A CNN Based Encrypted Network Traffic Classifier

Zulu Okonkwo
Griffith University, Australia,
zulu.okonkwo@griffithuni.edu.au

Qinyi Li
Griffith University, Australia,
quinyi.li@griffith.edu.au

Ernest Foo
Griffith University, Australia,
e.foo@griffith.edu.au

Zhe Hou
Griffith University, Australia,
z.hou@griffith.edu.au

ABSTRACT

The internet is responsible for global connectivity and ensuring its safety is a paramount task for governments and organisations. Cybersecurity concerns led to the encryption of over 87% of internet traffic. Encryption ensures security by improving privacy between sender and receiver but creates a problem of in-accurate traffic classification. Previous papers have used Artificial Intelligence to address this problem, however issues such as model simplicity, complexity, imbalanced dataset etc, are problems yet to be addressed. Overfitting, underfitting and ultimately poor classification are outcomes of poorly designed models. This paper applies deep learning to the problem of encrypted traffic classification. A Convolutional Neural Network (CNN) is used to address this problem. An eleven layered architecture is designed and trained with a range of images generated from the metadata of encrypted traffic. At its core, the design is made less complex for understandability and deals with overfitting. The proposed model is assessed with the standard metrics of accuracy, precision, recall and F_1 score then compared to a baseline model. The model is trained and tested for seven classification problems, using three encryption types (https, vpn, tor). For all classification tasks, the proposed model achieved accuracies ranging from 91% - 99%, which is an indication of optimum generalization strength. Our model outperformed the baseline model which had accuracies ranging from 67.6% - 99%, an indication of poor generalization strength.

CCS CONCEPTS

• Network security • Artificial Intelligence • Machine Learning

KEYWORDS

Classification, Encryption, Network traffic, Deep Learning, Convolution Neural Network

ACM Reference format:
Zulu Okonkwo, Ernest Foo, Qinyi Li. and Zhe Hou. 2022. A CNN based Encrypted Network Traffic Classifier. In *Proceedings of Australasian Computer Science Week (ACSW 2022), February 14-18, 2022, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 10 pages.

1 Introduction

The internet has been responsible for keeping the world digitally connected. Over the past decades, internet usage has increased geometrically. Remarkable advancements in technology have been made possible because of the internet. The rapid development of the internet has made it a dynamic space, accommodating different forms of data and protocols. In a bid to ensure privacy of data over the internet, encryption has been widely adopted. According to the 2021 transparency report by google, more than 90% of internet traffic is encrypted [1]. While encryption ensures privacy of information, it also comes with some disadvantages, one of which is the issue of network traffic classification. Accurately classifying network traffic is crucial for network resource management such as firewall, monitoring, anomaly, and intrusion detection [2]. In the case of anomaly detection, it can be detrimental to a network's cybersecurity outlook and cause a major incident if malicious traffic finds its way unnoticed into a network. A new trend noticed during the COVID-19 pandemic was an increase in Business email compromise (BEC) scams, which have become more organized, enhanced, and modernized to bypass cybersecurity protocols [3]. Watchguard (2021) reported 91.5% of malware in the second quarter of 2021 arrived over encrypted connection [22]. As researchers continue to develop resilient encryption methods and protocols, it is also important that security systems are developed to accurately analyse and classify traffic over these protocols while preserving privacy and confidentiality.

Classifying internet traffic is usually at the core of every security system design. How well the system can detect unusual traffic from benign traffic can save an organization from cyber-attacks. The issue of false positives also comes up when IDS (Intrusion detection systems) are discussed. False positives and negatives can also be hinderances to security systems achieving their goal of accurate traffic classification. Due to the massive rate at which data is transferred over the internet, analysing every packet will induce latency to the network. Looking at mitigating these issues and properly classifying network traffic, we utilize deep learning for encrypted traffic analysis and classification.

The conventional ways of network traffic classification are flow based and payload-based methods [4]. The flow-based method utilizes statistical features of traffic flows. A traffic flow is categorized by the packet's source/destination IP addresses, ports, and protocols. This method is effective in detecting and classifying intrusion and malicious traffic. The flow-based method, however, does not perform well when traffic is strongly encrypted e.g., if a TOR browser is used. It also requires high storage capacity to store flows, which can be time and resource consuming. The payload-based method also known as deep packet inspection, analyses every packet in a flow, in most cases, it decrypts the packet and scrutinizes its contents. This method is good for malware and threat detection, but it breaches privacy and is ineffective for representing the actual network behaviour. Understanding network behaviour is important for traffic classification and network security. To preserve the privacy of information, the flow-based model is the most widely utilized model.

Deep learning is a subset of machine learning and artificial intelligence ecosystem [26]. Deep learning tries to mimic the human brain by learning patterns from a pool of data, then makes near accurate predictions based on the learnt information. It has been applied to numerous problems ranging from visual recognition, natural language processing, fraud detection etc. Recently, researchers applied deep learning to the problem of encrypted traffic classification; it was used to classify network data according to specific parameters, further research in this area is still underway. Network traffic is transmitted on secure channels which encrypts traffic content. By leveraging deep learning, high level properties of traffic flows can be extracted and used to classify traffic with high accuracy. Deep learning heavily relies on statistical concepts for its basic operations. Feedforward neural networks [23] are one of the most primary deep learning models. At its core, it defines a mapping and learns the value for the parameters that results in the best function approximation [23]. Convolutional Neural Networks (CNN) are feed forward networks mainly used for image classification tasks. CNN are trained using back propagation.

This paper develops a CNN based traffic classifier (supervised learning). We achieve our goal by investigating the advantages of using deep learning for the problem of traffic classification. A unique feature extraction process [5] is modified to properly represent the metadata of traffic flows as images. Due to the unstructured nature of encrypted traffic, we use the metadata rather than the encrypted bits for analysis. The model is trained and tested on the traffic flow images. We evaluate the performance of our model using standard metrics.

The main contribution of this paper is as follows,

- Improved the feature extraction process for traffic representation to better identify traffic flows.
- Development of an optimized CNN based classifier with high generalization strength for identifying and classify encrypted network traffic.

Section 2 of this paper discusses the gaps unaddressed by previous works. Our proposed architecture and experimental setup are discussed in sections 3 and 4. In Section 5, we report our results and discuss them in 6. Section 7 concludes the paper.

