

## PUBLISHED VERSION

Daniel Smit, Kyle Millar, Clinton Page, Adriel Cheng, Hong-Gunn Chew and Cheng-Chew Lim

### Looking deeper: Using deep learning to identify internet communications traf

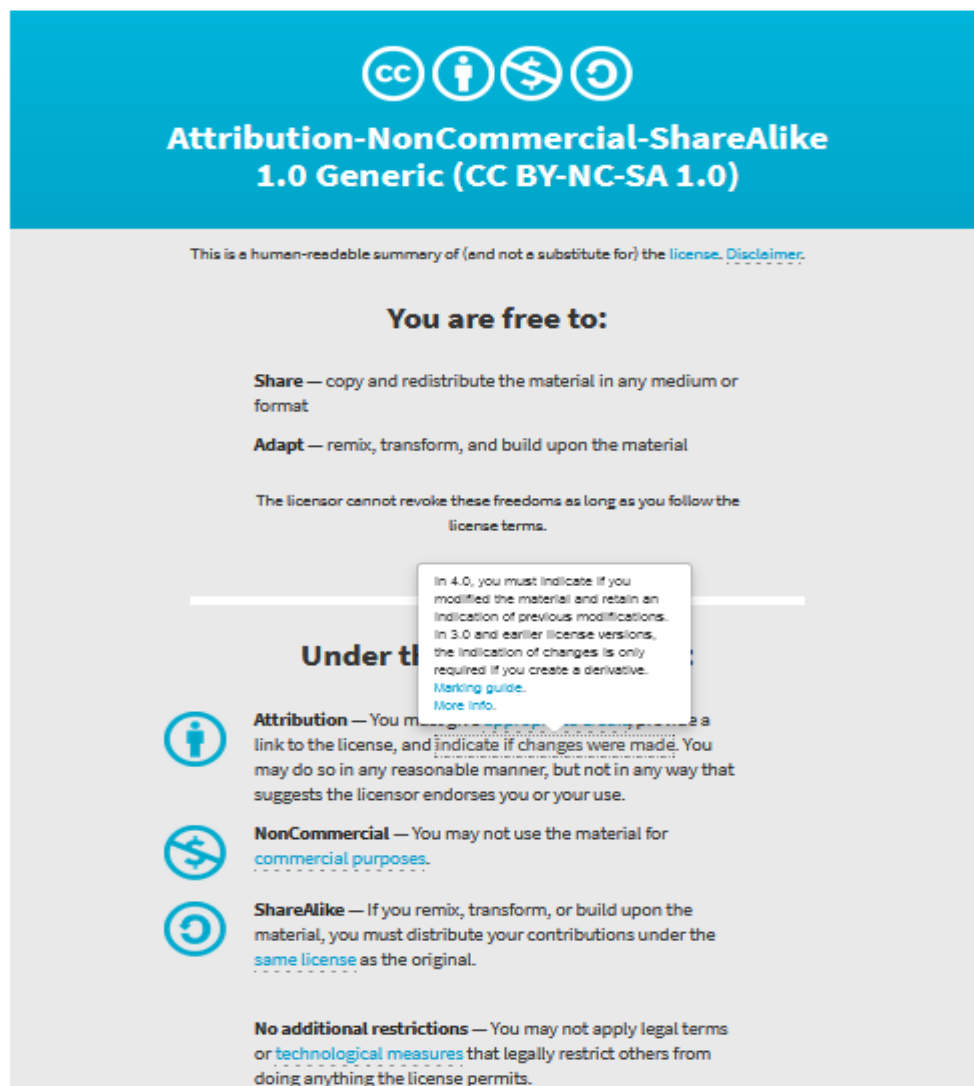
Proceedings of the 2017 Australasian Conference of Undergraduate Research, as published in Macquarie Matrix: Special edition, ACUR 2017, 2017 / vol.6.1, pp.124-144

© the author(s). Open Access. All works published are under an attribution, non-commercial, and share-alike Creative Commons licence. Attribution-NonCommercial-ShareAlike 1.0 Generic (CC BY-NC-SA 1.0)

Published version: <https://students.mq.edu.au/study/my-study-program/undergraduate-research-journal/acur2017>

## PERMISSIONS

<https://creativecommons.org/licenses/by-nc-sa/1.0/>



The image shows a summary of the Creative Commons Attribution-NonCommercial-ShareAlike 1.0 Generic (CC BY-NC-SA 1.0) license. It features a blue header with the license name and icons for Attribution (person), Non-Commercial (dollar sign with slash), and Share-Alike (circular arrow). Below the header, it states: "This is a human-readable summary of (and not a substitute for) the license. [Disclaimer.](#)"

**You are free to:**

- Share** — copy and redistribute the material in any medium or format
- Adapt** — remix, transform, and build upon the material

The licensor cannot revoke these freedoms as long as you follow the license terms.

