A Comparison of Header and Deep Packet Features when Detecting Network Intrusions

Gavin Watson, gkwatson@cs.umd.edu
University of Maryland, College Park

Abstract *– A classical multilayer perceptron algorithm and novel convolutional neural network payload classifying algorithm are presented for use on a realistic network intrusion detection dataset. The payload classifying algorithm is judged to be inferior to the multilayer perceptron but shows significance in being able to distinguish between network intrusions and benign traffic. The multilayer perceptron that is trained on less than 1% of the available classification data is judged to be a good modern estimate of usage in the real-world when compared to prior research. It boasts an average true positive rate of 94.5% and an average false positive rate of 4.68%.*

1 Introduction

As the internet continues to expand its userbase and becomes more and more synonymous with daily life, the threat of malicious traffic gains the possibility of becoming much greater in volume and more debilitating to home and commercial networks. Defending against this malicious traffic requires a suite of tools that will allow network administrators to detect, block, capture, and analyze the malicious streams that traverse their networks with high accuracy and speed.

One component of such a suite is known as an Intrusion Detection System (IDS), which is a tool for actively or retroactively detecting malicious streams within a body of traffic. These systems are usually meant to comprehensively identify any traffic that is unwanted by the network administrator and is generated within the network or sent from the outside. Examples include denial of service attacks, botnet traffic, heartbleed attacks, cross-site scripting attacks, worms, buffer overflow attacks, port scans, data exfiltration, and even malicious binary file transfers.

Two examples of commonly used IDS's are Cisco's Snort and the open-source Bro IDS [1][2]. These systems use techniques like generating signatures of known malicious files and having network security experts come up with rigidly defined rules for what constitutes malicious traffic. Signatures are generated automatically and matched against files that are extracted from network traffic, and rules include things like blacklisting certain IP addresses within or outside of the network, searching internet packet payloads for certain strings or regular expressions, and checking for combinations of metadata in internet packet headers that should never occur in legitimate traffic. Although both are effective, in actual use, signatures can only defend against exact replicas of known malware, and there can easily be hundreds of rules that need to be handwritten and tested. So, signatures cannot defend against differently coded versions of the same malware, zero-day attacks, and amorphous malware, which can alter its code, and new rules need to be created as new attack vectors are found, which constitutes a great commitment of time and expertise.

To improve upon these systems, a commonly researched approach is to classify network traffic at the session level by extracting a number of features from each session and using these features as the input to a machine learning (ML) algorithm like a random forest, support vector machine, bayesian network, or neural network [3][4][5]. The most successful ones are based on specific neural networks known as multilayer perceptrons (MLPs) and boast impressive accuracies of over 98% true positives and less than 1% false positives on the 2015 UNSW-NB15 dataset [6][7][8]. These accuracies, however, are suspect because the authors of [6] and [7] respectively use a staggering 90% and 70% of their datasets for training purposes and only evaluate their classifiers on the remaining 10% and 30%. Such results can be considered proofs-of-concept because they show that there is something that an MLP can learn from the input features to accurately predict a realistic subset of internet traffic. However, when this is thought of in a real-world context, it is not reasonable to assume that a network administrator will have access to over 50% of benign or malicious internet traffic for training.

For this reason, this paper presents similar experiments of MLP classification on network traffic with considerably smaller portions of the dataset allocated to training (>1%). A newer IDS dataset known as CI-CIDS2017 has also been chosen to evaluate the classifiers because it is significantly more recent and appears to be more representative of realistic traffic [4].

Most of the features for the MLP classifier were based on header features that have been shown to have an effective impact on network traffic classification [9]. However, experiments were also run with an additional feature that is analogous to what has been presented in other research as an effective classifier of network traffic into categories that are different from the IDS focus presented here, like transfer protocol type and application type [10][11]. This feature will be called the payload classifier because it predicts the maliciousness of every payload of a particular session based only on payloads on which it was trained. While this payload classifier showed significance and promise by itself, it was unfortunately unable to improve upon IDS predictions when incorporated into the MLP classifier.

The main contributions of this paper are showing that MLP IDS techniques are applicable to a modern environment and providing a better estimate of classification accuracy with these techniques. It also introduces a technique for leveraging payload features that shows significance in being able to classify packets with meaningful payloads.