2 Previous Work

Machine learning and deep learning methods have recently been applied to the problem of internet traffic classification with the latter predominantly adopted within the last five years. CNN and Long-short term memory (LSTM) models are the most frequently used deep learning models for this task. The models are sometimes used together [8], [9], [10] or separately [4], [5]. These papers used various models to achieve different accuracies with different datasets. Most of the designs are complex in nature and do not address the possibility of overfitting during the training and testing process. Wang et al., (2017) was able to achieve an accuracy of 99% for the classification of VPN traffic but achieved a lower accuracy of 86.6% for the classification of non-VPN traffic [6], this shows that the model has poor generalization strength and may have overfitted with the VPN traffic.

When deep learning techniques are used for specific tasks, the tendency of the model overfitting rises. This is because models are usually trained to fit a particularly small set of data. This creates a generalization problem as models tend to give uneven values of accuracy when tested with unseen data. Lu et al [8], used both CNN and LSTM for local and temporal feature extraction respectively. Both models were arranged in parallel, enabling features to be extracted and trained simultaneously, concatenated, then classified. They achieved an overall accuracy of 98.1%. The complexity of their model is a drawback as latency can become an issue if implemented for real-time classification. Hu et al [9], also used CNN and LSTM but arranged both models sequentially. The output of the LSTM is fed as input to the CNN and classification is done. Their model utilizes a squeeze and excitation (SE) mechanism. The SE operates using attention mechanism, where the most effective feature map has a higher weight, and the less effective feature map has a lesser weight. This makes the model biased toward high frequency traffic which can easily lead to overfitting. Bayat et al [10] used CNN and GRU (gated recurrent unit) to classify traffic. They classified traffic according to server name indication (SNI), reaching an overall accuracy of 82.3%. Which is relatively low.

Shapira et al [5] used only CNN to classify internet traffic. They converted the extracted features (packet size and arrival time) into images of traffic flows. The generated images were then fed to a simple CNN model for the task of classification. Their model produced accuracies ranging from 67.8% - 98.4%. A high point of their design is its ability to classify unknown traffic. In our paper, we adopt their [5] image generation method and use a deeper model to improve classification generalization strength. Vu et al [4] used LSTM for traffic classification by analysing internet traffic in time series. They combined both the payload and flow-based technique to properly represent the behaviour of traffic. They extracted seven features from packets to complete this task. Source port, destination port and protocol were used to identify the direction of a flow. Data length was used to represent unique application traffic. The last three features TCP/UDP header, application header and application data (all represented with byte values) made up the IP payload. They achieved a F_1 score of 98% not considering accuracy as a metric.

Recent papers on traffic classification laid more emphasis on model design than the data to be analysed. Datasets used for model training were imbalanced and assessing a model with such data will have a direct impact on accuracy. It is therefore important to properly augment dataset before the task of classification is performed, this will show the true performance of the model across all classification tasks. Other gaps like model simplicity leading to poor generalisation (especially for low frequency data), overfitting and poor granularity are issues with current designs.

After reviewing recent works that addressed a similar topic, we developed a new model. The proposed architecture uses the flowpic [5] method to generate traffic images, then combines the LeNet5 [11] and VGG16 [12] model to develop a deeper classifier. Distinctive layers are added to make the algorithm deeper and curb the tendency of the model overfitting.

Our goal is to develop a model with a simple design that has optimal generalization strength. When developing a deep learning model, the task the model aims to perform should be a deciding factor in determining how deep the model should be. The traffic classification problem as defined by [5] has been reduced to an image classification problem. This is to properly visualise traffic behaviour for proper classification. The image generation process as outlined in Figure 1 involves extracting the metadata from every packet, preparing the data which involves data wrangling and features scaling, then generating images corresponding to traffic flows. The process is defined in-depth in section 4.2. We design our CNN model as described in Figure 2.

