NYU Deep Learning Final Project — Encrypted Network Traffic Classification Using A 1D-CNN

Written by Marc Chiu, Sathish Vijay, Hafez ElBoghdadi

mmc9967@nyu.edu, svk319@nyu.edu, ae2405@nyu.edu

Abstract

Using data encryption in computer networking communications is widely used for multiple purposes, some are benign for ensuring privacy and confidentiality of the traffic between different hosts or remote connectivity via VPN tunnels, while some others are malicious like hiding malware payload, bypassing corporate internet policy restrictions, or bandwidth consumption on streaming, which motivates the classification of encrypted network traffic. This model/classification is commonly used to apply Quality-Of-Services policies on the traffic, for example by prioritizing the bandwidth for work-related activities like emailing and voice calls while limiting non-work related traffic like streaming or social networking. We attempted to classify network traffic into six different categories: Email, Chat, Streaming, File Transfer, VoIP, P2P using a 1D CNN. We trained and tested using the public ISCX dataset which consists of both VPN and non-VPN traffic. Resulting in a possible 12-class categorization problem: VPN-Email, non-VPN email, VPN-Chat, non-VPN Chat, etc.

Literature Review

We surveyed multiple literature papers about the applications of NLP in Malware detection, some techniques were based on the specific classification task required such as Application Identification Vs Malware Identification, and 1D=CNNs were among the most common techniques and are usually used as a stand-alone network, or with LSTM networks [1]. Determining the input features is one of the main criteria in model selection, so we opted to follow an end-to-end Representation Learning approach used in [2] that requires only the raw PCAP files as an input.

Methodology

i. Dataset

One of the main obstacles is to acquire a proper dataset since most of the cybersecurity datasets are either confidential or only privately accessible. We decided to use the public ISCX dataset used in [2] "VPN/non VPN traffic dataset (ISCXVP- N2016)", which consists of raw PCAP files of VPN/non-VPN traffic from 6 classes.

Traffic Type	Content
Email	Email、Gmail (SMPT, POP3,IMAP)
VPN-Email	
Chat	ICQ, AIM, Skype, Facebook, Hangouts
VPN-Chat	
Streaming	Vimeo, Youtube, Netflix, Spotify
VPN-Streaming	
File transfer	Skype, FTPS, SFTP
VPN-File transfer	
VoIP	Facebook, Skype, Hangouts, Voipbuster
VPN-VoIP	
P2P	uTorrent, Bittorrent
VPN-P2P	

Table 1: Types of Traffic

The files must be passed through an initial data preprocessing stage with 4 main operations: 1) Data Splitting into unified-size files (output is either PCAP or binary), Padding and Sampling are used to maintain a fixed-size input to the CNN model, 2) Data Sanitization and de-duplication to remove or randomize the hosts IP and MAC addresses to avoid their bias on the model, 3) Convert the Traffic data into a Black and White Image embedding, 4) Conversion to IDX format (formatted to optimize memory storage of MINST image dataset), the resultant files consisted of the first 784 bytes of features and are labeled by class/Traffic type and VPN/non-VPN status. After pre-processing is done there will be 3 classifications: 1.) VPN/non-VPN, 2.) Chat/Streaming/Email/File Transfer-/VoIP/P2P, 3.) VPN-Chat/Non-VPN Chat/VPN Email/etc. Within each of these classes there will also be the choice of which features to extract: SessionL7, SessionAllLayers, FlowL7, FlowAllLayres. For the 6 class classification there will also be the choice of selecting VPN or non-VPN data: VPN-SessionL7, nonVPN-SessionL7, etc. Given time constraints we were able to pre-process the data but not able to link the process to google colab.