**Under the following conditions:**

- Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- NonCommercial** — You may not use the material for commercial purposes.
- ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.

**No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

In 4.0, you must indicate if you modified the material and retain an indication of previous modifications. In 3.0 and earlier license versions, the indication of changes is only required if you create a derivative. [Marking guide.](#) [More info.](#)

21 September 2020

<http://hdl.handle.net/2440/128170>

## Looking deeper: Using deep learning to identify internet communications traffic

Daniel Smit<sup>1</sup>, Kyle Millar<sup>1</sup>, Clinton Page<sup>1</sup>, Adriel Cheng<sup>1,2</sup>, Hong-Gunn Chew<sup>1</sup> and

Cheng-Chew Lim<sup>1</sup>

1. The University of Adelaide

2. Defence Science and Technology Group

### Abstract

Recent years have shown an unprecedented reliance on the internet to provide services essential for business, education, and personal use. Due to this reliance, coupled with the exponential growth of the internet traffic being generated, there has never been a greater necessity for effective network management techniques. Network traffic classification is one key component of this network management which aims to identify the types and quantity of traffic flowing through a network. Previous traffic classification techniques are limited by the use of non-standardised port numbers and the encryption of traffic contents. To tackle these challenges, we propose using deep learning techniques for network traffic classification. This paper investigates the viability of using deep learning for traffic classification with a focus on both network management applications and detecting malicious traffic. Our preliminary results thus far show that a highly accurate classifier can be created using the first 50 bytes of a traffic flow.

**Keywords:** deep learning, internet traffic classification, artificial neural networks, network security

## **1. Introduction**

The Internet has become a key facilitator of large-scale global communications and is vital for providing an immeasurable number of services. With the ever-expanding growth of Internet use, it is critical for the underpinning networks to be managed effectively. Network traffic classification is one such technique that allows the forecasting of traffic on a network to improve the Quality of Service (QoS) and identification of potential security threats. It is for this reason network traffic classification has become a crucial component of network management for Internet Service Providers (ISPs), large enterprise companies and government agencies. Deep learning is a subfield of machine learning which enables classification models to be trained without explicit programming. Deep learning models generally consist of a large number of processing units which can be used to classify data. This research investigates the viability of using deep learning techniques to provide this network traffic classification.

Previously used methods of network traffic classification include port based and signature based methods. Port based classification operates by classifying network traffic based on port numbers contained in the header section of each datagram sent over a network. This simplistic method has become less effective in recent years as it has become easier to circumvent by applications changing their port number. Signature based methods involve matching a sequence of bytes or properties to a known signature, but is limited to identifying only signatures for known application protocols.

Deep learning addresses the issues faced by other methods by automating the adaptation process of new signatures. With the recent rise of deep learning, there has been limited documented approaches to utilising these deep learning methods for the purpose of network traffic classification. In this paper, we aim to (1) establish the viability of using deep

learning for classifying network traffic flows, (2) devise strategies to optimise neural-network parameters for traffic classification, and (3) to expand and investigate the potential of using deep learning for identifying malicious network traffic flows for Cyber Security. The contributions of this paper are as follows:

- Extending from prior work by [Wang](#) (2015), we devised a deep neural network classifier achieving classification accuracy of over 90% for application protocols in the UNSW-NB15 dataset ([Nour & Slay](#), 2015).
- Passing the first 50 bytes of a network flow rather than the first 1000 bytes to a deep neural network can result in higher classification accuracies.
- We have expanded on the experiments conducted by Wang to investigate its ability to classify malicious traffic.
- Through experimentation, we have determined an optimal set of network parameters for use in these experiments.

## **2. Relevant work**

Machine learning techniques can be applied to classify network traffic flows and has been the target of several recent studies. These methods generally differ in the representations of network traffic flow data, and the machine learning algorithms used.