3 Proposed Architecture

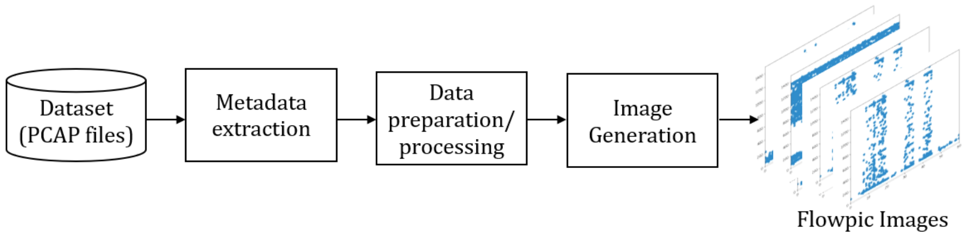


Figure 1: Image generation process

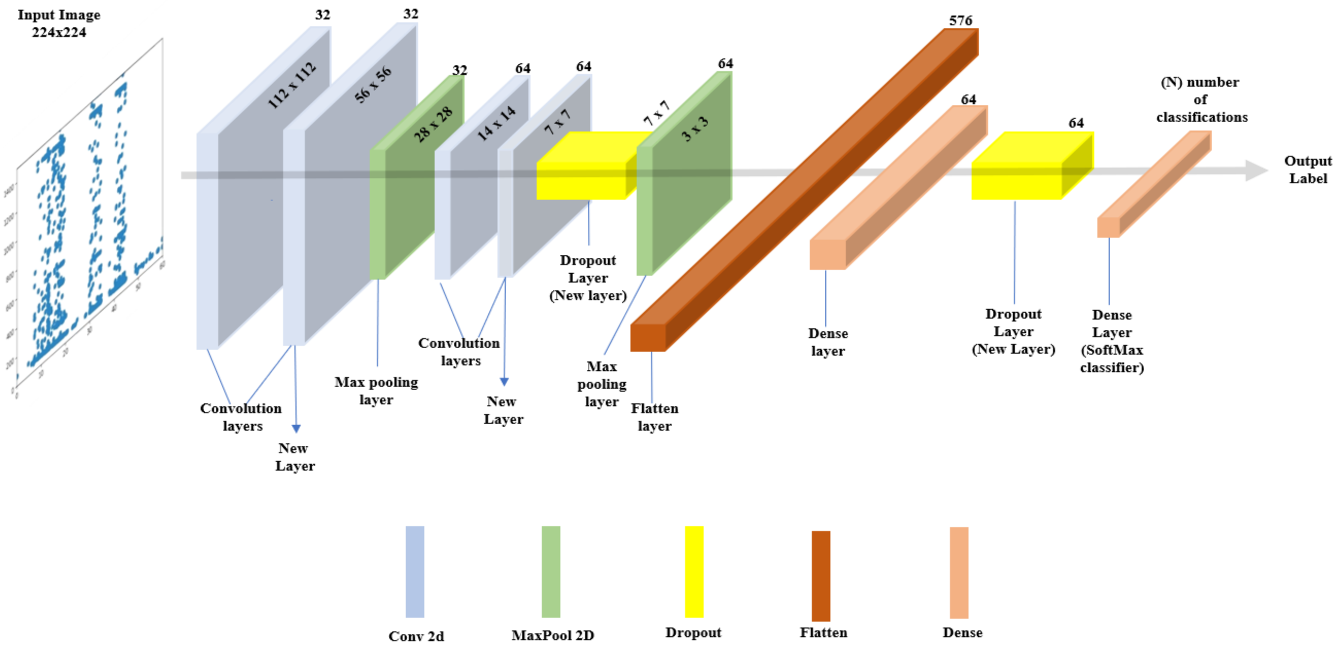


Figure 2: Proposed Convolutional Neural Network based classifier

The model has a total of eleven layers which consists of, four convolution layers, two max pooling layers, two dropout layers, one flatten layer and two dense layers (one used as the soft max classifier). New layers are labelled for clarity. The model takes input images of size 224x224, hence we process our images as such. The size was chosen based on empirical study done on image classification. Richer et al [13] conducted experiments with different image sizes (32x32, 224x224, and 1024x1024) to ascertain the effect of image size on a model's accuracy. When the size was reduced to 32x32 the model's accuracy dropped by 8.78% and when it was increased to 1024x1024 the model's accuracy dropped by 6.49%. The model achieved its best accuracy when trained with 224x224 sized images. We also adopt this format because the renowned VGG16 model [12] processed images in this format and used them for classification task. All the hidden layers are equipped with the rectified linear unit (ReLU) non-linearity.

The first two layers are a stack of convolutional layers. The intuition behind stacking two convolutional layers is because the next layer is a pooling layer which decreases the image dimension exponentially. The pooling action, while decreasing dimensionality causes loss of spatial information. It is therefore important to build up better representation of the images before passing it through a pooling layer. Zisserman & Simonyan [12] demonstrated the advantages of stacking multiple convolutional layers without the pooling in-between. By this practice, they were able to increase the effective receptive fields and make the decision function more discriminative. A kernel size of 3x3 with a stride of 2 is used to perform the convolution operation on all layers. Choosing a kernel size with a small receptive field is important to capture features from all notions. The same kernel size and stride was used across all convolutional layers in the model. After taking an input image of 224x224 the first convolutional layer gives an output image of 112x112. A channel of 32 is chosen for the first two convolutional layers for the model to learn properly. The intuition behind the channel size is efficiency and memory. Literatures like Mao et al [14] used more channels starting with 128, empirically this didn't have much effect on the model's accuracy as image dimensions kept changing after each layer. Using a large number of channels can cause a model to overfit. The second convolution layer takes the output of the first as its input. Using the same filter, kernel size, stride, and activation (ReLU) it gives an output shape of 56x56.

The next layer is the pooling layer, a max pool operation is performed on this layer with a kernel of size 2x2 and stride of 2. An output shape of 28x28 is gotten from this layer. Max pooling is used to preserve the effective receptive field of the previous layer. The choice of a max pool operation amongst other options is due to its ability to adapt to features with low probability of activation, Boureau et al [15] demonstrated the recognition performance of pooling operations and max pooling stood out as the best technique.

The fourth and fifth layers are a stack of convolution layers with an increased channel size of 64. The increment in channel size is because of the reduction in output shape, more channels will be needed to increase the learning capacity of the model. An increment by a multiple of 2 was chosen based on empirical study conducted, Zisserman & Simonyan [12] increased their channels from 64 till it reached 512. The fourth layer gives an output shape of 14x14 while the fifth layer gives an output shape of 7x7. The reduction in output shape is a direct function of the kernel used for convolution. By sliding the kernel across the image array an output image of a smaller dimension is obtained.

A dropout layer with rate 0.25 is introduced as the sixth layer. This is to tackle the problem of over fitting as discussed in the previous section. The rate of 0.25 was chosen because we want our model to retain as many features as possible. Wu & Gu [16] conducted experiments placing dropout layers at unique positions in their design. They achieved the lowest error when dropout was placed after the max pooling and fully connected layers. In our model we use dropout after the second convolution stack and second fully connected layer, as this is where we achieved the lowest loss and highest accuracy from our experiments.

The seventh layer is another max pooling layer with kernel size of 2x2 and stride of 2, the addition of this layer is to give depth to the model and improve its learning capability. The eighth layer is a flatten layer which converts the pooled feature map into a single column. It converts the mapped features into a one-dimensional (1D) array. The next layer is a fully connected layer which is simply a feed forward neural network. This is fed to a second dropout layer with rate 0.5 to check the model for overfitting again. The final layer is a SoftMax layer which is edited based on the defined number of classifications required.

The proposed model was designed after an empirical review of well-known image classification models was carried out. The VGG I6 and flowpic model [5], [12] became the baseline for our design after this review. The flowpic model [5] was selected because of its ability to classify traffic with high accuracy while having a simple design, The LeNet5 [11] design. Accuracies of 98.4% and 99.7% were attained for VPN traffic class and application identification tasks respectively. This implies that making the model deeper will improve the classification accuracy and generalization strength. The model can also classify unknown applications. The VGG16 model [12] designed by Zisserman & Simonyan was one of the best performing models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014 [24]. The model achieved an accuracy of 92.7% on the popular ImageNet dataset which contains 14,197,122 million images of 21,841 categories. The VGG 16 model stacks its convolution layers and uses a 3 x 3 kernel size for the convolution operation. We combine the deep structure of the VGG16 model and the simple structure of Shapira's design to build our CNN model.