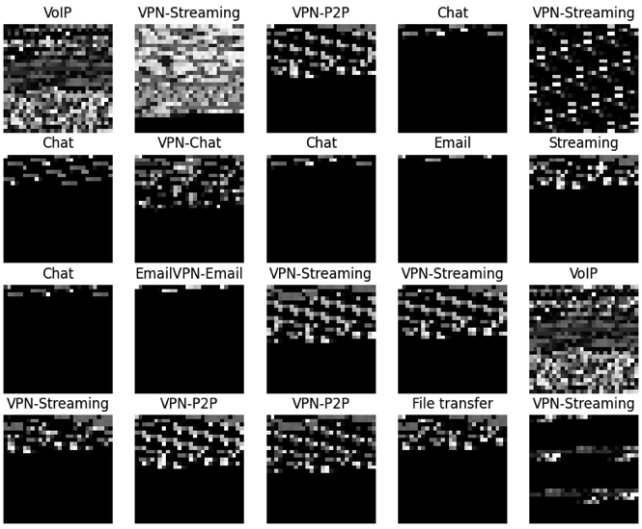

Figure 1: 12-class Data Visualized as Images

i. Classifications

Recalling the OSI Model 7 Layers protocols, the 2 main encryption techniques discussed at [2] are ; a) Regular Encryption: The inherent encrypted traffic of an application, defined by the application semantics ex; BitTorrent encryption, in this case only the application layer L7 is encrypted or has some encrypted data fields, but the rest of the layers are not encrypted, and b) Encapsulation Encryption: The traffic data layers are encrypted upwards from the Network L3 layer (like IPsec) or from the Presentation L6 upwards. The authors we replicated opted to rely on Representation Learning, by building a DL model that can extract the features directly from the raw traffic without having to define features directly.

There are 6 classes of traffic/labels called Traffic Types, indicating the encryption type not algorithm, and they are combinations of VPN/Non- VPN against 6 different traffic types: Email, Chat, Streaming, File Transfer, VoIP and P2P. There are four types of classifications/Experiments: a) Binary Classifier: VPN or not VPN , b) 6-Class Classifier: Which one of the 6 Traffic types with VPN, c) 6-Class Classifier: Which one of the traffic types without VPN, d) 12-Class classifier: Combination of VPN/Non-VPN with the 6 classes. The main metrics for success will be accuracy, precision, and recall

iii. Model

Since our data can be represented as images we decided to compare a 1D Convolutional Neural Net with a 2D Convolutional Neural Net. We used A 1D Convolutional Neural Network (CNN) due to its ability to capture local temporal patterns in traffic data. Our data is represented as a sequence of features that change over time, such as the packet headers. The benefit of using a 1D CNN is that they are computationally efficient and require fewer parameters than traditional 2D CNN. We then compared our results using a 1D to a 2D CNN by converting our 784 vectors to 28x28 images to compare accuracy, precision, and recall. We structured our model based on the paper [2] as a base line model and then tried to improve upon the model by testing various hyperparameters and by adding more convolutional layers. The base-line model consists of two convolutional layers each respectively followed by a maxpooling layers, a fully connected linear layer, and finally an output layer of varying size depending on the classification problem.

