CNN for Network Intrusion Detection

Anthony Orlowski
Indiana University

Abstract— This study partially replicates experiments found in the paper “Malware traffic classification using convolutional neural network for representation learning” [1] which explores the idea of using CNN models to classify network traffic that is represented as gray scale images. This study explores two classification problems including a two-class and twenty-class classification problem, as well as four distinct representations of the network traffic including: using all network traffic header and payload data, using only payload data, using a session representation of the traffic, and using a flow representation of the traffic. We provide an updated and independent implementation and and analysis of our results compared to those found in [1]. The results (98.81% accuracy for the best model) show that CNN models are a promising method to learn a classification function from raw network traffic and thus show utility in network intrusion detection system implementation.

Index Terms—Convolutional Nural Network, CNN, Network Intrusion Detection, NIDS, Image Recognition, Deep Learning

Introduction

Organizations use digital communication and the internet as key mechanisms to achieve their goals. However, this exposes their networks, their data, and the data of their customers to the intrusive behaviours of malware and internet attackers. Network intrusion detection systems (NIDS) protect the confidentiality, integrity, and authenticity of organisations’ networks and data by serving as a sentinel to detect unauthorized access or abuse. Security personnel rely on these alerts to take remedial action to prevent damage or loss. NIDS are a well-studied field but are continually evolving to meet the increasing sophistication of network attacks and to adopt advances in machine learning.

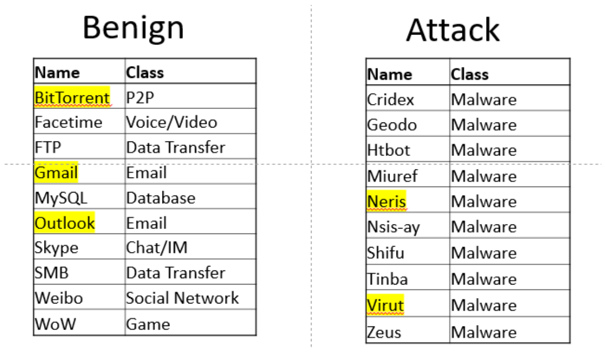
There are multiple types of NIDS including rule based, statistical feature based, and deep learning based. Rule based models such as Snort [2], Suricata [3], and firewalls use manually crafted rules to classify or authorize network traffic. Statistical feature based NIDS, such as those based on Zeek [4] or CIC-Flow-Meter [5] compute features from network traffic and then use classic machine learning models (such as random forests or support vector machines) or deep learning models to classify traffic based on those features. Deep learning based methods learn to classify from only the raw labelled network traffic. This is an improvement from the rule based and statistical feature based systems that require the time consuming or computationally expensive processes of manual rule creation or feature set computation.

In this study, we replicate some of the ideas and convolutional neural networks (CNN) first published in [1], provide our independent analysis of our results, and compare our results to the original paper. We study two classification problems and four different representations of network traffic provided by the USC-TFC2016 dataset (from [1]) to train CNN models. In our Github repository [8] we contribute updated code that works with Python 3 and Pytorch compared to the implementation from [1] that uses Python 2.7 and Tensorflow v1.

1. Method: Data Representation

We solve two different classification problems; a two-class problem that labels traffic as “attack” or “benign”; and a twenty-class problem that labels the distinct classes within those two categories. We expect the two-class models to provide more accurate results than the twenty-class models. Figure 1. presents the two and twenty class view of the classification problem. Highlighted in yellow are the sub-classes that we identified the CNN models have the most trouble classifying.