While designing our model we took into consideration the limitations of the baseline models. The VGG16 [12] model, which is well known for its classification strength, suffers some major drawbacks: it is very slow to train and has a very large architecture of over 530MB. Shapira & Shavitt's [5] model is not deep enough and possesses low generalization strength.

4 Experimental Setup and Methodology

The experimentation process is divided into four major parts, as follows. The datasets, Data preparation and processing, Implementation and training, Performance Evaluation.

4.1 The Datasets

The University of New Brunswick (UNB) developed numerous datasets for different tasks of traffic classification. This dataset is made public and has been utilized by research universities and organizations for numerous research purposes. In this paper we used two of their datasets VPN-nonVPN traffic dataset (ISCXVPN2016) [27] and Tor-nonTor dataset (ISCXTor2016) [28].

The VPN-nonVPN traffic dataset [27] is currently one of the most popular datasets for traffic classifications tasks. Reviewed literatures like [4], [5] [7], [8], [9] used this dataset for traffic classification task. It is also used for comparative analysis. To generate this dataset the UNB Canadian Institute for Cybersecurity team, created accounts for Alice and Bob to use services (applications). A summary of the traffic created is described below in **Table 1**.

The Tor-nonTor dataset [28] is also a popular dataset used for traffic classification task. Recent papers like [17], [18] used this dataset for encrypted traffic analysis. The traffic categories for the (ISCXVPN2016) and (ISCXTor2016) dataset are the same, hence **Table 1** can be used to describe both datasets. The datasets are 28gb and 22gb respectively. Traffic for Bit torrent, Google, Twitter, FTPS, ICQ were low below 1000 samples, and we had to augment to make up. The files are in pcap format.

Table 1: Dataset Summary

Traffic Category	Content
Browsing	Firefox and Chrome
Email	SMTPS, POP3S and IMAPS
Chat	ICQ, AIM, Skype, Facebook, and Hangouts
Streaming (Audio)	Spotify
Streaming (Video)	Vimeo and YouTube
File Transfer	Skype, FTPS and SFTP
VoIP	Facebook, Skype, and Hangouts
P2P (Transfer)	uTorrent and Transmission (BitTorrent)
VPN (All)	Same as above but captured using VPN

4.2 Data Preparation and Processing

The goal is to generate images from traffic flows with their corresponding labels. These images are then fed to a convolutional neural network for the task of classification. To generate images, we extracted two features from each packet (packet size and packet arrival time). In wireshark the columns *frame.len* and *frame.time_relative* represents the extracted features.

To create images, each pcap file was split into window sizes of 15-, 30- and 60-seconds based on the *frame.time_relative*. We used different window sizes to have abundance of image samples for augmentation and to address the problem of traffic arrival time disparity. Traffic arrival time disparity happens when no packet arrives within the selected window leading to a blank image **Figure 3**c.
The packet size was capped at 1500 bytes (maximum transmission unit), packets greater than 1500 bytes were not considered during analysis. The extracted features were converted to csv format. The values were plotted on a scatter plot, the y-axis signified *frame.len* and x-axis signified *frame.time_relative*. The y-axis was capped at 1500 and x-axis was capped based on the time window size used (15, 30, 60secs).

The resulting images were saved as JPEG files. The images generated from this process are called flowpics [5]. Images were classified into three encryption categories, https (Non-VPN), VPN and TOR. For each encryption category the images were further classified into application type (e.g., Facebook Netflix, skype etc.) and traffic type (e.g., Video, Chat, Audio etc.).

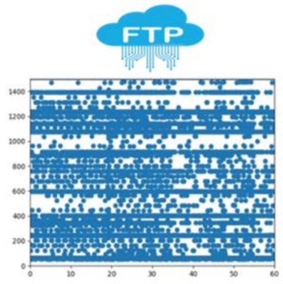
Data augmentation was heavily applied to the images to balance the data for the training and testing phase. Shorten & Khoshgoftaar (2019) conducted a survey of data augmentation techniques for DL, they achieved a higher accuracy when geometric transformation techniques like, flipping, cropping, rotating, colour space transformation was used to augment image data [25]. Hence, we use image rotation and flipping as augmentation techniques.
The images are further processed to 224x224 pixel images on Keras before being fed to the model for training. Table 2 shows the total number of images for each window size.

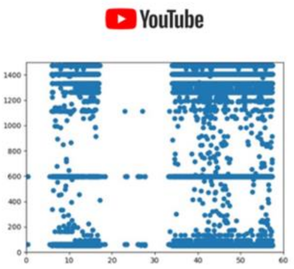
Table 2: Number of images generated for different time windows

Datasets	Images Created		
	60 Secs	30 Secs	15 Secs
VPN-nonVPN	2693	5386	10772
TOR	2482	4964	9928

Figure 3a-c shows three sample of traffic flow images (flowpics) 3a is a flowpic for FTP when the TOR browser is used and 3b is a YouTube flowpic. **Figure 3**c shows the

problem of traffic arrival time disparity, leading to an almost blank flowpic.


(a) Tor: File Transfer FTP


(b) Tor: YouTube

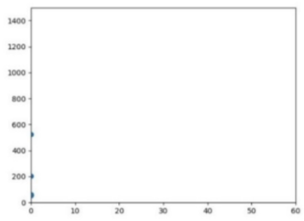

(c) nonVPN: email (Demonstrating a critical problem of traffic arrival time disparity leading to an almost blank flowpic

Figure 3: Traffic flow images (Flowpic)

4.3 Implementation and Training

The model was trained for seven different experiments (Identification problems) as show in **Table 3** below. Applications classes and Traffic type classes are defined in Section 4.1 as contents (Facebook, twitter etc) and categories.