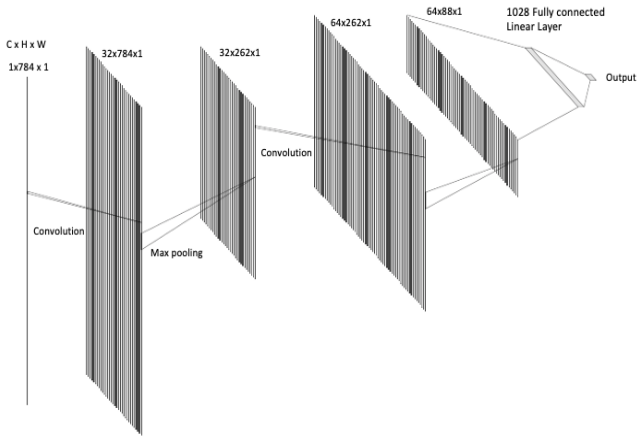

Figure 2: 1D CNN Diagram

Layer	Operation	Input (Batch Size, H, W, Cin)	For Conv: Filter (H,W,Cin,Cout) For Max Pool: Filter (N,H,W,C) For Linear : Weight (N,H,W)	Stride (N,H,W,C)	Pad (H,W)	Output (Batch Size, H, W, C) Linear Output : (N, Feat Output)	Model
1	Conv+ReLu 1	N,1,784,1	1,1,25,32	1,1,1,1	Same or (0,12)	N,1,784,32	CNN-1D
2	1D Max Pool 1	N,1,784,32	1,1,3,1	1,1,3,1	Same or (0,1)	N,1,262,32	CNN-1D
3	Conv+ReLu 2	N,1,262,32	1,1,25,64	1,1,1,1	Same or (0,12)	N,1,262,64	CNN-1D
4	1D Max Pool 2	N,1,262,64	1,1,3,1	1,1,3,1	Same or (0,1)	N,1,88,64	CNN-1D
5	Linear Full Connect 1	N, (88 x 64 x 1)	(88 x 64 x 1) , 1024	N/A	N/A	N, 1024	CNN-1D
6	Dropout	N, 1024	Keep/Drop Ratio	N/A	N/A	N, 1024	CNN-1D
7	Linear Full Connect 2	N, 1024	1024, No. of Classes (2 or 6 or 12 Depending on Experiment Category)	N/A	N/A	N, No. of Classes	CNN-1D
8	Softmax	N, No. of Classes	N/A	N/A	N/A	N, No. of Classes	CNN-1D

Table 2a: Base 1D CNN Model

Layer	Operation	Input (Batch Size, H, W, Cin)	For Conv: Filter (H,W,Cin,Cout) For Max Pool: Filter (N,H,W,C) For Linear : Weight (N,H,W)	Stride (N,H,W,C)	Pad (H,W)	Output (Batch Size, H, W, C) Linear Output : (N, Feat Output)	Model
1	Conv+ReLu 1	N,28,28,1	N,5,5,32	1,1,1,1	Same or (2,2)	N,28,28,32	CNN-2D
2	2d Max Pool 1	N,28,28,32	1,2,2,1	1,2,2,1	Same or (0,0)	N,14,14,32	CNN-2D
3	Conv+ReLu 2	N,14,14,32	1,5,5,32,64	1,1,1,1	Same or (2,2)	N,14,14,64	CNN-2D
4	2d Max Pool 2	N,14,14,64	1,2,2,1	1,2,2,1	Same or (0,0)	N,7,7,64	CNN-2D
5	Linear Full Connect 1	N, (7 x 7 x 64)	(7 x 7 x 64) , 1024	N/A	N/A	N, 1024	CNN-2D
6	Dropout	N, 1024	Keep/Drop Ratio	N/A	N/A	N, 1024	CNN-2D
7	Linear Full Connect 2	N, 1024	1024, No. of Classes (2 or 6 or 12 Depending on Experiment Category)	N/A	N/A	N, No. of Classes	CNN-2D
8	Softmax	N, No. of Classes	N/A	N/A	N/A	N, No. of Classes	CNN-2D

Table 2b: Base 2D CNN Model

Tuning Parameters

Once the model had been created the next step was to initialize all the parameters. The parameters were initialized using two different normal distributions. To initialize the weights of the linear layer we used a normal distribution with a mean of 0 and a standard deviation of 1e-3. To initialize the weights of the convolutional layers we used Kaiming initialization.

We chose to initialize the convolutional layers using Kaiming initialization because it uses a scaling factor which aids in mitigating the exploding/vanishing gradient problem for networks that use the ReLu activation. The Kaiming initialization changes the normal distributions we are sampling from by changing the standard deviation at each layer, meaning layers further down the network will have slightly incremented means, thus making our

model more robust. We experimented with changing various hyperparameters including: drop-out probability, optimizer, learning rate, training/validation split ratio, batch size, and number of epochs. We trained using both the ADAM and SGD optimizers as well as used Cyclical Learning Rates to overcome plateaus in training.