We also study four different representations of the raw network traffic to determine which combinations provide the most accurate results. As seen in Figure 2. we first compare a variation of the amount of header data kept from the raw network traffic including


Benign

Name	Class
BitTorrent	P2P
Facetime	Voice/Video
FTP	Data Transfer
Gmail	Email
MySQL	Database
Outlook	Email
Skype	Chat/IM
SMB	Data Transfer
Weibo	Social Network
WoW	Game

Attack

Name	Class
Cridex	Malware
Geodo	Malware
Htbot	Malware
Miuref	Malware
Neris	Malware
Nsis-ay	Malware
Shifu	Malware
Tinba	Malware
Virut	Malware
Zeus	Malware

Figure 1. Two (benign vs attack) vs twenty class classification.

models that include all-layer information and models trained only on the data payload (layer 7). We expect the all-layer representation to provide more accurate results than the layer 7 representation as it provides more information for the model to learn on. However, it is possible that using these extra layers may cause the model to overfit to the internet network topology where the data was collected and therefore produce models that may not generalize well for use on other networks. So, we investigate the layer-7 only representation of the data to limit the amount of network topology specific information contained in the model. If layer-7 models prove to be close in accuracy to all-layer models, then they may prove to be better in general use.

It is important to note, that even if MAC (layer 2) and IP (layer 3) layers are included, those addresses are sanitized in the provided

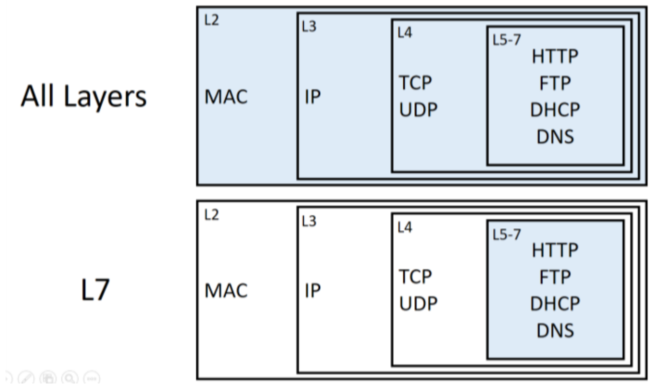


Figure 2. All layer vs layer 7 only data representation

dataset. This is because those addresses uniquely identify a machine, and thus uniquely identify an "attacker", so leaving in these features is akin leaving the label attached to the training data. However, we keep those layers in the all-layer representation as there are other features such as flags, protocol identifiers, and lengths that are valuable information.

Next, we study the effect of representing the raw network traffic in flow or session form. Network flows represent a uni-directional flow of data from a client to a server with the same five-tuple (source IP, source port, destination IP, destination port, and protocol). The session representation is a bi-directional flow of data between a client and server. Flows and sessions are time dependent, thus multiple flows or sessions can have the same five-tuple because they are from a different communication period. It is important to note that we did not study models trained on individual packets, but this is a valid approach as seen in [6]. Figure 3. visually depicts these different data representations. Only the solid black arrows are considered in the specified representation. We expect the session representation of the data to create more accurate models as it provides more information in the form of server to client response data.

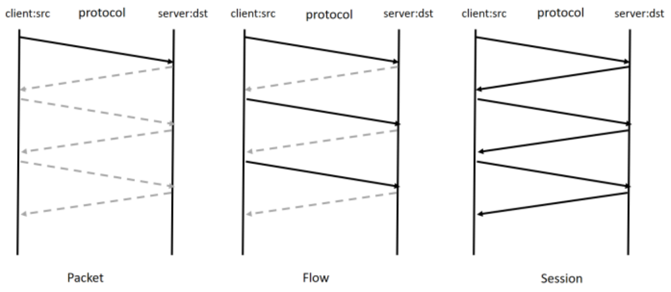


Figure 3. Packet vs. flow vs session representation of network traffic.

In totality, we create eight CNN models with the following features:

1. 2 class, Flow, L7-only
2. 2 class, Session, L7-only
3. 2 class, Flow, All-layer
4. 2 class, Session, All-layer
5. 20 class, Flow, L7-only
6. 20 class, Session, L7-only
7. 20 class, Flow, All-layer
8. 20 class, Session, All-layer

We compare the accuracy of these models and the precision, recall and F1 scores of the individual subclasses in the 20-class case.