### ***2.1 Network application protocol identification***

Wang (2015) used deep learning to identify common network traffic within a self-generated data set. Wang's method used the first 1000 bytes of each Transmission Control Protocol (TCP) flow as input data, with results indicating the most important bytes for classification were early in the flow. Results showed that 55% of flows were correctly classified using a

90% probability cut-off point, but this could potentially be improved by ignoring the least significant input bytes.

[Trivedi, Chow, Nilsson, & Trussell](#) (2002) used a neural network with statistical information and IP layer packet attributes to classify TCP flows into different application protocols. Using this method, they achieved an overall accuracy above 98%.

[Auld, Moore, & Gull](#) (2007) used Bayesian neural networks and features of TCP flows to classify network traffic flows to a high accuracy. An accuracy of 99% was achieved when using a testing and training dataset from the same day, while 95% accuracy was achieved with a testing set eight months after the training set.

[Singh, Agrawal, & Sohi](#) (2013) used five machine learning methods, (including Multilayer Perception (MLP), Radial Basis Function (RBF), C4.5, Bayes Net and Naïve Bayes) to classify IP traffic in real time. They concluded that Bayes Net gave the highest classification accuracy of 91.875%, potentially lower than the other papers discussed due to their focus on real time classification and necessity for fast classification results.

This research paper uses deep learning methods similar to Wang (2015), but also investigates the effect of fine tuning the various deep learning parameters. Investigation is expanded over these studies by classifying UDP traffic in addition to TCP traffic. Our method does not use any statistical information or features of the network traffic flows while these are used for Trivedi et al. (2002) and Auld et al. (2007) as their input data. Auld et al. (2007) used two datasets (training and testing) generated eight months apart, while two datasets generated one month apart are used in our method. Additionally, our method only focuses on using neural networks and deep learning while Singh et al. (2013) also investigate a range of traditional machine learning methods.

## **2.2 Malicious traffic identification**

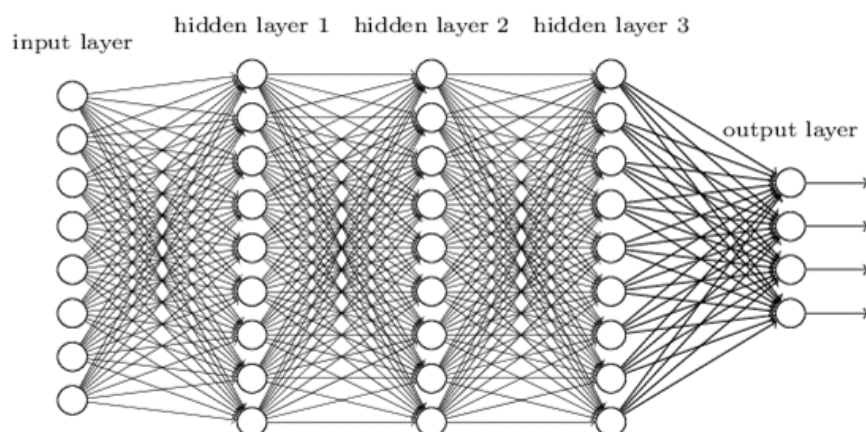
Accurate classification of different malicious network traffic types generally proved to be more challenging than standard application protocols. Dong & Wang (2016) used methods including Support Vector Machine (SVM), Support Vector Machine Restricted Boltzmann Machines (SVM-RBMs), Naïve Bayes and C4.5 to classify various malicious traffic. SVM-RBMs were found to give the best results, however classification of different malicious types did not reach precision values above 90%.

This research paper focuses on the effectiveness of deep learning rather than the traditional machine learning methods used in [Dong & Wang \(2016\)](#). The optimal model parameters and deep learning approach are also determined for the model.

## **3 Background knowledge**

### **3.1 Neural networks and deep learning**

Deep learning was the focus of this research paper due to its potential ability in producing an accurate network traffic flow classifier. Deep learning is a machine learning method which uses neural networks containing multiple hidden layers, known as a deep neural network. As shown in Figure 1, a neural network takes in a number of input features, and through a variable number of hidden layers and nodes, connects through weighted values to the output nodes. This method allows the computer to learn a representation of a complex system, by considering basic concepts. Multiple hidden layers also achieve automatic feature discovery in which a higher-level set of input features can find lower level features. Through this method, a combination of weights between layers are trained and a suitable combination of input features can be used to calculate an appropriate output.