The best model results were achieved at using Validation Ratios = 0.8 of the training data, batch size=50, 60 Epochs, ADAM Optimizer, normalized images between [0,1], CNN-1D model, Dropout ratio=1 (keep all) and by selecting the best validation losses.

Model ID	100	102	188	184	93	330
model	CNN-1D	CNN-1D	CNN-1D	CNN-1D	CNN-1D	CNN-2D
drop_probability	0.1	0.1	0.1	0.1	0	0
OptimizerType	SGD	SGD	ADAM	ADAM	SGD	SGD
SGD_LR	0.0001	0.0001			0.01	0.01
SGD_momentum	0.9	0.9			0.9	0.9
SGD_WeightDecay	0	0			0.9	0.9
ADAM_LR			0.0001	0.0001		
Classifier Class	12	12	6	6	6	6
Connection	Flow	Session	Session	Flow	Session	Flow
Protocol	AllLayers	AllLayers	AllLayers	AllLayers	AllLayers	L7
Encryption			Vpn	Vpn	Novpn	Vpn
key	12-Flow-AllLayers-None-CNN-1D-100	12-Session-AllLayers-None-CNN-1D-102	6-Session-AllLayers-Vpn-CNN-1D-188	6-Flow-AllLayers-Vpn-CNN-1D-184	6-Session-AllLayers-Novpn-CNN-1D-93	6-Flow-L7-Vpn-CNN-2D-330
trainableParameters 1E6	5.832588	5.832588	5.826438	5.826438	5.826438	3.270536
duration	90.6016	77.0586	41.2774	55.0951	207.65	18.66
Training Accuracy	0.857	0.8599	1	1	1	1
Training Loss	0.3274	0.3024	0	0	0	0.02
Validation Accuracy	0.8418	0.8418	0.9939	0.9869	0.94	0.94
Validation Loss	0.3563	0.3393	0.029	0.0542	0.36	0.33
Test Loss	0.3717	0.3368	0.0625	0.0788	0.35	0.3
Test Accuracy	0.8345	0.8422	0.9869	0.9859	0.94	0.94
Epoch Stop					79	18
folder	Models_2023-05-14__12-18-17	Models_2023-05-14__12-18-17	Models_2023-05-14__20-58-33	Models_2023-05-14__20-58-33	Models_2023-05-14__22-59-39	Models_2023-05-14__19-56-09
Avg-TP	0.069600001	0.070200004	0.164399996	0.164299995	0.159999996	0.159999996
Avg-FP	0.0138	0.0132	0.0023	0.0024	0.01	0.01
Avg-FN	0.0138	0.0132	0.0023	0.0024	0.01	0.01
Avg-TN	0.902899981	0.903500021	0.831099987	0.830900013	0.819999993	0.819999993
Avg-Precision	0.856599987	0.877099991	0.977500021	0.979200006	0.949999988	0.930000007
Avg-Recall	0.836799979	0.859099984	0.970600009	0.974399984	0.949999988	0.939999998
Avg-Accuracy	0.97240001	0.973699987	0.995500028	0.995199978	0.980000019	0.980000019
Avg-F1	0.842899978	0.86500001	0.97390002	0.976499975	0.949999988	0.930000007
earlyStop	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE
earlyStopMax	-1	-1	-1	-1	-1	15

Figure 3: Best Models Parameters and Results

Training and Testing

There are three classification problems: 2 Class, 6 Class, and 12 Class. Within 2 Class or 12 Class we can choose 4 different datasets: SessionL7, SessionAllLayers, FlowL7, and FlowAllLayers. Within the 6 classifcation there are 8 different datatsts: VPN-SessionL7, nonVPN-SessionL7, VPN-SessionAllLayers, nonVPN-SessionAllLayers, etc. Hence, there are 16 different combinations of classifications and datasets. We trained and tested all 16 combinations using various hyperparemters on both a 1D CNN and 2D CNN and compared the results.