The paper is organized as follows. Section 2 introduces this paper's implementation of MLP classification. Section 3 introduces the application of deep convolutional neural networks to payload classification and how it can be combined with an MLP. Section 4 presents the testing methodology and results of using the MLP and payload classifier. And section 5 provides the conclusion.

2 The MLP IDS Classifier

2.1 MLP Network Architecture

The basic MLP classifier was implemented in keras on top of TensorFlow and is a deep neural network that utilizes the Adam optimizer and consists of a 27-node input layer, which is followed by three fully connected 64-node layers that each have a dropout probability of 0.5 and rectified linear unit activation, which are followed be a single-node output layer with a sigmoid activation [12][13][14][15].

The 27 inputs are packet header features extracted from .pcap files by a custom Python framework and are based on ones that were found to be successful in [9]. They include (1) number of TCP packets, (2) number of UDP packets, (3) number of ICMP packets, (4) number of other packets, (5) average interarrival time of packets, (6) number of self-to-self connections, (7) number of wrong TCP packets, (8) number of urgent TCP packets, (9) number of packets sent over FTP, (10) number of packets sent over SSH, (11) number of packets sent over Telnet, (12) number of packets sent over SMTP, (13) number of packets sent over DNS, (14) number of packets sent over DHCP, (15) number of packets sent over TFTP, (16) number of packets sent over HTTP, (17) number of packets sent over POP3, (18) number of packets sent over NTP, (19) number of packets sent over NetBIOS, (20) number of packets sent over IMAP3, (21) number of packets sent over SNMP, (22) number of packets sent over BGP, (23) number of packets sent over LDAP, (24) number of packets sent over HTTPS, (25) number of packets sent over LDAPS, (26) number of packets sent over FTP using TLS/SSL, and (27) number of packets sent over another service. Each of these features was extracted over an entire .pcap, which was preprocessed to contain only one 5-tuple internet session, and each extraction except for (5) was then divided by the appropriate possible upper bound of the feature within the session in order to alleviate classification bias that arises from sessions having different numbers of packets. For example, the number of urgent TCP packets was divided by the total number of TCP packets.

After the features for a session are extracted and run through the MLP, it produces a score based on its training that is in the range [0, 1], where 0 corresponds to the session being benign and 1 corresponds to the session being malicious. A session was considered benign in practice when the output was below 0.5 and malicious otherwise.

2.2 MLP Training

A successful neural network classifier usually relies on clean and evenly sampled data to produce effective results, and it became clear from early tests that the IDS application is no different. Training the MLP classifier on unbalanced sets of benign and malicious examples always showed biased results, and even within each classification category, subcategories of types of benign and malicious traffic needed to be well balanced to work effectively.

After some testing on initial datasets, the following methodology was settled upon and relied on undersampling the training data and leveraging biased results. The total dataset is first split into malicious and benign subcategories based on the dataset. Then, each subcategory has 1% or less of its data randomly selected to appear in the training set; the rest is reserved for testing. The training data then has the number of benign and malicious streams compared; either benign or malicious streams are randomly removed such that the ratio of benign to malicious streams is 1.2:1, which is called undersampling because all of the training data is not actually being used. This has an advantage over other techniques like oversampling because all of the used data is still unique, and there is more benign than malicious training data to intentionally bias the classifier towards benign classifications since false positives can be considered a great limiting factor on the usefulness of an IDS [16].