**Figure 1:** Example of a deep neural network ([Yang, 2015](#))

### 3.2 Network traffic flows

A network traffic flow is a sequence of packets forming a conversation between two end-points of a network, and defined by various properties. The definition of flow used in this research was selected as the bi-directional five-tuple set of properties: source IP address, destination IP address, source port, destination port, and protocol type. The protocol types used were TCP and User Datagram Protocol (UDP).

## 4. Dataset

Deep learning benefits from a large and extensive dataset. The UNSW-NB15 dataset ([Nour & Slay, 2016](#)), released in 2015, contains raw packet level data and some extracted flow level features. The network traffic flows in this dataset contain a range of application and malicious labels. Through deep learning, we intend to generate a deep learning model to classify these flows accurately against the labels. While only 51% of the dataset has predefined application labels, it contains 1.29 million network application entries and 0.32 million malicious entries, making the dataset suitable for general network traffic

identification and detecting malicious traffic. A process of adding additional application labels to the dataset was performed using nDPI, which were used for entries which did not originally contain labels. The application and malicious classes chosen can be seen in Table 1. The application protocols and malicious classes shown in Table 1 represent several prominent types of traffic or malicious traffic that can be found in a given network. Unlabelled application and malicious samples have been classified as either *Other* or *Other Malicious* respectfully.

**Table 1:** Chosen classifications

Application Protocols	Malicious Classes
DNS	Exploits
FTP	Fuzzers
FTP-DATA	Generic
Mail	DOS
SSH	Reconnaissance
P2P	Other Malicious
NFS	Non-Malicious
HTTP	
BGP	
OSCAR	
Other	

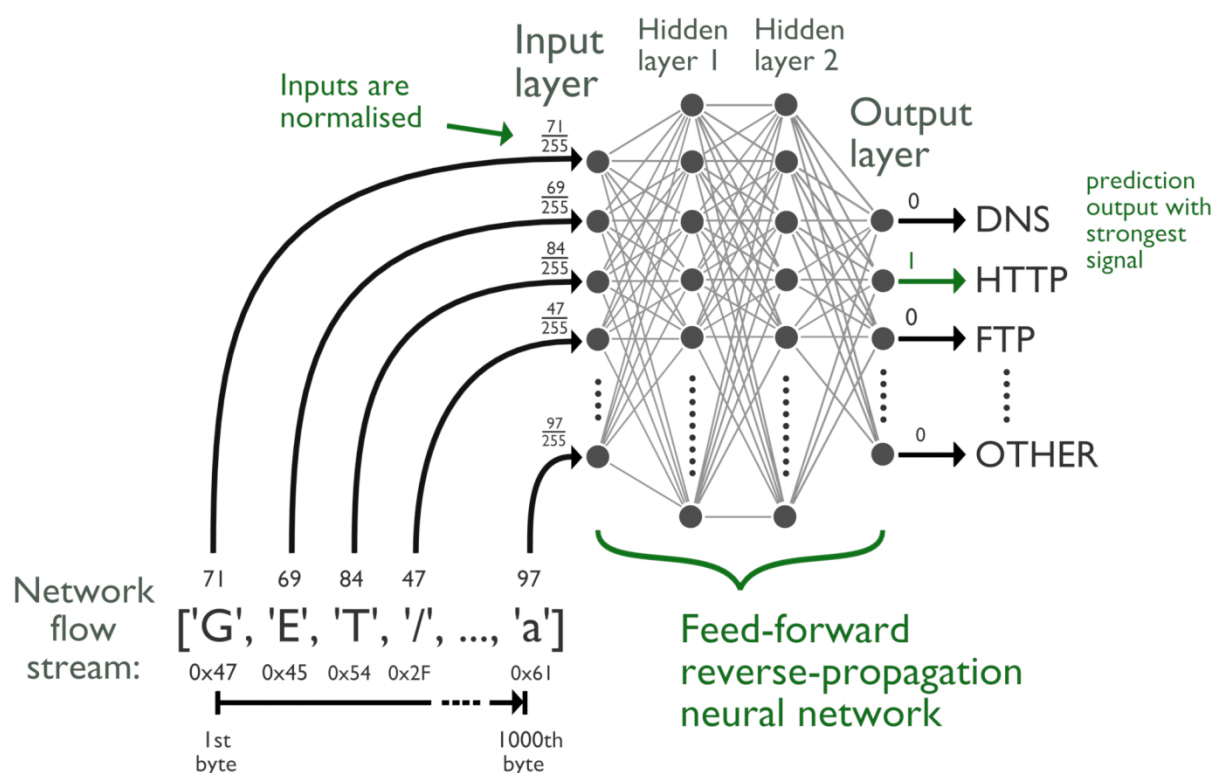
## 5. Data representation

The format of the input data directly affects the results achievable from a neural network.