2. Method: Data Set

The data set used is the USC-TFC2016 dataset described by [1] and found at [8]. It includes labelled network sessions and flows converted to IDX format images for the eight representation we present. The data is generated from ~3.7 GB of raw network traffic (pcap format) using tools found at [8]. The ten malware classes are real-world network traffic captured and compiled for this data set. The benign network classes were generated using the IXIA network simulator tool [9]. The number of total samples for each model range from 63120 to 138145. We use a 90% training and 10% testing split of the data.

3. Method: Image Representation of Network Traffic

The authors in [1] were inspired by CNNs for image recognition and thus decided to try an image representation of network traffic. The network flows and sessions are trimmed to 784 bytes or padded with x00 if they are under 784 bytes to create 28x28 grey scale images. These images serve as the input to our CNN models. In

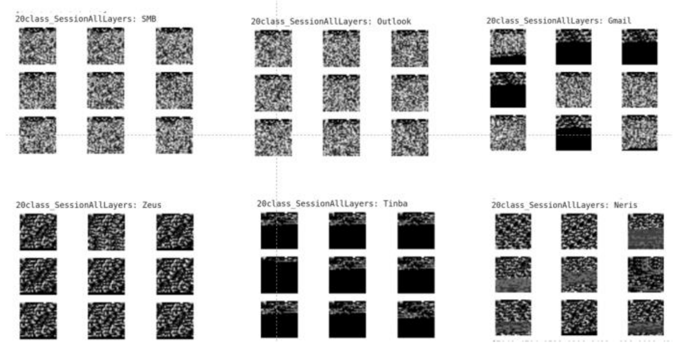

20class_SessionAllLayers: SMB
20class_SessionAllLayers: Outlook
20class_SessionAllLayers: Gmail
20class_SessionAllLayers: Zeus
20class_SessionAllLayers: Tinba
20class_SessionAllLayers: Neris

Figure 4. Example image representation of network traffic

Figure 4. we depict nine random samples of six example classes of these images from the 20-class-session-all-layer representation. We notice from inspection of these images that homogenous and often distinct texture patterns are observable in most of the classes leading us to believe that CNN is a suitable model for the task as it is efficient in image recognition. However, some classes, such as SMB and Outlook appear to the human eye as random noise. We determine that these seemingly similar classes are easily determined by the CNN. Finally, we see some classes such as Gmail and Neris that have multiple texture classes. We determine these are the classes that the CNN has the most trouble in distinguishing.

4. Experimental Setup: CNN Design

In Table 1. we describe our CNN design and experimental setup. We strove to keep a design similar to that as used in [1]. However, we modified the model parameters to attempt to reduce training time. The main differences include halving the number of filters, increasing the learning rate, and not including dropout. We also reduced the size of the final fully connected linear layers. Despite these changes we obtained results similar to those published in [1].

Table 1. CNN Design

Parameter	Our CNN	CNN from [1]
Layer 1	5x5 kernel, 16 filters + ReLu	5x5 kernel, 32 filters + ReLu
Layer 2	2x2 max pool	2x2 max pool
Layer 3	5x5 kernel, 32 filters + Relu	5x5 kernel, 64 filters + Relu
Layer 4	2x2 max pool	2x2 max pool
Layer 5-6	Flatten to fully connected: 50 node + Relu to 20 node	Flatten to fully connected: 1024 node + Reluto 20 node
Layer 7	Softmax (# classes)	Softmax (# classes)
Optimizer Function	Minibatch SGD, MB=50	Minibatch SGD, MB=50
Learning Rate	0.01	0.001
Loss Function	Cross entropy loss	Cross entropy loss
Number of Epochs	40	40
Includes Dropout	No	Yes
Environment	Pytorch	Tensorflow v1.0
Hardware	Google Colab Pro, GPU, Standard Shape	Dell R720, 16 core, 16GB, Nvidia Tesla K40m

5. Results

In this section, we provide our results and analysis of the eight various CNN models. We begin with a discussion on the training speed of the models, follow with a discussion of the model accuracy,