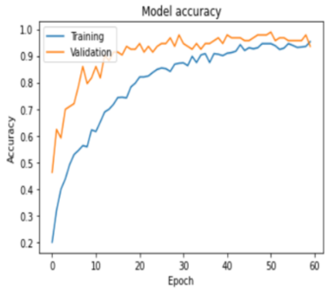
Table 3: Experiment Content

Experiment	Description
1	non-VPN applications Identification
2	non-VPN Traffic Type Identification
3	VPN Application Identification
4	VPN Traffic type Identification
5	TOR Application Identification
6	TOR Traffic type Identification
7	Encryption Type Identification

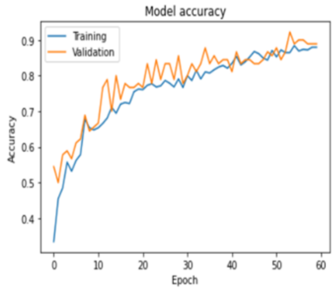
The following specifications are used for training. Implementation and experiment configuration are done with Keras API [29] using TensorFlow [21] (python 3.8.8) as its backend. The environment is a windows 10 pro-64-bit operation system the processor is an Intel (R) Core (TM) i5-6300U CPU @ 2.40GHz 2.50 GHz, 8 GB ram size. For all experiment conducted we divided the data (images) into three sets, 80% training set, 10% validation set and 10% testing set.

The hyperparameters of the model are as follows: batch size is 10, a small batch size is used due to storage limitation, epoch is 60, Adam optimizer is used to improve the categorical cross entropy loss function with learning parameters of 0.0001 learning rate. The decay rates β_1, β_2 and $\in$ are set to their default values of 0.9, 0.999 and 10^{-8} respectively. We used the same training specifications for all experiment.

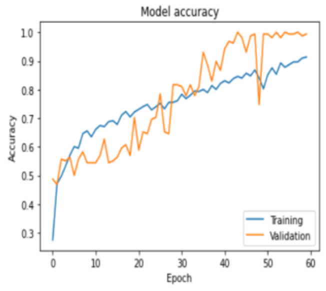
Figure 4a-g Shows the training and validation accuracy curves for all seven experiments



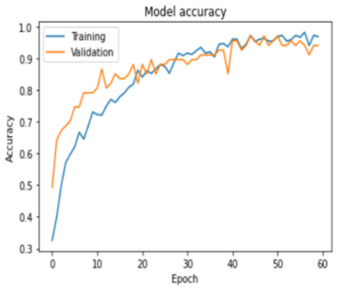
(a) non-VPN applications Identification



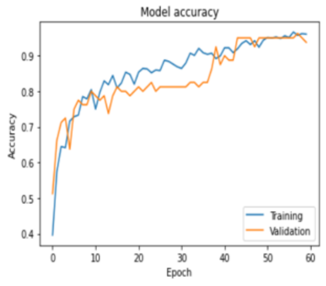
(b) non-VPN Traffic Type Identification



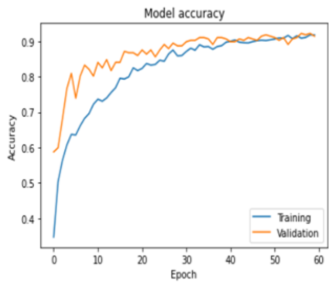
(c) VPN Applications Identification



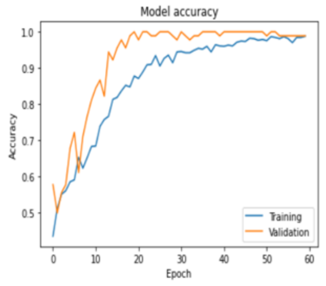
(d) VPN Traffic Type Identification



(e) TOR Applications Identification



(f) TOR Traffic Type Identification



(g) Encryption Type Identification

Figure 4: Training and validation curves of experiments

From **Figure 4a-g** the curves show gradual increment in accuracies for all experiments. For most of the experiments, the validation accuracy graduated faster than the training accuracy due to the difference in sample size (80% training set, 10% validation/testing sets). DL models learn faster but not properly when the dataset is small, this does not affect the training process. Both training and validation curves have similar convergence properties, however in **Figure 4**c the training accuracy graduated slowly but didn't converge when trained for 60 epochs, it converged when the epoch was within 100 – 200. Too many epochs can cause overfitting.

4.4 Performance Evaluation

To properly assess the usefulness of the developed model, its performance was evaluated. Performance evaluation is an important aspect of deep leaning, it is necessary for trusting a model as it gives an unbiased analysis of the model. The evaluation metrics used for our model are defined below:

Classification Accuracy: Accuracy can be defined as the total number of predictions a model gets correct against the total number of predictions in the model. Mathematically it can be defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{1}$$

Accuracy can also be calculated in terms of positives and negatives, as define with the formular below:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

Precision: This can also be called the positive predictive value; it is used to find out the proportion of positive identifications that are correct. It is defined as the ratio of relevant instances amongst all instances received. It can be

$$Precision = \frac{TP}{TP+FP} \tag{3}$$

Recall: This is used to find the proportion of actual positives that was identified correctly. It can be defined as the ratio of the number of correct results to the number of results that should have been returned. Recall can also be known as sensitivity, hit rate or true positive rate.

$$Recall = \frac{TP}{TP+FN} \tag{4}$$

$\boldsymbol{F_1}$**-Score**: This is used to assess the accuracy of a model on a dataset. It is called the harmonic mean as it combines the recall and precision scores of a model. It is used to give a picture of the model's true performance. Mathematically it can be defined as:

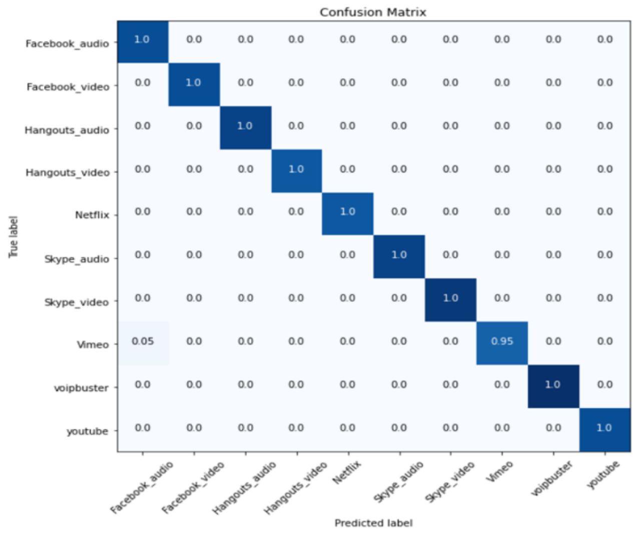
$$F_1\ Score = \frac{Precision*Recall}{Precision+Recall} \tag{5}$$

Confusion Matrix: This is a technique used for summarizing the general performance of a model. It shows how the classification model is confused when it makes prediction, by giving a table of true labels against predicted labels. It gives the count of correct and incorrect classifications in normalized or non-normalized form.