Wang (2015) conducted similar research in which the first 1000 bytes of each bi-directional



flow were collected from a different network traffic dataset and bi-directional flows were classified. The first 1000 bytes were therefore also inserted as input values into our neural network model and output classifications were produced, through a method outlined in Figure 2. Wang showed however that the most influential bytes were generally within the first 300 bytes, and as such, varying input data sizes below 1000 bytes was also investigated.



**Figure 2:** Presentation of network flows to the neural network. The input to the neural network is the start of a HTTP GET flow and the model's output prediction is HTTP.

## 6. Choosing a deep neural network architecture

This research aims to provide an extensive analysis of deep learning performance on traffic classification using a multitude of feedforward reverse-propagation neural networks. To achieve these comparisons, four model parameters were adjusted sequentially in order to find an optimal architecture (refer to Table 2). TensorFlow was utilised to build and evaluate

the different model architectures ([Abadi et al., 2016](#)). It should be noted that parameters associated with the classification model are referred to in this paper as hyperparameters to avoid confusion with parameters associated with the underlying system.

**Table 2:** Neural network hyperparameters

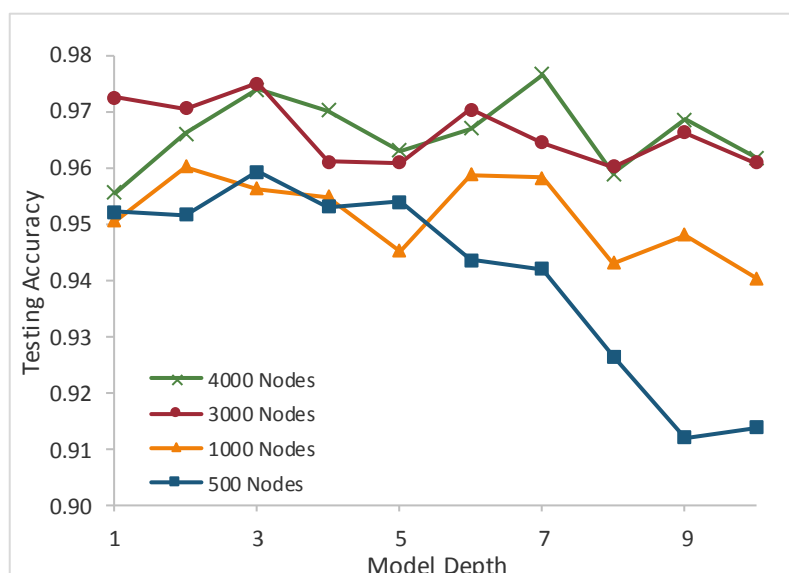
Hyperparameter	Description and Significance
Number of hidden layers	Each hidden layer transforms its input data into a reduced set of features that contain more relevant information from the input data. Figure 1 shows a deep neural network consisting of three hidden layers.
Number of nodes in the hidden layer(s)	Hidden nodes are responsible for feature discovery. Figure 1 shows a deep neural network consisting of a combined 27 nodes in its hidden layers.
Input Length	The number of bytes used from each bi-directional flow. As each input byte maps to a respective node in the input layer, the number of bytes used also determines how many nodes are present in the input layer. For example, Figure 1 would be trained on the first eight bytes of a bi-directional flow.
Padding Style	Padding a flow with either zeros or random values such that each flow meets the required input length.

### 6.1 Number of hidden layers

In a fully connected neural network, the output of every node in one layer is connected to the input of every node in the next. As features propagate through each layer their complexity is increased, which allows for deeper networks to make predictions based upon higher order analysis than their shallower counterparts. As stated by [Sontag \(1992\)](#), neural networks with a non-linear threshold function can always achieve full generality with just two hidden layers, although increasing the number of hidden layers further can also increase the learning rate in specific applications ([LeCun et al., 1989](#)).