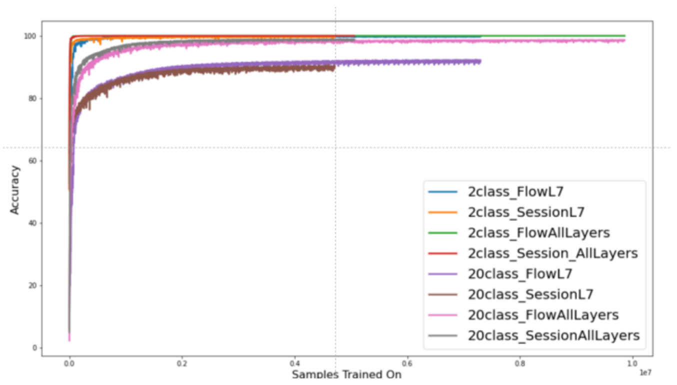


Figure 5. Training accuracy plot

and then consider the precision, recall and F1 score of the twenty class models.

5.1 Training Speed

In Figure 5. we provide a plot that shows the testing accuracy as the eight models are being trained. We can distinguish three distinct groupings of models in this plot. First, all of the two class models are shown to be trained faster and be more accurate than the twenty class models. Next, the twenty class models that use all layer information are shown to be more accurate than the twenty class models that use only layer 7 information but they train at about the same pace.

Further, our CNN model reduces the number of filters and fully connected nodes compared to that in [1] and increases the learning rate. Therefore, it is likely that our model trains faster due the reduced number of parameters and increased learning rate. Most of our models did not need the full 40 epochs the train.

5.2 Overall Model Accuracy

In Figure 6. we provide the final testing accuracy for the eight models. On the left side are the four two-class models, and on the right, the four twenty-class models. Those models that used the same layer information and data representation have the same color. As expected the two-class models are more accurate than the twenty-class models.

Further, we see that models trained with all-layer information are more accurate than those trained on with only layer 7 information. This is due to the increased amount of information provided in these extra layers such as TCP port numbers, protocol identifiers, packet length information, and flags.

Finally, a surprising result is that the session representation does not always outperform the flow representation as in the twenty-class with layer 7 only case. This is surprising as the session representation provides more information than the flow representation in the form of server response data. This extra data should provide more information for the model to discriminate on, however we find that this is not always the case. In all other comparisons, the session representation was better than or equal to the flow representation. This shows that a flow representation may be a sufficient data representation to achieve the most accurate models. This is important as many network devices come with built-in with flow computing capabilities, but not session computing capabilities. We analyse this finding further in section 4.3.

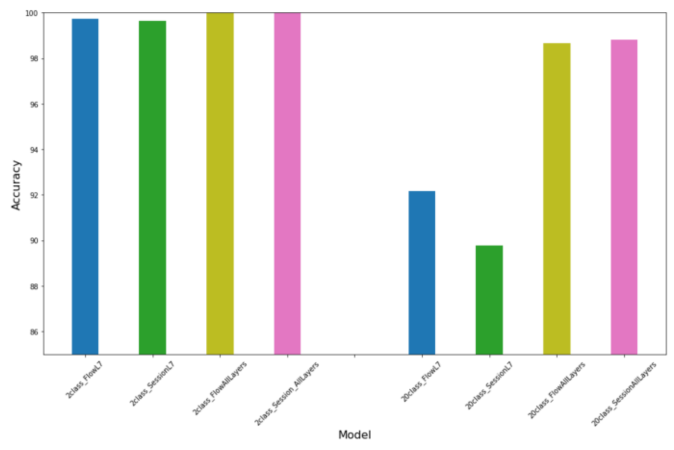


Figure 6. Testing accuracy plot

5.3 Further Analysis of twenty-class Models

Next we provide a further analysis of the twenty-class models by observing the precision, recall, and F1 scores for the various traffic classes, the definitions of which are provided in Equation Set 1. In Figure 7. we compare the worst performing (20class, session, layer 7) and best performing (20 class, session, all layer) twenty-class models. We find that there are five classes with a F1 score below 0.9 in the layer-7 model. In the all-layer model these classes are still the lowest, but the F1 score are drastically improved above 0.9. This shows these five classes are very susceptible to being identified by non-layer 7 data. It is interesting that both classes of email (Outlook and Gmail) as well as another data transfer class (BitTorrent) are among those that are the most difficult to classify, but a further data transfer class (SMB) is not. This suggests that not all arbitrary data transfer classes are difficult to detect, which could have input into privacy preserving protocol design.