key	Training_Accuracy	Training_Loss	Validation_Accuracy	Validation_Loss
12-Flow-AllLayers-None-CNN-1D	1.0000	0.0200	0.9500	0.1800
12-Flow-AllLayers-None-CNN-2D	0.9900	0.0300	0.9000	0.5200
12-Flow-L7-None-CNN-1D	0.9500	0.1200	0.8300	0.7400
12-Flow-L7-None-CNN-2D	0.9500	0.1200	0.8300	0.8600
12-Session-AllLayers-None-CNN-1D	1.0000	0.0100	0.9400	0.2300
12-Session-AllLayers-None-CNN-2D	1.0000	0.0200	0.8700	0.6600
12-Session-L7-None-CNN-1D	0.9400	0.1500	0.8600	0.4800
12-Session-L7-None-CNN-2D	0.9400	0.1400	0.8600	0.5600
2-Flow-AllLayers-None-CNN-1D	1.0000	0.0001	0.9995	0.0018
2-Flow-AllLayers-None-CNN-2D	1.0000	0.0006	0.9998	0.0012
2-Flow-L7-None-CNN-1D	0.9937	0.0169	0.9341	0.3050
2-Flow-L7-None-CNN-2D	0.9922	0.0222	0.9243	0.4305
2-Session-AllLayers-None-CNN-1D	1.0000	0.0001	0.9997	0.0007
2-Session-AllLayers-None-CNN-2D	1.0000	0.0003	0.9996	0.0013
2-Session-L7-None-CNN-1D	0.9992	0.0028	0.9576	0.2475
2-Session-L7-None-CNN-2D	0.9987	0.0049	0.9579	0.2619
6-Flow-AllLayers-Novpn-CNN-1D	1.0000	0.0200	0.9200	0.3200
6-Flow-AllLayers-Novpn-CNN-2D	1.0000	0.0100	0.8800	0.7700
6-Flow-AllLayers-Vpn-CNN-1D	1.0000	0.0200	0.9800	0.0600
6-Flow-AllLayers-Vpn-CNN-2D	1.0000	0.0100	0.9800	0.0900
6-Flow-L7-Novpn-CNN-1D	0.9200	0.1700	0.8400	0.5200
6-Flow-L7-Novpn-CNN-2D	0.9200	0.1800	0.8200	0.6200
6-Flow-L7-Vpn-CNN-1D	1.0000	0.0200	0.9400	0.2700
6-Flow-L7-Vpn-CNN-2D	1.0000	0.0200	0.9400	0.3100
6-Session-AllLayers-Novpn-CNN-1D	1.0000	0.0200	0.8900	0.3900
6-Session-AllLayers-Novpn-CNN-2D	1.0000	0.0100	0.8900	0.7000
6-Session-AllLayers-Vpn-CNN-1D	1.0000	0.0300	0.9800	0.0800
6-Session-AllLayers-Vpn-CNN-2D	1.0000	0.0100	0.9600	0.1300
6-Session-L7-Novpn-CNN-1D	0.9012	0.2144	0.8369	0.5363
6-Session-L7-Novpn-CNN-2D	0.9009	0.2207	0.8283	0.5303
6-Session-L7-Vpn-CNN-1D	1.0000	0.0200	0.9500	0.2100
6-Session-L7-Vpn-CNN-2D	1.0000	0.0200	0.9500	0.2000

Figure 4: Training Results

We also compared the results to the original paper [2] we tried to replicate. As shown in the charts we were able to improve both 12-class and 6-class classification. 2-class classification was already at 99 percent so we did not improve it. In addition, we can also see that a 1D CNN preformed equally or favorably in every single category