3 The Payload Classifier

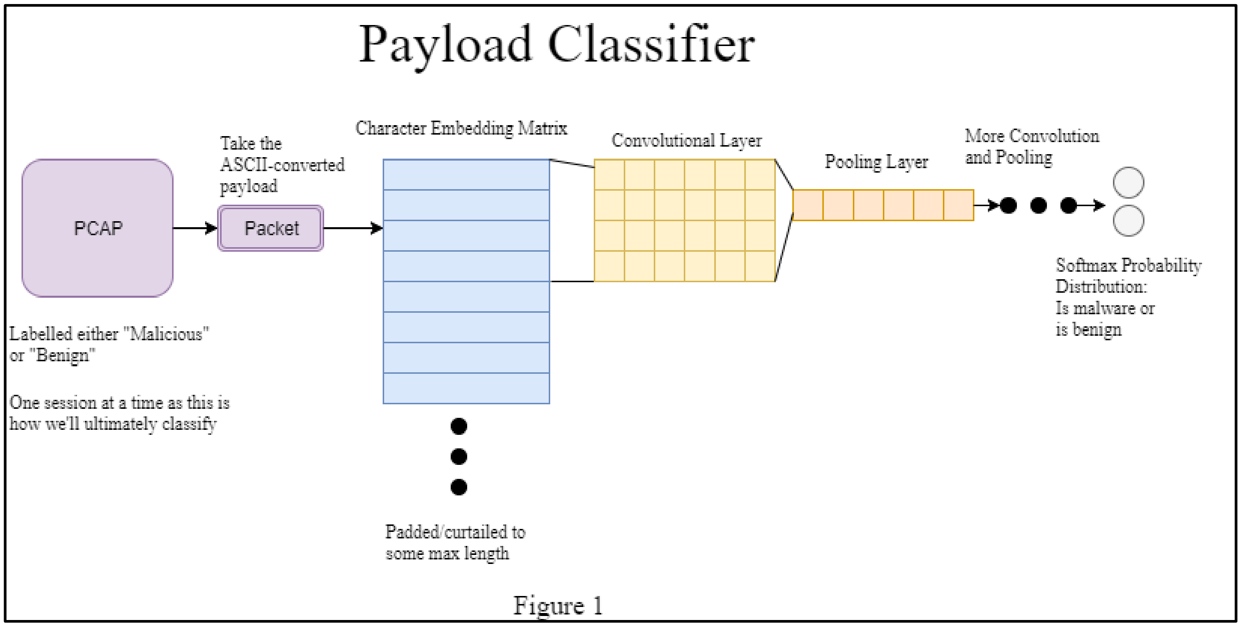

Payload Classifier
PCAP
Labelled either "Malicious" or "Benign"
One session at a time as this is how we'll ultimately classify
Take the ASCII-converted payload
Packet
Character Embedding Matrix
Padded/curtailed to some max length
Convolutional Layer
Pooling Layer
More Convolution and Pooling
Softmax Probability Distribution: Is malware or is benign

Figure 1

3.1 Payload Classifier Architecture

The payload classifier is basically a heavy-duty feature that makes a prediction of session maliciousness based on only the packet payloads in a session. Its original presentation and testing can be found in [17] and will be summarized here. It is also implemented in keras on top of TensorFlow and utilized the Adam optimizer, but it is a deep convolutional neural network [12][13][14]. It consists of a character embedding layer that's followed by four convolutional and pooling layers that are followed by a two-classification softmax layer as shown in Figure 1. Most of the parameters like number of layers, sliding window size, number of kernels, activation functions, and pooling layer sizes are based on the designs of previous effective works or initial testing with a small dataset [10][11].

The embedding layer is a pre-trained character-to-character co-occurrence matrix of 257x256 values. Each encodes a vector of the context for each of the possible 256 values that a byte may contain, except for the last vector, which is maintained as a zero vector so that packets that are too short and input into the model can be padded with this pseudo-byte that will not contribute any features to the convolutional layers. For pre-training, all of the payloads from the selected training set have a sliding context window of three characters run over them, and the vector in the character-to-character matrix that represents the middle character is given plus one in weight at the index of the other characters seen in the window. This is the same idea as a word-to-word co-occurrence matrix with a sliding window of size three, but by using characters in place of words, and it relies on deterministically converting every byte into an index in the range [0, 255] based on the decimal value of each byte. Each vector in the embedding matrix, except for the last one that is reserved for a vector of zeros, is then normalized to a magnitude of one.

When the payload classifier is used after this pre-training, a payload is converted into a 1500x257 matrix by the embedding layer, which converts each character into its learned embedding and appends all of the embeddings onto each other in the same order as the payload. If a payload is larger or smaller than 1500 characters, this matrix is curtailed or padded with the zero vector. They should not be longer than this by convention as stated by Lotfollahi et al., so this should not highly affect results [18].