To investigate the effect in which the depth of the model had on its ability to classify network traffic, 10 models were constructed each with the same total number of nodes in their hidden layers but spread across varying depths. These depths ranged from one to ten hidden layers. This experiment was then repeated, varying the number of total nodes seen in the model.

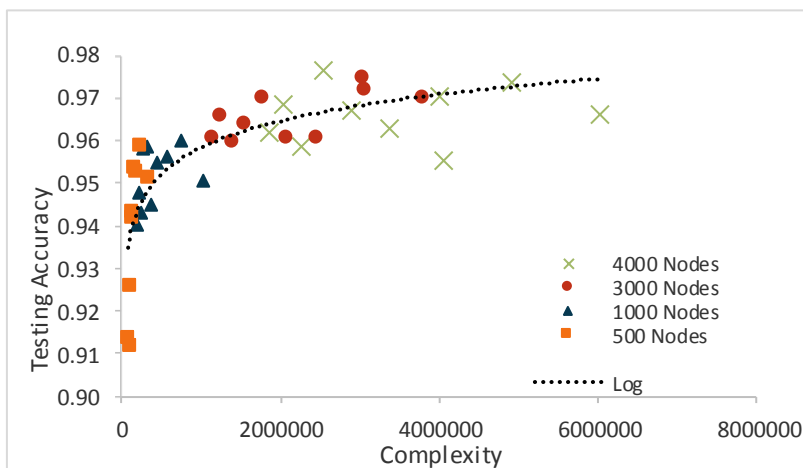
The results from this experiment have been shown in Figure 3. From this, it can be seen that the addition of more layers does not have a conclusive effect on the testing accuracy achieved and in many cases, reduces the testing accuracy. This result shows that the addition of more nodes to the network has a greater effect on the model's accuracy than the addition of hidden layers. To explore this result, the testing accuracy was then compared against model complexity. For a fully connected feedforward model, model complexity is simply defined as the number of connections made within the model and is directly affected by the depth of the model and the number of nodes in the model.



**Figure 3:** Testing accuracy against network depth

It may be observed that there exists a rough logarithmic correlation between the model complexity and testing accuracy (Figure 4). This result seems to suggest that there exists a cut-off point in which the complexity of the model isn't sufficient enough to encapsulate all of the important input features.

To examine the effect in which model complexity had upon its training time, a series of models were created up to a depth of three hidden layers and ran for 10 epochs. Figure 5 indicates that for increasingly complex networks, deeper networks become more efficient to train. Due to the fully connected structure, deeper networks can contain more connections within the model with a smaller number of nodes. As each node acts as a small processor, using less nodes greatly affects the training time. For this reason and the outcomes stated by Sontag (1992), a two-layered model has been chosen for the subsequent experimentation.



**Figure 4:** Testing accuracy against model complexity

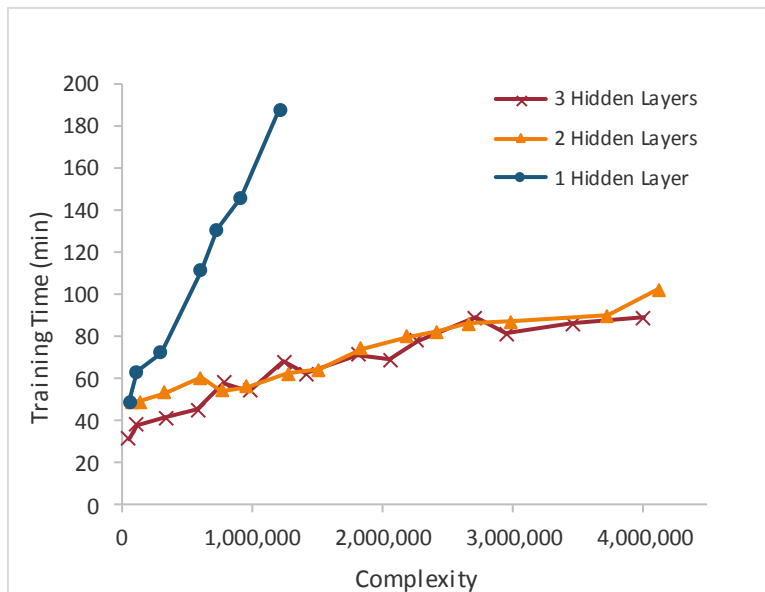
## 6.2 Nodes in the hidden layer(s)