Further in Figure 7. we compare the results of our most accurate model, the twenty-class-session-all-layer model to that produced in [1]. We find that our accuracy is less than three tenths of a percent lower at 98.81% compared to 99.17%. We see that our model has similar F1 scores for the various classes and has lower F1 scores for the same classes, suggesting we did an adequate job recreating their results.

In Figure 8. We further explore the surprising result where layer 7 flow out performed layer 7 session. We see in these results that these models struggled with the same five classes as in Figure 7. In this case the added data of the session representation provided more noise and less accurate results than the flow representation.

Finally, in both Figure 7. and Figure 8. we highlight in blue the lower of the precision and recall measures, but no discernible pattern was established.

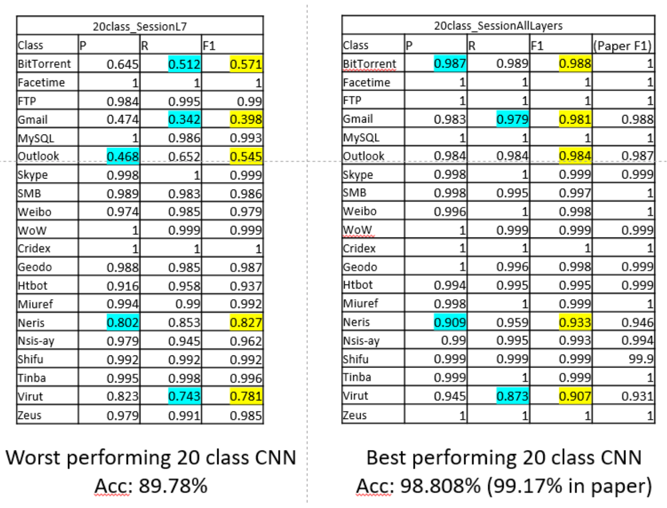
- $A = \frac{TP+TN}{TP+FP+FN+TN}$
- $P = \frac{TP}{TP+FP}$
- $R = \frac{TP}{TP+FN}$
- $F_1 = \frac{2PR}{P+R}$

Equation Set 1: Definition of accuracy (A), precision (P), recall (R) and F1 score.

6. Limitations and Future Work

In this section, we discuss limitations and opportunities identified by this experiment.

The first limitation of this experiment is the choice of the input size of 28x28 images. This is a seemingly arbitrary choice based on existing image classification models. In [6] they do more extensive



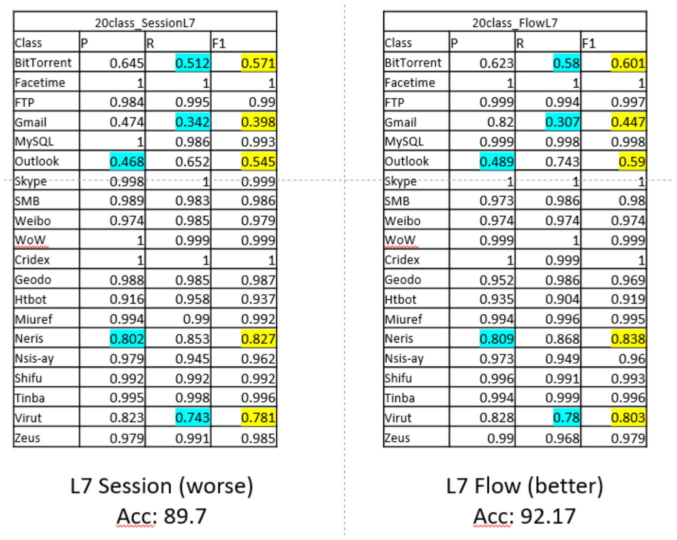
20class_SessionL7			
Class	P	R	F1
BitTorrent	0.645	0.512	0.571
Facetime	1	1	1
FTP	0.984	0.995	0.99
Gmail	0.474	0.342	0.398
MySQL	1	0.986	0.993
Outlook	0.468	0.652	0.545
Skype	0.998	1	0.999
SMB	0.989	0.983	0.986
Weibo	0.974	0.985	0.979
WoW	1	0.999	0.999
Cridex	1	1	1
Geodo	0.988	0.985	0.987
Htbot	0.916	0.958	0.937
Miuref	0.994	0.99	0.992
Neris	0.802	0.853	0.827
Nsis-ay	0.979	0.945	0.962
Shifu	0.992	0.992	0.992
Tinba	0.995	0.998	0.996
Virut	0.823	0.743	0.781
Zeus	0.979	0.991	0.985