The kernels in the convolutional layers each span four entire character vectors and use rectified linear unit activation. Each of the four layers learns 128 kernels, and each is followed by a max pooling layer over four entire character vectors, except for the last one that pools over sixteen vectors. The last pooling layer is then fully connected to a sixteen-node rectified linear unit layer, which is then fully connected to a two-node softmax layer. The output of this layer corresponds to a percentage of confidence that input payload was malicious or benign.

Building upon this payload classifier, there is a slightly larger architecture to actually predict things for sessions of multiple packets, although the payload classifier has to be trained before this prediction can occur. This session classifier inputs all of the packets from a single session, passes all of the TCP packets through the payload classifier to get a prediction for each, aggregates these predictions into a malware score by adding all of the malicious prediction percentages and subtracting all of the benign prediction percentages, and divides this score by the number of packets input to get a final malware prediction that is independent of session length. What is produced is a single number score in the range [-1, 1] that corresponds to how strong a prediction the network has made on the session as to whether it is malicious or benign. Positive scores indicate maliciousness, negative scores indicate benignity, and scores near zero either correspond to unseen data or low confidence predictions. This score can be used directly to predict whether a session is malicious or benign with everything below or equal to zero being classified as benign, but this score has also been used as a single heavy-duty input feature for the MLP as mentioned earlier.

3.2 Payload Classifier Training

The payload classifier was trained independently of the MLP but in a similar manner that utilized undersampling. Instead of training on the sessions, the payload classifier first extracts all of the benign and malicious packet payloads from the data and randomly removes some of them until the ratio of benign to malicious payloads is 1.25:1. Then, all of the remaining payloads are run in a random order through the network to train it. Similar to the MLP network, the result should be a classifier that is biased towards picking benign classifications when it does not detect features that are clearly indicative of maliciousness or benignity.

4 Evaluation

Several datasets were used to tune and bug test the MLP classifier both with and without the payload classifier. In order to keep the final results free of possible bias from hyperparameters or coding mistakes, the classifying program must be complete before introducing the final evaluation dataset and drawing conclusions from it. This section goes through the tests that were run to initially create the classifier and evaluate it against a wholistic malicious and benign IDS dataset.

It should also be noted that for performance reasons, any .pcap session that was larger than 10,000 packets was split into chronologically contiguous, non-overlapping increments of 10,000 packets before being input into the MLP classifier. If any increment of a session was predicted as malicious, the entire session was predicted as malicious, and this only counts towards one correct or incorrect prediction in the statistics.

4.1 Initial Testing

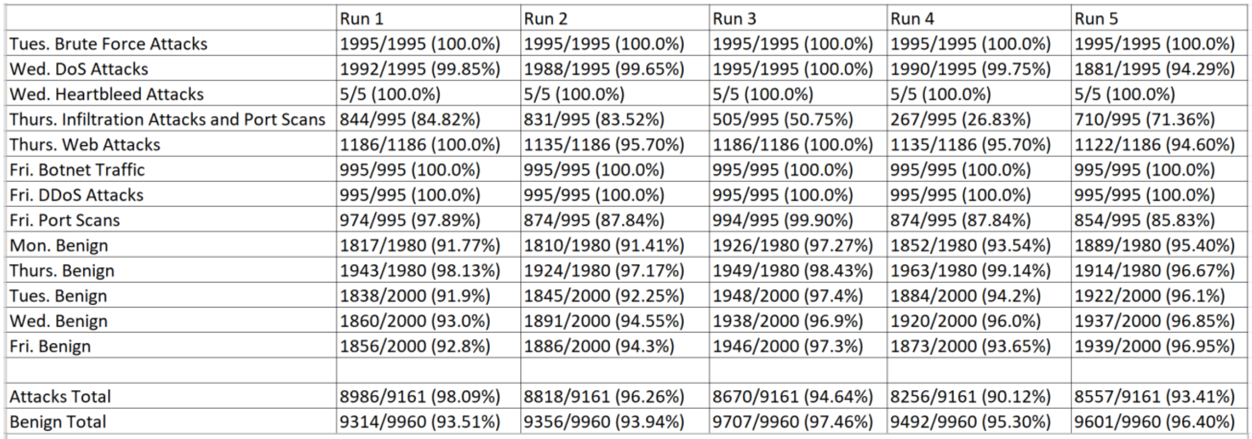
The first dataset that was used to test and tune the coding of the system was a combination of three separate datasets and was originally introduced in [17]. The first is called Contagio and is an amalgamation of 217 distinct samples of the malware executables producing internet traffic that are captured in individual .pcaps [19]. The second is called ISCX IDS 2012, which is a simulated IDS dataset created by the UNB Canadian Institute for Cybersecurity [20]. The third is dubbed the Benign Binary dataset and is a collection of 60 HTTPS downloads of popular pieces of software that were monitored with tcpdump. Malicious and benign .pcaps were used directly from Contagio and Benign Binary while the benign .pcaps present in the ISCX 2012 dataset were initially cut apart on a per-session basis by SplitCap and then had 150 samples randomly selected before usage [21].