Experiment Accuracy				**Which is better**			
		All	L7	Session	Flow	All	L7
Exp 1	Session	99.9	95.4	2	0	2	0
	Flow	99.9	92.3				
Exp 2	Session	81.7	83.0	1	1	0	2
	Flow	81.8	82.5				
Exp 3	Session	98.3	95.3	1	1	2	0
	Flow	98.6	93.0				
Exp 4	Session	86.6	83.7	2	0	2	0
	Flow	86.5	81.0				
Total	**---**			**6**	**2**	**6**	**2**

Table 3: Results of the Paper [2]

Improvements

We tried to explore different design choices from the lecture model, so we tested +600 different tuned models to able to boost the accuracy in lower training time while maintaining the same general model details, we used multiple different features such as ;

1- Implemented a function to use Cyclical Learning Rate to dynamically tweak the Learning Rates and eliminate plateaus during training [5].

2- Implemented early stoppage to halt model tuning if no reduction in validation loss happened after a max no. of epochs (15 in our case).

3- Tested different optimizers : The paper used only GD with a fixed Learning Rate, we tried using SGD and ADAM

with different Learning rates/Momentum and Weight decay Regularization.
4- Normalized the images to [0,1] prior to model tuning.
5- Tested for different number of epochs and different dropout layers probabilities.
6- Used Pandas for Models hyper-parameters generation to test different model in a structured format.
7- Achieved higher accuracy/precision/recall than the original paper without changing the model architecture, some models achieved the same accuracy in considerably less time.
8- Implemented Xavier and Kaiming initialization for the linear and Conv1D model weights respectively, in order to keep the variance of all layers to be similar and avoid exploding or vanishing gradients.

Observations

We split the model selection among the 4 different experiments (categories) 1- Session data is better than Flow data for model accuracy which is expected since the Session is bi-directional.
2- All Layers data is better than L7 for model accuracy since most of L7 identification data becomes redundant once the main L7 headers are captured.
3- The binary classifier achieved near perfect results so it was excluded from the following models testing to focus on the harder 6 and 12 classes classification.

Classifier Classes	Encryption	Description
2	N/A	VPN/Not VPN
6	vpn	VPN 6 Applications
6	Novpn	Non-VPN 6 Applications
12	N/A	VPN/Non-VPN 6 Applications

CNN/Pooling Parameters Design Plane

1- Using the output/input dimensions equations, we derived some equations for other possible values for the filters padding/stride/window sizes whilst maintaining the input/output dimensions ratios, assuming I Input dimension, O output dimension, F filter, S stride and P padding across this dimension , assuming α to be a small quantity $\alpha\epsilon[0, 1($

$$O = Floor(\frac{I + 2P - F}{S} + 1) \rightarrow O + \alpha = \frac{I + 2P - F}{S} + 1$$