Worst performing 20 class CNN
Acc: 89.78%

20class_SessionAllLayers				
Class	P	R	F1	(Paper F1)
BitTorrent	0.987	0.989	0.988	1
Facetime	1	1	1	1
FTP	1	1	1	1
Gmail	0.983	0.979	0.981	0.988
MySQL	1	1	1	1
Outlook	0.984	0.984	0.984	0.987
Skype	0.998	1	0.999	0.999
SMB	0.998	0.995	0.997	1
Weibo	0.996	1	0.998	1
WoW	1	0.999	0.999	0.999
Cridex	1	1	1	1
Geodo	1	0.996	0.998	0.999
Htbot	0.994	0.995	0.995	0.999
Miuref	0.998	1	0.999	1
Neris	0.909	0.959	0.933	0.946
Nsis-ay	0.99	0.995	0.993	0.994
Shifu	0.999	0.999	0.999	99.9
Tinba	0.999	1	0.999	1
Virut	0.945	0.873	0.907	0.931
Zeus	1	1	1	1

Best performing 20 class CNN
Acc: 98.808% (99.17% in paper)

Figure 7. Precision, recall, and F1 comparison of the worst and best performing 20-class CNN models.



20class_SessionL7			
Class	P	R	F1
BitTorrent	0.645	0.512	0.571
Facetime	1	1	1
FTP	0.984	0.995	0.99
Gmail	0.474	0.342	0.398
MySQL	1	0.986	0.993
Outlook	0.468	0.652	0.545
Skype	0.998	1	0.999
SMB	0.989	0.983	0.986
Weibo	0.974	0.985	0.979
WoW	1	0.999	0.999
Cridex	1	1	1
Geodo	0.988	0.985	0.987
Htbot	0.916	0.958	0.937
Miuref	0.994	0.99	0.992
Neris	0.802	0.853	0.827
Nsis-ay	0.979	0.945	0.962
Shifu	0.992	0.992	0.992
Tinba	0.995	0.998	0.996
Virut	0.823	0.743	0.781
Zeus	0.979	0.991	0.985

L7 Session (worse)
Acc: 89.7

20class_FlowL7			
Class	P	R	F1
BitTorrent	0.623	0.58	0.601
Facetime	1	1	1
FTP	0.999	0.994	0.997
Gmail	0.82	0.307	0.447
MySQL	0.999	0.998	0.998
Outlook	0.489	0.743	0.59
Skype	1	1	1
SMB	0.973	0.986	0.98
Weibo	0.974	0.974	0.974
WoW	0.999	1	0.999
Cridex	1	0.999	1
Geodo	0.952	0.986	0.969
Htbot	0.935	0.904	0.919
Miuref	0.994	0.996	0.995
Neris	0.809	0.868	0.838
Nsis-ay	0.973	0.949	0.96
Shifu	0.996	0.991	0.993
Tinba	0.994	0.999	0.996
Virut	0.828	0.78	0.803
Zeus	0.99	0.968	0.979

L7 Flow (better)
Acc: 92.17

Figure 8. Precision, recall, and F1 comparison of the 20-class-session-L7 model and 20-class-Flow-L7 model.

statistical analysis to identify what a proper size of input may be. This input size should be experimented with as well as trying out 1D CNN layers compared to 2D CNN layers.