Hyperparameters that were tested include payload classifier and MLP layers being varied from 1 to 6, payload classifier layer sizes that were varied between 64, 128, and 256, payload classifier kernel sizes that were varied from 2 to 6, MLP layer sizes that were varied between 16, 32, and 64, payload classifier and MLP optimization algorithms that were varied between Adam, RMSProp, and stochastic gradient descent, method of evening input data that was varied between oversampling and undersampling, and amount of data used in training that was varied between 1%, 10%, 30%, and 50%. The parameters were varied mostly independently because testing times were prohibitively long to test every combination, and it was noticed that the percentage of input data used only had a small effect on either classifier’s ability to correctly predict the testing set. In the end, about 10% of the data was used for training simply because the dataset had so few sessions in it.

By itself, the payload classifier achieved a significant F1 score of 0.7538 by predicting 90.9% of the benign files and 65.99% of the malicious files in the testing set correctly. The MLP performed even better by itself by achieving an F1 score of 0.8988 by predicting 85.64% of the benign files correctly and 93.33% of the malicious files correctly. Unfortunately, adding the payload classifier as an extra input feature at this point did not show any discernable improvement to the classification accuracy of the MLP. Although the payload classifier undersampling could be altered to give it a near-perfect benign prediction rate while still predicting malicious files at a rate >40%, it did not translate to picking up on any predictions that the original MLP missed as was expected.

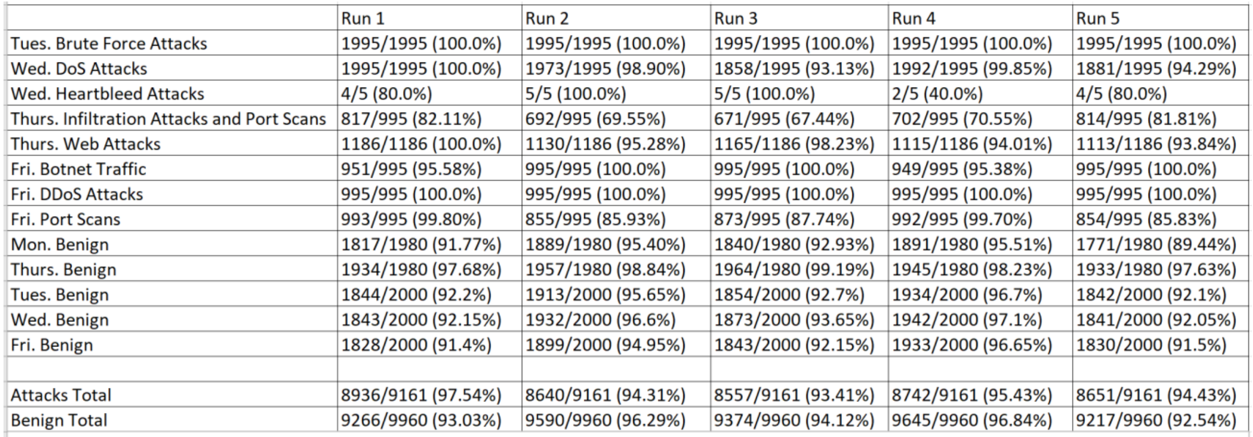
4.2 Final Tests on CICIDS2017 Dataset

After the classifier code, sampling method, and hyperparameters were finalized, the classifier was finally evaluated on the CICIDS2017 dataset [4]. This dataset contains a wide variety of activity that was generated in a twelve-machine network over five days. The malicious activity included spans portions of each of the last four days and can be categorized into (1) brute force password cracking attacks, (2) DoS attacks, (3) heartbleed attacks, (4) web attacks (brute force, SQL Injections, and XSS), (5) infiltration attacks (malicious Dropbox and flash memory downloads leading to port scans), (6) botnet attacks, (7) DDoS attacks, and (8) port scans [4]. The benign traffic included runs continuously throughout the five days and includes realistically profiled traffic over