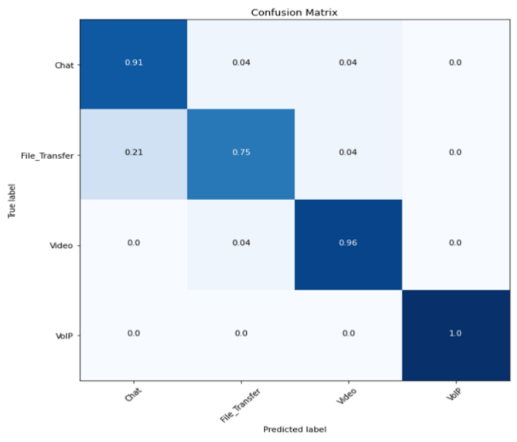
5 Results

After training the model across all seven experiments. We tested the model using the standard metrics discussed in section 4.4.

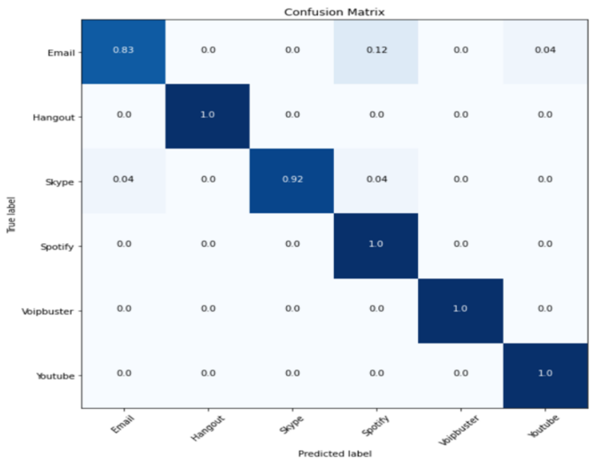
Figure 5a-g shows the confusion matrix of the test carried out. The matrix is used to show how well our model matches an input (image) to its correct label. It shows how well the model predicts and where it gets confused.



