and 0.01 for 6 class and 12 class. Training time is for Epochs 20.

I have conducted four experiments, out of which 12 class gives the result for final encrypted traffic classification. 2 class experiment only looks for data classified as VPN and non-VPN. 6 class characterizes the traffic data, which is, it maps the traffic to application type e.g., chat, email, VPN chat. For 6 class, there are two experiments, one for regular encrypted data and the other for protocol encapsulated data.

4. Evaluation

4.1. Evaluation Metrics

I have used accuracy, precision and recall for the evaluation of model. Accuracy is the measure of out of all the values that were obtained how many were predicted as part of correct class, true positive (TP) and true negative (TN). Precision is the measure of out of all the values which were predicted correct class how many were supposed to be classified to that class i.e., TP. Recall is the measure of out of all the values which belong to the class how many were correctly classified.

TP: values classified as a correct class and belongs to the class.

TN: values that do not belong to a class and are classified as does not belonging to a class.

FP: values that do not belong to a class but are classified as the value of correct class.

FN: values that do belong to a class but are classified as not belonging to a class

$$Accuracy = \frac{TP+FP}{TP+FP+TN+FN} \tag{7}$$

$$Precision = \frac{TP}{TP+FP} \tag{8}$$

$$Recall = \frac{TP}{TP+FN} \quad (9)$$

4.2. Representation

The table below compares the results among the choices of traffic representations with 1DCNN.

Experiment Accuracy % 1DCNN				Better Protocol and representation			
		All	L7	Session	Flow	All	L7
2 class	Session	68.89	62.21	1	1	2	0
	Flow	68.99	58.75				
6 class VPN	Session	56.97	57.29	2	0	1	1
	Flow	54.47	54.01				
6 class non-VPN	Session	71.45	70.89	2	0	1	1
	Flow	66.23	68.46				
12 class	Session	61.77	59.97	2	0	2	0
	Flow	60.34	58.44				
Total				7	1	6	2

Table 3. Results of experiment for 1DCNN in terms of accuracy.

When we compare session and flow, for my model Session gave better results by an average of 2.83% in accuracy for those 7 cases in which the output for session was better. Going by the definition of session and flow, session should give better results as it considers the bidirectional data and hence contains more data about the interaction between source and destination which means that the model can learn more features. In that one case where Flow performs better than Session, the accuracy differs by 0.1% only.

Next is the choice of protocol for either all the layer or only the application layer. There are 6 instances for which All layers gave better accuracy as compared to only Layer 7. The average percentage by which all layers gave better results in those 6 instances is 3.6%. And understandably so we can infer that the protocols from all layers gives better traffic representation than considering only layer 7 as they contain data from more number of protocols which can provide additional information for feature learning. For example, the TLS version can give details about the cipher suites supported by the source. The result of choice of representation, Session+All, is consistent with the state-of the-art model [1]. And only this choice should be considered for model evaluation.

4.3. CNN Model Evaluation

As mentioned earlier, I trained two models, 1DCNN and 2DCNN. My analysis of the two models shows that the choice of representation for both the models give same result, Session+All. Although, the accuracies in case of 2DCNN model is somewhat less than 1DCNN model the values are still comparable. I used one model with 5-fold cross validation and one model without cross validation for each of the learning models. I trained the models without cross validation for 30 epochs and the average accuracy over test data was 2.1% less for 1DCNN model whereas 0.89% less for 2DCNN model.

Experiment Accuracy % 2DCNN				Better Protocol and representation			
		All	L7	Session	Flow	All	L7
2 class	Session	68.31	62.61	2	0	2	0
	Flow	64.06	58.67				
6 class VPN	Session	55.95	51.33	1	1	1	1
	Flow	52.98	60.4				
6 class non-VPN	Session	66.42	64.76	2	0	1	1
	Flow	61.08	63.41				
12 class	Session	60.3	55.43	2	0	2	0
	Flow	60.12	54.88				
Total				7	1	6	2

Table 4. Results of experiments for 2DCNN in terms of accuracy

	Average precision over test data for all four experiments
1DCNN	89.32%
2DCNN	81%

Table 5. Results of the two CNN models in terms of precision

The recall results for both the CNN models were not up to the mark. The average recall for 1DCNN for 12 class experiment was 60.1%. And the average recall for 2DCNN was 58.66%.

We can also see from the tables that both IDCNN and 2DCNN models did not particularly give good accuracy for second experiment which is 6 class VPN. By intuition we can say that this was because of all the hidden data in the protocol encapsulated traffic and it is difficult to characterize as compared to regular encrypted traffic.