	Run 1	Run 2	Run 3	Run 4	Run 5
Tues. Brute Force Attacks	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)
Wed. DoS Attacks	1992/1995 (99.85%)	1988/1995 (99.65%)	1995/1995 (100.0%)	1990/1995 (99.75%)	1881/1995 (94.29%)
Wed. Heartbleed Attacks	5/5 (100.0%)	5/5 (100.0%)	5/5 (100.0%)	5/5 (100.0%)	5/5 (100.0%)
Thurs. Infiltration Attacks and Port Scans	844/995 (84.82%)	831/995 (83.52%)	505/995 (50.75%)	267/995 (26.83%)	710/995 (71.36%)
Thurs. Web Attacks	1186/1186 (100.0%)	1135/1186 (95.70%)	1186/1186 (100.0%)	1135/1186 (95.70%)	1122/1186 (94.60%)
Fri. Botnet Traffic	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)
Fri. DDoS Attacks	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)
Fri. Port Scans	974/995 (97.89%)	874/995 (87.84%)	994/995 (99.90%)	874/995 (87.84%)	854/995 (85.83%)
Mon. Benign	1817/1980 (91.77%)	1810/1980 (91.41%)	1926/1980 (97.27%)	1852/1980 (93.54%)	1889/1980 (95.40%)
Thurs. Benign	1943/1980 (98.13%)	1924/1980 (97.17%)	1949/1980 (98.43%)	1963/1980 (99.14%)	1914/1980 (96.67%)
Tues. Benign	1838/2000 (91.9%)	1845/2000 (92.25%)	1948/2000 (97.4%)	1884/2000 (94.2%)	1922/2000 (96.1%)
Wed. Benign	1860/2000 (93.0%)	1891/2000 (94.55%)	1938/2000 (96.9%)	1920/2000 (96.0%)	1937/2000 (96.85%)
Fri. Benign	1856/2000 (92.8%)	1886/2000 (94.3%)	1946/2000 (97.3%)	1873/2000 (93.65%)	1939/2000 (96.95%)
Attacks Total	8986/9161 (98.09%)	8818/9161 (96.26%)	8670/9161 (94.64%)	8256/9161 (90.12%)	8557/9161 (93.41%)
Benign Total	9314/9960 (93.51%)	9356/9960 (93.94%)	9707/9960 (97.46%)	9492/9960 (95.30%)	9601/9960 (96.40%)

Table 1: CICIDS2017 Tests with MLP without payload classifier



	Run 1	Run 2	Run 3	Run 4	Run 5
Tues. Brute Force Attacks	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)	1995/1995 (100.0%)
Wed. DoS Attacks	1995/1995 (100.0%)	1973/1995 (98.90%)	1858/1995 (93.13%)	1992/1995 (99.85%)	1881/1995 (94.29%)
Wed. Heartbleed Attacks	4/5 (80.0%)	5/5 (100.0%)	5/5 (100.0%)	2/5 (40.0%)	4/5 (80.0%)
Thurs. Infiltration Attacks and Port Scans	817/995 (82.11%)	692/995 (69.55%)	671/995 (67.44%)	702/995 (70.55%)	814/995 (81.81%)
Thurs. Web Attacks	1186/1186 (100.0%)	1130/1186 (95.28%)	1165/1186 (98.23%)	1115/1186 (94.01%)	1113/1186 (93.84%)
Fri. Botnet Traffic	951/995 (95.58%)	995/995 (100.0%)	995/995 (100.0%)	949/995 (95.38%)	995/995 (100.0%)
Fri. DDoS Attacks	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)	995/995 (100.0%)
Fri. Port Scans	993/995 (99.80%)	855/995 (85.93%)	873/995 (87.74%)	992/995 (99.70%)	854/995 (85.83%)
Mon. Benign	1817/1980 (91.77%)	1889/1980 (95.40%)	1840/1980 (92.93%)	1891/1980 (95.51%)	1771/1980 (89.44%)
Thurs. Benign	1934/1980 (97.68%)	1957/1980 (98.84%)	1964/1980 (99.19%)	1945/1980 (98.23%)	1933/1980 (97.63%)
Tues. Benign	1844/2000 (92.2%)	1913/2000 (95.65%)	1854/2000 (92.7%)	1934/2000 (96.7%)	1842/2000 (92.1%)
Wed. Benign	1843/2000 (92.15%)	1932/2000 (96.6%)	1873/2000 (93.65%)	1942/2000 (97.1%)	1841/2000 (92.05%)
Fri. Benign	1828/2000 (91.4%)	1899/2000 (94.95%)	1843/2000 (92.15%)	1933/2000 (96.65%)	1830/2000 (91.5%)
Attacks Total	8936/9161 (97.54%)	8640/9161 (94.31%)	8557/9161 (93.41%)	8742/9161 (95.43%)	8651/9161 (94.43%)
Benign Total	9266/9960 (93.03%)	9590/9960 (96.29%)	9374/9960 (94.12%)	9645/9960 (96.84%)	9217/9960 (92.54%)