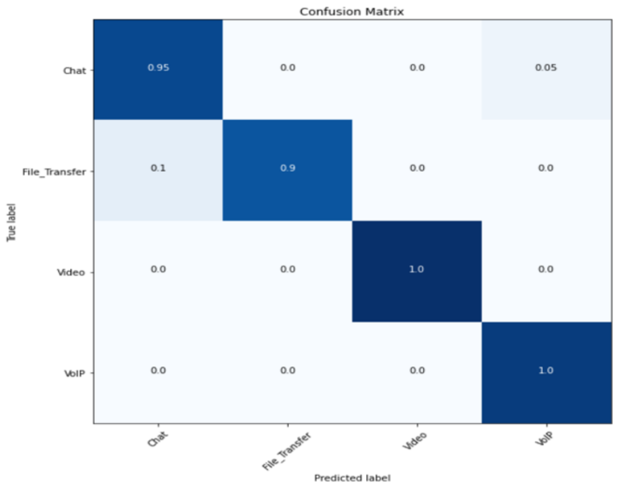
(a) Non-VPN applications Identification



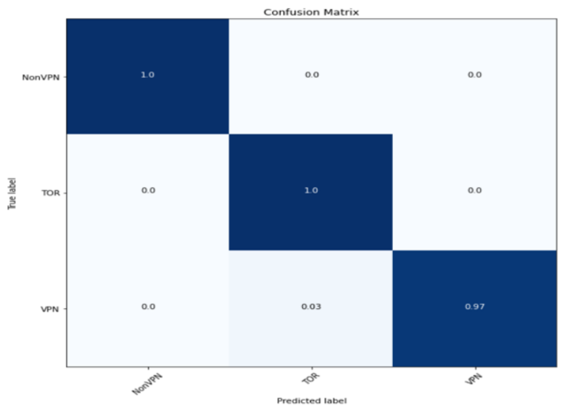
(b) Non-VPN Traffic Type Identification



(c) VPN Application Identification

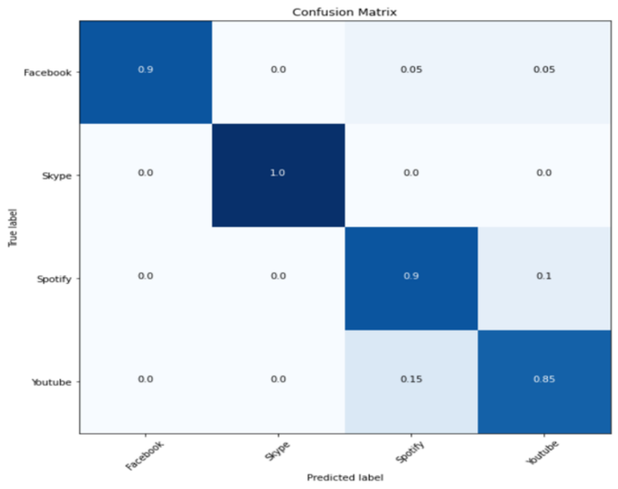


(d) VPN Traffic Type Identification

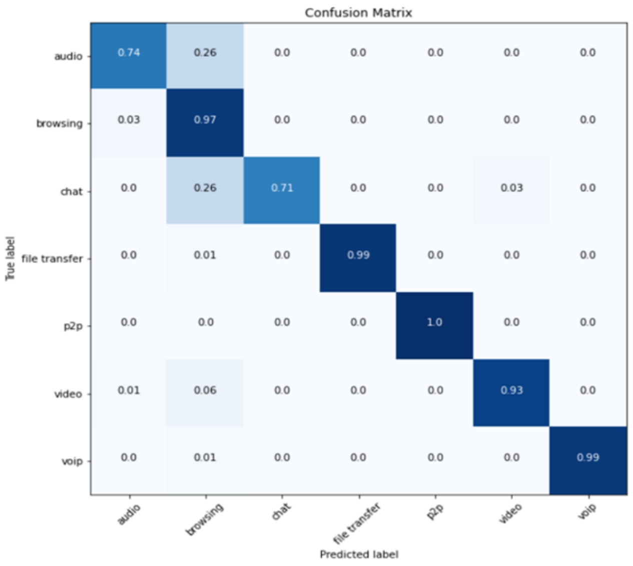


(g) Encryption Type Identification

Figure 5: Confusion matrix of all experiments during model testing



(e) TOR Applications Identification



(f) TOR Traffic Type Identification

From **Figure 5**a-g our model shows good classification strength for the range of experiments conducted, however for some identification tasks it gave wrong predictions.
The anomalies observed were found to be directly related to the imbalanced nature of the dataset. From **Figure 5** The model couldn't correctly classify "File_Transfer" images for non-VPN Traffic identification (5b). Email application images were not correctly classified for VPN application identification (5c). For TOR traffic identification the model couldn't accurately classify audio and chat traffic types (5f). Not every application class and traffic type were considered during the training and testing phase because of data imbalance. Augmentation was heavily implemented to supplement low frequency data. After augmentation, classes with low image samples (less than 150) were dropped. The traffic class "browsing" was excluded when analysing the nonVPN and VPN application and traffic type classification because it wasn't properly defined by the
originators of the dataset.

Relatively, our model performed well as no value is below the 91% mark. Table 4 shows the overall performance summary of our model based on the standard metrics. The average precision is 94.86% and values for recall and F_1 Score are 94.29%. This interprets to an overall good generalization strength when compared to other literatures as seen in the next section.

6 Discussion

To further analyse the performance of our model, we compare our results with other literatures using [5] as the baseline model. Table 5 shows the summary of comparisons. In comparing our work to others, we use accuracy as the metric. This is because accuracy is the generally accepted metric for assessing DL and ML models. Although, the flowpic [5] model performed better in two experiments, it gives inconsistent values for accuracy across all the conducted experiment.

Table 4: Overall performance summary

PROBLEM	Accuracy	Precision	Recall	F_1 Score
non-VPN Application Identification	93%	97%	96%	96%
non-VPN Traffic Type Identification	91%	91%	91%	91%
VPN Application Identification	96%	96%	96%	96%
VPN Traffic Type Identification	96%	96%	96%	96%
TOR Application Identification	91%	92%	91%	91%
TOR Traffic Type Identification	91%	93%	91%	91%
Encryption Type Identification	99%	99%	99%	99%
Average	**93.86%**	**94.86%**	**94.29%**	**94.29%**

Table 5:Result summary comparison

	Accuracy		
PROBLEM	**Our Model**	**Flowpic [5]**	**Other work**
Non-VPN Applications Identification	93%	99.7%	93.9 % [19]
Non-VPN Traffic Type Identification	91%	85.0%	84.0 % [20]
VPN Applications Identification	96%	-	-
VPN Traffic Type Identification	96%	98.4%	98.6% [6]
TOR Applications Identification	91%	-	-
TOR Traffic Type Identification	91%	67.8%	84.3 % [20]
Encryption Type Identification	99%	88.4%	99. % [6]

For the task of non-VPN traffic type identification, our model outperformed the baseline model by 6%, ending up with an accuracy of 91%. For the task of non-VPN application identification, the baseline model outperformed our model. We ended up with an accuracy of 93% while [5] ended up with 99.7%. This result was unexpected but can be associated with the imbalanced nature of the dataset.

For VPN traffic type identification, the "browsing" class was dropped due to its poor definition. The baseline model once again outperformed the proposed model. Our model achieved an accuracy of 96% while the baseline model achieved an accuracy of 98.6%. For the TOR traffic type and application identification. Our model outperformed the baseline model, reaching an accuracy of 91% for both experiments. The proposed model showed good generalization strength across all experiments. When the weighted average accuracy is calculated, the model has an overall 93.86%. The weighted accuracy can be misleading as one cannot properly assess the model's strength or weakness. It is therefore important to consider individual accuracies along with precision, recall and F_1 score (Table 4) when assessing a model's classification strength.

One of the major problems encountered was the issue of having an imbalanced dataset. While this didn't directly affect the training model because of the augmentation techniques utilized. It affected the validation accuracy in some experiments, e.g., in interpreting the validation accuracy curve for VPN application identification task in **Figure 4**c. The model shows overfitting tendencies and is slightly biased towards the training data. A similar issue was noticed for Tor application classification task **Figure 4**e. The problem of traffic arrival time disparity as discussed in section 4.2 can also be detrimental to the learning process of the model because it reduces the learnable features of the images. This goes to show that a more systematic feature extraction process will be needed to improve the current design. From the training/validation curves for loss and accuracy, our model showed a gradual rise in accuracy and drop in loss for all identification tasks. When we introduced the test (unseen) data the accuracies did not drop below 91% across all experiments conducted. Overfitting occurs when a model achieves a good fit "only" for the training data i.e., the model is heavily biased towards the training data that it does not perform optimally when a test (unseen) data is introduced. By this definition, our model does not overfit.

Overall, our model does a better job of generalization as values for accuracy remain stable (within the 90th percentile). The proposed model performed well across all experiment with the accuracy not dropping below 91%.

7 Conclusion

This work presents a convolutional neural network (CNN) based classifier for the problem of encrypted traffic identification. One major issue facing encrypted network traffic is the problem of accurate traffic classification. This issue as discussed previously, can cause a ripple effect of other cybersecurity problems. The CNN-based classifier is used to classify encrypted internet traffic based on specific tasks. An existing design based on the popular LeNet-5 architecture was modified and trained on a range of datasets. The extracted features of packet size which is important for defining applications, and packet arrival time which defines temporal features of a network packet was used to generate unique images for different application classes. After the images were generated, they were fed to the designed CNN classifier for seven (7) different classification tasks. The proposed model gave the best result when identifying different encryption types (99%). It achieved its worst results of 91% for the tasks of non-VPN (Traffic type), TOR (Application) and TOR (Traffic type) Identification. Overall, our model achieved its aim based on the results gotten, the proposed model was found to have good generalization strength and didn't overfit or underfit for any task. This design can serve as a baseline for security researchers when building a classifier, thereby reducing the initial burden of selecting a model or algorithm for the work.

To improve the current or similar design. The feature extraction process needs to be more systematic from the initial stages of data processing to the final stages of model testing. Data augmentation also needs to be systematic to deal with low frequency traffic. Finally, classifiers should be trained on more applications and encryption range as this will improve granularity and generalization strength.