Increasing the number of nodes in the hidden layers can result in overfitting (modelling the noise instead of the underlying relationship) while not having enough nodes can lead to underfitting (when the model cannot capture the underlying relationship). This is due to the number of connections inside the model increasing as the number of nodes increase. Each connection has a corresponding weight that can be adjusted by the neural network's training algorithm. The total number of variables that can be adjusted by the training algorithm is called the degrees of freedom. For any given feedforward reverse-propagation neural network the degrees of freedom can be expressed as:

$$\text{Degrees of freedom} = \sum_i^{N-1} L_i \times L_{i+1} \quad (1)$$

where  $L_i$  is the number of nodes in node layer  $i$ , and  $N$  is the total number of layers in the neural network ( $N = \text{input layer} + \text{hidden layers} + \text{output layer}$ ).

“In most situations, there is no way to determine the best number of hidden units without training several networks and estimating the generalization error of each.” (Swingler, 1996). For this reason, the neural networks defined in the [Appendix](#) each had varying number of nodes in their hidden layers. Each model was trained and then evaluated with the testing dataset, with preliminary results showing models with higher degree of freedom having 0.7% increase in testing accuracy.



**Figure 5:** Training time against model complexity

### 6.3 Input Length

The number of inputs to the neural network is determined by how many bytes from each bi-directional flow is passed into the neural network. If the input length is too small then the model will be unable to detect patterns in the input, resulting in poor training and testing accuracies. As the input length is increased so does the degrees of freedom in the model. However, increasing the input length too much can result in the neural network detecting patterns in the redundant data that do not reflect the output categories.

The lower boundary for input length was determined by visually inspecting samples from the dataset to ensure each output category was visually distinguishable from one another. Figure 6 shows value of the first 1000 bytes across two different application categories. This figure indicates that different applications have noticeably distinct patterns in the first 1000 bytes.

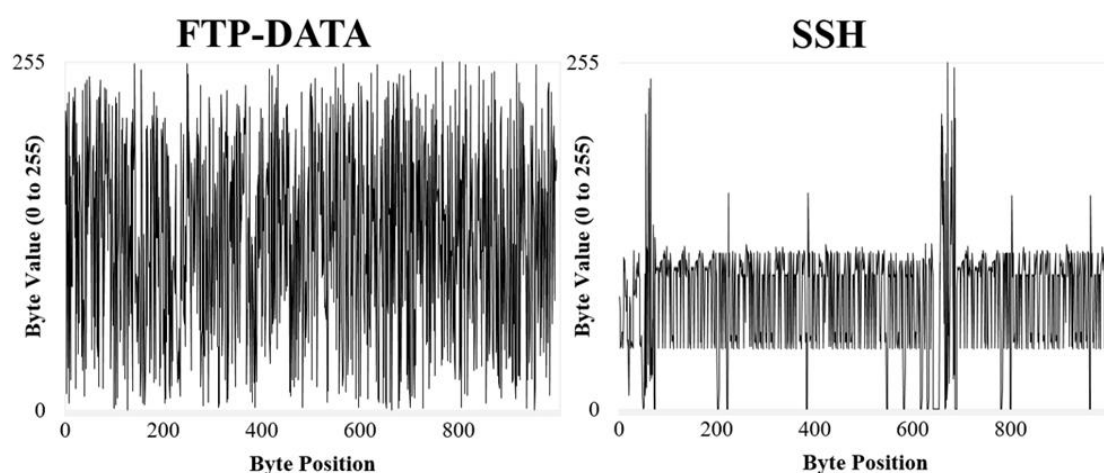


Figure 6: Flow composition: first 1000 bytes

Another one of our preliminary studies investigated how the testing accuracy of models changes as the input length increases. The result is shown in Figure 7. It shows that models with low degrees of freedom are affected more heavily as the input length is increased, while models with a higher degree of freedom remain fairly constant.