Table 2: CICIDS2017 Tests with MLP with payload classifier

HTTP, HTTPS, FTP, SSH, and email protocols [4]. This dataset was distributed in five .pcaps, one for each day of traffic, which was prepared by using SplitCap to separate all of the sessions into five benign subcategories for each day's benign traffic and eight malicious subcategories for each of the attack categories listed [21].

For each test of the MLP, 5 sessions were randomly selected for the training set and 995 sessions were randomly selected for the testing set from each of six of the eight attack subcategories – the web attacks had the remaining 1186 sessions always picked for the testing set and heartbleed attacks were only included in the testing set because the dataset only included five of them. Furthermore, 20 sessions were randomly selected for the training set and 1980 sessions were randomly selected for the testing set from both the Monday and Thursday benign subcategories, and only 2000 sessions of training data were randomly selected from the Tuesday, Wednesday, and Friday benign subcategories. Five such tests were run both by using the MLP classifier with and without the payload classifier. The payload classifier was initially run by itself without the MLP, and although it performed significantly better than random, it was not comparable to the inclusion of header features.

The results of the five runs using the MLP classifier without the payload classifier can be explicitly seen in Table 1; the percentages in each cell represent the number of correct predictions. The average F1 score was 0.9488, the average attack detection rate was 94.5%, and the average false positive rate was 4.68%. The average classification time was about 0.41 seconds per megabyte, which is reasonable for modern home bandwidths but may need further optimization in a business setting.

The results of the five runs using the MLP classifier with the payload classifier can be seen in Table 2. The average F1 score was 0.9480, the average attack detection rate was 95.02%, and the average false positive rate was 5.44%. This is almost identical to the MLP without the payload classifier, but it took around fifteen times longer to train and test, so adding the payload classifier clearly has not improved the basic MLP.

For both classifiers, the predictions are generally very accurate across all subcategories, and each run's accuracies are generally closely grouped across all subcategories except for "Thurs. Infiltration Attacks and Port Scans." Upon more detailed inspection of the data, it appears that this subcategory contains mostly port scans, which is the same as the second category with the lowest successful detection rates, "Fri. Port Scans." Furthermore, the runs with the lowest port scan detection rates generally seem to be the runs that have the fewest false positives. This makes it appear as though the features used are not sufficient to clearly identify port scans because these particular features are similar in port scans and some benign connections.

5 Conclusion

Using 27 basic network packet header features, the MLP presented in this paper was able to classify malicious and benign streams at a very high rate at a reasonable speed. Although prior works were able to boast better accuracy, the MLP presented here used such a smaller portion of a more modern dataset for training that it is expected to be much more indicative of real-world usage. As such, network administrators can be more confident when looking at this data and deciding the costs in terms of false positives and negatives when implementing a similar system to alert them of likely intrusions into their networks.