To cancel the dependency on the input dimensions, assume $O = I/m$ and $m = 1/S$ where m is a dimension multiplier between input/output dimensions, for example if the output dimensions are half of the input then $m = 1/2 \rightarrow S = 2$
By solving for Possible F in terms of a chosen P : $F = 2P - \frac{\alpha - 1}{m}$, For F to be an integer then $\alpha - 1 = Km, K = 0, \pm1, \pm2, .. \rightarrow 0 \leq Km + 1 < 1 \rightarrow -1 \leq Km < 0 \rightarrow \frac{-1}{m} \leq K < 0$, but since by definition K is an integer, then we can restrict K to be -ve to impose the R.H.S. of the inequality $\frac{-1}{m} \leq K \rightarrow -s \leq K : K = -1, -2, ...$ Then $F = 2P - K$, setting $P\epsilon\{P_{min}, P_{min} + 1, ..., P_{max}\}$
For example when $P_{min} = 0, P_{max} = 2 and S = 2 \rightarrow m = \frac{1}{S} = 0.5 \rightarrow K \geq -S \rightarrow K = -2, -1$ then the possible values of Filter Sizes $F = 2P - K \rightarrow F|_{P=0} = 1, 2, F|_{P=1} = 3, 4, F|_{P=2} = 5, 6$, and in all cases the output dimension will be a ratio of the input dimensions, in this case half of it ; $O = \frac{I}{S} = mI = \frac{I}{2}$.
Using this way by choosing a value of S and limits of P we can derive different values of F that can satisfy the same dimensions ratios independent of the input dimensions, in the paper the first layer Conv1D used F=25, P=12, S=1 so we can notice that here K = -1 only, so the only value possible for F when P=12 is F=2P+1 = 25 , If the Ceiling mode is used in the Convolutional function then a similar equation can be derived but with different sign of α .
2- For cascaded layers i.e. sequential convolution layers or max pooling functions, the net effect will be a product of all layers m-factor, e.g. $O = m_1 m_2 m_3 I$ for 3 successive layers, so we can put another design constraint by setting a minimum output dimension then $\frac{O}{I} = \frac{O_{min}}{I} = \Pi_{i=1}^{NumberofLayers} m_i = m_{effective}$

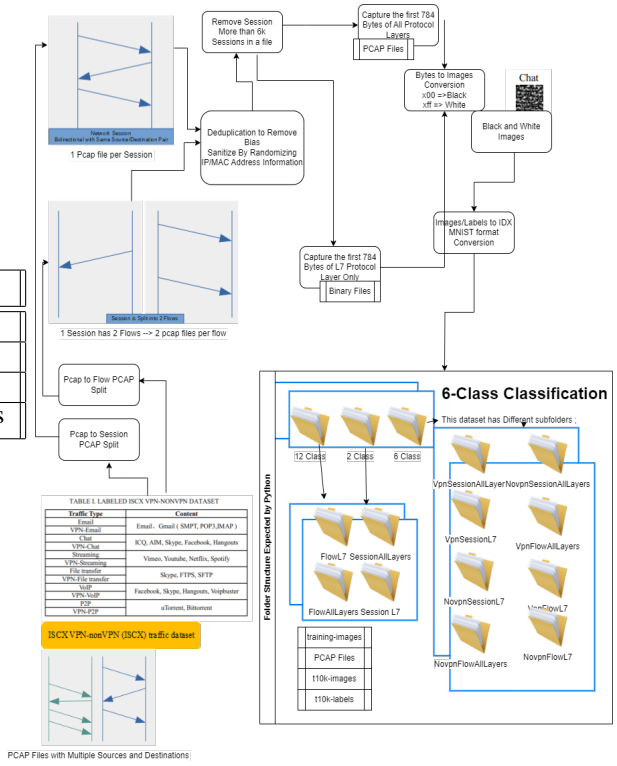

Figure 5: Dataset Pre-Processing Steps

Future Work

We plan to create a dynamic model using reinforcement learning based model paired with the current model both as a stand alone as well as a scalable distributed model.

Github Repository

https://github.com/Svk319/EncryptedTrafficClassificationByCNN

References

[1] S. Rezaei and X. Liu, "Deep Learning for Encrypted Traffic Classification: An Overview," in IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019, doi: 10.1109/ MCOM.2019.1800819. .

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

[3] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye and Yiqiang Sheng, "Malware traffic classification using convolu- tional neural network for representation learning," 2017 In- ternational Conference on Information Networking (ICOIN), Da Nang, Vietnam, 2017, pp. 712-717, doi: 10.1109/ ICOIN.2017.7899588.

[4] https://medium.com/analytics-vidhya/creating-a-custom-dataset-and-dataloader-in-pytorch-76f210a1df5d

[5] Leslie N. Smith, "Cyclical Learning Rates for Training Neural Networks", Presented at WACV 2017; https://arxiv.org/abs/1506.01186