Finally, as I am using the activation functions and loss function which are faster and have proven to given better results [2][3], I expected to get better results in terms of training and testing time. So, the time it took me to train the models is considerably less. An important observation was that 2DCNN trained faster than 1DCNN.

4.4. Model Comparison

I compare the results of my model with the state-of-the-art method in [1] for 1DCNN and also with state-of-the-art RNN model. Since the hardware and the amount of data used for training and testing differs, timing

could not be precisely compared. Still the Table 5 does the comparison.

	Training time(s)	Test time(s)
1DCNN	55	0.34
Tree-RNN	54.20	1.22
State of the art 1DCNN	56.37	0.4

Table 6. Comparison of training time and test time with other models

5. Discussion and future work

The 1DCNN model can learn the features on its own and this is a huge benefit when compared to traditional methods. The experiment results were not as good as expected in terms of accuracy but for precision of the model, average precision of 1DCNN was 89.32% and 2DCNN was 81%, which is satisfactory. With the more advanced hardware the training and test time could be reduced even more. The most important use that I can see is the improvement in QoS and user experience. Also, real-time classification would be useful with Federated learning once it could be implemented with accepted level of data privacy. One important takeaway was that the results from the model I trained follow the same trend as obtained by [1]. For example, the choice of protocols of ALL layers gave better results for my model as well as model proposed by [1].

6. Conclusion

The encrypted traffic classification for both regular and protocol encapsulated encryption can be done using the deep learning models. Although the training time and the test time are going to be an issue. In the model I trained, even though the train and test time for model was less the accuracy was compromised by a substantial margin. As already mentioned by [1], the end-to-end encryption technique gives more optimal solution as compare to divide and conquer ML models. But the time taken in both the techniques is large so neither can be used for real-time classification. I found that using Swish and AM-Softmax with 1DCNN and 2DCNN models did produce the results faster but not accurate. There can be a few reasons for that including the choice of hyperparameters. I will continue to train the model with better choices of hyperparameters.

References

[1] W. Wang, M. Zhu, J. Wang, X. Zeng and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017, pp. 43-48, doi: 10.1109/ISI.2017.8004872

[2] F. Wang, J. Cheng, W. Liu and H. Liu, Additive Margin Softmax for Face Verification, in IEEE Signal Processing Letters, vol. 25, no. 7, pp. 926-930, July 2018, doi: 10.1109/LSP.2018.2822810.

[3] Prajit Ramachandran∗, Barret Zoph, Quoc V. Le SWISH: a self-gated activation function. arXiv:1710.05941v1 [cs.NE] 16 Oct 2017

[4] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521,** 436–444 (2015). https://doi.org/10.1038/nature14539

[5] Bradbury, J., Merity, S., Xiong, C. and Socher, R., 2016. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*.

[6] Ren, X., Gu, H. and Wei, W., 2021. Tree-RNN: Tree structural recurrent neural network for network traffic classification. *Expert Systems with Applications*, *167*, p.114363.

[7] Lotfollahi, M., Siavoshani, M.J., Zade, R.S.H. and Saberian, M., 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, *24*(3), pp.1999-2012.

[8] Da Silva, A.S., Machado, C.C., Bisol, R.V., Granville, L.Z. and Schaeffer-Filho, A., 2015, September. Identification and selection of flow features for accurate traffic classification in SDN. In *2015 IEEE 14th International Symposium on Network Computing and Applications* (pp. 134-141). IEEE.

[9] Y. Heng, V. Chandrasekhar and J. G. Andrews, UTMobileNetTraffic2021: A Labeled Public Network Traffic Dataset, in *IEEE Networking Letters*, vol. 3, no. 3, pp. 156-160, Sept. 2021, doi: 10.1109/LNET.2021.3098455.

[10] W. Wang, X. Zeng, X. Ye, Y. Sheng and M. Zhu, Malware Traffic Classification Using Convolutional Neural Networks forRepresentation Learning In the *31st International Conference on Information Networking (ICOIN)*, 2017.

[11] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun and A. A. Ghorbani, Characterization of encrypted and VPN traffic using time-related features, In Proceedings of *the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407-414, 2016.

[12] D. Wang, L. Zhang, Z. Yuan, Y. Xue and Y. Dong, Characterizing application behaviors for classifying p2p traffic, *Computing Networking and Communications (ICNC) 2014 International Conference on. IEEE*, pp. 21-25, 2014.

[13] S. E. Coull and K. P. Dyer, Traffic analysis of encrypted messaging services: Apple iMessage and beyond, *ACM SIGCOMM Comput. Commun. Rev.*, pp. 5-11, 2014.