Unfortunately, although the payload classifier technique presented was shown to be effective at other classification tasks besides intrusion detection in [10]and [11], it was unable to add anything to an MLP classifier. As it stands, the payload classification showed significance in being able to detect some attacks by itself, but it was less accurate, more complicated, and about fifteen times slower to train and test than the MLP by itself. In fact, some sessions that had the densest payloads took up to 95 times longer to classidy with the payload classifier. Future work might include attempting to find exactly what subcategories of traffic the payload classifier can effectively identify and incorporating it in a different manner into an IDS.

Both classifier techniques also appeared to perform particularly poorly at detecting port scan attacks. Most of the classifier tests detected them at a fairly high rate, but removing port scans clearly would have improved on overall malicious detection rates and may have improved on benign detection rates if the cause of the problem is that port scans have very similar features to benign traffic. Evaluating the usage of a classifier like the presented MLP on everything but port scans while using some other method to detect port scans may be a fruitful avenue for future work to further increase accuracy.

5 Acknowledgements

This work was supported by the United States Office of Naval Research under Contract N000141612107. I would also like to thank my adviser Dr. James Purtilo for all

his assistance with this project and Dr. Dave Levin and Ronald Cheng for their initial help in designing and testing the payload classifier presented in [17].

7 References

[1] https://www.snort.org

[2] V. Paxson, "Bro: a system for detecting network intruders in real-time", Computer Networks, vol. 31, no. 23-24, pp. 2435-2463, 1999.

[3] Sabhnani, Maheshkumar and Gürsel Serpen. “Application of Machine Learning Algorithms to KDD Intrusion Detection Dataset within Misuse Detection Context.” *MLMTA*, 2003.

[4] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”, 4th International Conference on Information Systems Security and Privacy (ICISSP), Purtogal, January 2018.

[5] N. Moustafa and J. Slay, "The Significant Features of the UNSW-NB15 and the KDD99 Data Sets for Network Intrusion Detection Systems," *2015 4th International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS),* Kyoto, 2015, pp. 25-31.

[6] M. Baig, E. S. M. El-Alfy and M. M. Awais, "Intrusion detection using a cascade of boosted classifiers (CBC)," *2014 International Joint Conference on Neural Networks (IJCNN),* Beijing, 2014, pp. 1386-1392.

[7] M. Al-Zewairi, S. Almajali and A. Awajan, "Experimental Evaluation of a Multi-layer Feed-Forward Artificial Neural Network Classifier for Network Intrusion Detection System," *2017 International Conference on New Trends in Computing Sciences (ICTCS),* Amman, 2017, pp. 167-172.

[8] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference (MilCIS),* Canberra, ACT, 2015, pp. 1-6.

[9] Preeti Aggarwal, Sudhir Kumar Sharma, “Analysis of KDD Dataset Attributes – Class wise for Intrusion Detection”, Procedia Computer Science, Volume 57, 2015, Pages 842-851, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2015.07.490.

[10] M. Lotfollahi, R. Hossein Zade, M. Jafari Siavoshani and M. Saberian, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning", ARXIV, vol. 1709, no. 02656, 2017.

[11] J. Saxe, K. Berlin, “eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys", ARXIV, vol. 1702, no. 08568, 2017.

[12] https://keras.io/

[13] https://www.tensorflow.org/

[14] Diederik Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, International Conference on Learning Representations, 2014.

[15] Srivastava, Hinton, Krizhevsky, Sutskever, and Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research 15 (2014) 1929-1958, 2014.

[16] Stefan Axelsson, “The Base-Rate Fallacy and the Difficulty of Intrusion Detection”, 20 May 1999.

[17] R. Cheng and G. Watson, “D2PI: Identifying Malware through Deep Packet Inspection with Deep Learning”, 2018.

[18] M. Lotfollahi, R. Hossein Zade, M. Jafari Siavoshani and M. Saberian, "Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning", ARXIV, vol. 1709, no. 02656, 2017.

[19] http://contagiodump.blogspot.com/2013/04/collection-of-pcap-files-from-malware.html

[20] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, Ali A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection”, Computers & Security, Volume 31, Issue 3, May 2012, Pages 357-374, ISSN 0167-4048, 10.1016/j.cose.2011.12.012.

[21] https://www.netresec.com/?page=SplitCap

[22] https://www.hybrid-analysis.com/

[23] https://httpd.apache.org/