Next, this model is trained and tested on data collected from the same network topologies within each subclass. It should be tested on data generated from separate topologies to determine if the model is generalizable to other networks. Can these modes be trained on data from one network and successfully deployed on a different organization's network? A study into whether the flow or session representation is more generalizable is called for.

Further, while IP and MAC address data are sanitized from the input, I posit that some of this information may be repeated within the data payload itself. Inspection of some of the data payloads, especially HTTP data, show that IP addresses are repeated in the payload. The data payloads should be sanitized from all IP addresses and the experiment reconducted to see if this had significant impacts on the result.

Due to unavailability of prepared data and time, we did not test models trained on individual packets. This is accomplished in [6]. This is an important research approach as models trained on individual packets will provide for a transition from detection (NIDS) to prevention (NIPS) as multiple packets are not required to traverse a network boundary to allow classification. A malicious packet can simply be dropped if detected. In [6] they demonstrate that models trained on individual packets are significantly less accurate that those using a flow or session representation. Research into improving individual packet models is called for.

The CNN model in this experiment only captures the spatial information contained in the network traffic, however, network traffic contains valuable temporal data as seen in statistical feature base NIDS approaches. These temporal features can potentially be added by exploring RNN or LSTM modifications to the classification model, or adding separate statistical features into the data.

Finally, we see that Neris and Virut are the two hardest malware classes to classify. The characteristics of these two classes should be studied to identify how these traffic characteristics may bypass model detection. From cursory glances at the image representation of these classes, we determine that multiple texture patterns exist. This suggests malware trying to evade CNN model detection could use polymorphic approached to change their image signature.

Conclusion

In conclusion, this study replicates and validates many of the findings presented in [1]. Namely, that the CNN models trained with all layer information and a session data representation provided the most accurate results over those that use only layer 7 data or a flow representation. Both this study and [1] show the validity of using CNN for network traffic classification tasks and the advantage of learning features through deep learning instead of time and compute consuming rule and feature computation. We provide our independent analysis of the results and identify a surprising finding that the session representation does not always outperform the flow representation. A flow-representation has advantages as it is simpler to capture and compute while providing mostly similar results to the session representation. We provide a code base in [8] that updates this experiment to use Python 3 and Pytorch. Finally, we provide extensive discussion of the limitations and opportunities identified by this experiment.

Acknowledgments

The author wishes to thank Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng, who authored the paper [1] that this project is based on. We also wish to thank Nicholas Majeske for providing a Pytorch CNN implementation that the code for this project is derived from. We also are thankful for Dr. Ariful Azad and the teaching staff of ENGR-503, Intelligent Systems Engineering, at Indiana University, for their valuable instruction.

References

[1] Wang, Wei & Zhu, Ming & Zeng, Xuewen & Ye, Xiaozhou & Sheng, Y.. (2017). Malware traffic classification using convolutional neural network for representation learning. 712-717.10.1109/ICOIN.2017.7899588. Available: https://www.researchgate.net/publication/316176048_Malware_traffic_classification_using_convolutional_neural_network_for_representation_learning

[2] Snort - Network Intrusion Detection & Prevention System. https://www.snort.org/. [Online]. Available: https://www.snort.org/

[3] Home - Suricata. https://suricata.io/. [Online]. Available: https://suricata.io/

[4] The Zeek Network Security Monitor. https://zeek.org/. [Online]. Available: https://zeek.org/

[5] Applications | Research | Canadian Institute for Cybersecurity | UNB. (n.d.). https://www.unb.ca/cic/research/applications.html. https://www.unb.ca/cic/research/applications.html

[6] G. Marín, P. Casas, and G. Capdehourat, DeepMAL – Deep Learning Models for Malware Traffic Detection and Classification. 2020. Available: https://arxiv.org/pdf/2003.04079.pdf

[7] Wang, Wei. https://github.com/echowei/DeepTraffic
[8] Orlowki, Anthony P. https://github.com/aporlowski/nids-cnn
[9] Ixia Corporation, Ixia Breakpoint Overview and Specifications, https://www.ixiacom.com/products/breakingpoint, 2016.