REFERENCES

[1] Google Transparency Report, (2021). HTTPS encryption on the web. Google. https://transparencyreport.google.com/https?hl=en

[2] Nie, S., Jiang, L., Sun, H., Ma, C., Liang, Y., Zhou, Y., & Zuo, Y. (2020). Network traffic classification model based on multi-task learning. Journal of Physics. Conference Series, (1693) 012097. (p 2) http://doi:10.1088/1742-6596/1693/1/012097

[3] Australian Cyber Security Centre (ACSC). Annual Cyber Threat Report 2020-21. Retrieved October 22, 2021, from https://www.cyber.gov.au/acsc/view-all-content/reports-andstatistics/acsc-annual-cyber-threat-report-2020-21

[4] Vu, L., Thuy, H. V., Nguyen, Q. U., Ngoc, T. N., Nguyen, D. N., Hoang, D. T., & Dutkiewicz, E. (2018). Time series analysis for encrypted traffic classification: A deep learning approach. 2018 18th International Symposium on Communications and Information Technologies (ISCIT), (pp 121–126). https://doi: 10.1109/iscit.2018.8587975

[5] Shapira, T., & Shavitt, Y. (2019). FlowPic: Encrypted internet traffic classification is as easy as image recognition. IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), (pp 680–687). https://doi: 10.1109/infcomw.2019.8845315

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

[7] Rasteh, A., Delpech, F., Aguilar-Melchor, C., Zimmer, R., Shouraki, S. B., & Masquelier, T. (2021). Encrypted Internet traffic classification using a supervised Spiking Neural Network. In arXiv [cs.LG]. (pp 5–19). http://arxiv.org/abs/2101.09818

[8] Lu, B., Luktarhan, N., Ding, C., & Zhang, W. (2021). ICLSTM: Encrypted traffic service identification based on Inception-LSTM neural network. Symmetry, 13(6), 1080. (pp 4–14). https:// doi.org/10.3390/sym13061080

[9] Hu, F., Zhang, S., Lin, X., Wu, L., Liao, N., & Song, Y. (2021). Network traffic classification model based on attention mechanism and spatiotemporal features. In Research Square. (pp 7–28). https://doi.org/10.21203/rs.3.rs-353938/v1

[10] Bayat, N., Jackson, W., & Liu, D. (2021). Deep learning for network traffic classification. In arXiv [cs.NI]. 1-10. http://arxiv.org/abs/2106.12693

[11] LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., & Henderson, D. (1989). Handwritten digit recognition with a backpropagation network. In Advances in neural information processing systems (NIPS) 2. 396–404. Morgan Kaufmann. http://doi/10.5555/109230.109279

[12] Simonyan, K. and Zisserman, A. (2015) Very Deep Convolutional Networks for Large-Scale Image Recognition. The 3rd International Conference on Learning Representations (ICLR2015), 3-6. https://arxiv.org/abs/1409.1556

[13] Richter, M.L., Byttner, W., Krumnack, U., Schallner, L., & Shenk, J. (2021). Size Matters. 4-6. ArXiv, abs/2102.01582.

[14] Mao, J., Zhang, M., Chen, M., Chen, L., Xia, F. et al. (2021). Semi supervised Encrypted Traffic Identification Based on Auxiliary Classification Generative Adversarial Network. Computer Systems Science and Engineering, 39(3), 373–390. http://doi:10.32604/csse.2021.018086

[15] Boureau, Y., Ponce, J., Lecu, Y., (2010). A Theoretical Analysis of Feature Pooling in Visual Recognition. Conference: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 6-7. https://dl.acm.org/doi/10.5555/3104322.3104338

[16] Wu, H., and Gu, X., (2015). Towards Dropout Training for Convolutional Neural Networks. Neural Networks (71), 8-10. Doi:10.1016/j.neunet.2015.07.007

[17] VishnuPriya., Singh, H. K., SivaChaitanyaPrasad., & JaiSivaSai. (2021). RNN-LSTM based deep learning model for tor traffic classification. Cyber-Physical Systems, 1–18. https://doi.org/10.1080/23335777.2021.1924284

[18] Hodo, E., Bellekens, X., Iorkyase, E., Hamilton, A., Tachtatzis, C., & Atkinson, R. (2017). Machine learning approach for detection of nonTor traffic. Information 2018, 9(9), 231. http://arxiv.org/abs/1708.08725

[19] Yamansavascilar, B., Guvensan, M. A., Yavuz, A. G., & Karsligil, M. E. (2017). Application identification via network traffic classification. 2017 International Conference on Computing, Networking and Communications (ICNC), 843–848. http://doi:10.1109/iccnc.2017.7876241

[20] Draper-Gil, G., Lashkari, A. H., Mamun, M. S. I., and Ghorbani, A. A. (2016). "Characterization of encrypted and vpn traffic using time-related features," in Proceedings of the 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,, INSTICC. SciTePress, 2016, pp. 407–414.

[21] TensorFlow. (n.d.). Tensorflow.Org. Retrieved October 18, 2021, from https://www.tensorflow.org/

[22] WatchGuard threat lab reports 91.5% of malware arrived over encrypted connections in Q2 2021. (n.d.). Watchguard.Com; WatchGuard Technologies. Retrieved November 17, 2021, from https://www.watchguard.com/wgrd-news/press-releases/watchguardthreat-lab-reports-915-malware-arrived-over-encrypted

[23] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. Adaptive computation and machine learning. MIT Press. ISBN: 9780262035613

[24] ImageNet. (n.d.). Image-Net.Org. Retrieved November 17, 2021, from https://www.image-net.org/challenges/LSVRC/2014/

[25] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). 7-11 .https://doi.org/10.1186/s40537-019-0197-0

[26] Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. In Preprints. https://doi.org/10.20944/preprints202108.0060.v1

[27] UNB, 2016. www.unb.ca.(n.d.). VPN 2016 UNB. http://www.unb.ca/cic/datasets/vpn.html

[28] UNB, 2017. www.unb.ca.(n.d.). Tor 2017. http://www.unb.ca/cic/datasets/tor.html

[29] Chollet, F., et al., (2015). Keras: Deep Learning for humans. (n.d.). https://github.com/fchollet/ker