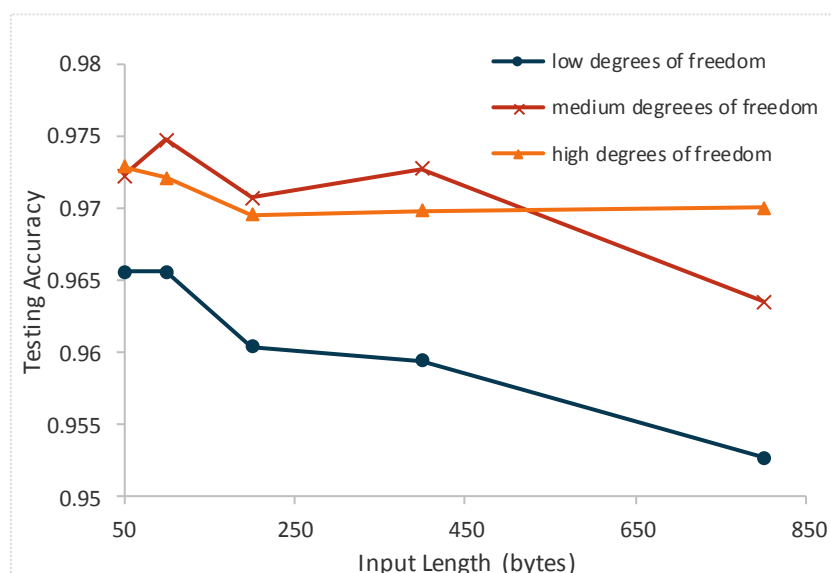
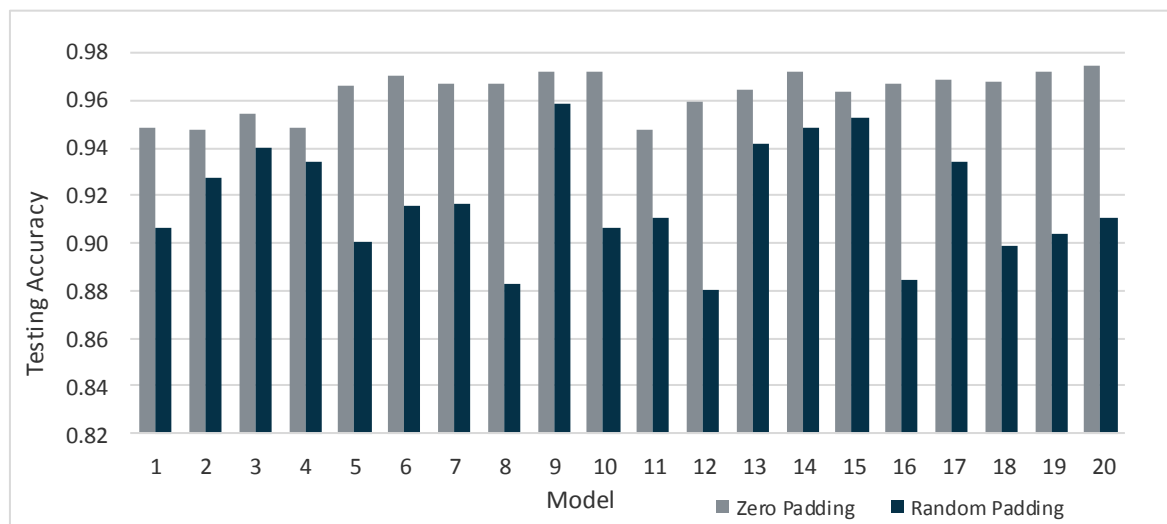


Figure 7: Testing accuracy against input length

### 6.4 Padding style

For flows that did not meet the 1000-byte input length, zero and random padding were investigated. The two methods fill the remaining positions with either a 0, or a pseudorandom value between the ranges of 0 to 255, respectively. Neural networks look for patterns in the input data to help the classification prediction, therefore random padding is used to reduce the chance that the model will learn from a pattern it finds in the padded region of a flow.

Twenty models of various complexities were then tested with both zero and random padding. The findings, shown in Figure 8, indicate that for all models zero padding achieves a much greater classification accuracy than random padding. As it is clear where zero padding starts, a part of this result is speculated to come from the added feature of flow length which the model could be learning from.



**Figure 8:** Zero Vs Random Padding



## 7. Results

From the previous experiments, a set of optimal hyperparameters was found for the specified classification task. Of note:

- It was seen that there exists a threshold for the complexity of the model for which the accuracy will sharply decline if not met.
- Accurate classification can be made using just the first 50 bytes of a flow.
- Zero padding results in consistently greater accuracy than random padding.

Table 3 shows the chosen model's hyperparameters selected based of the preliminary experiments. The classification performance of this model was then investigated. Application and malicious classification was considered separately.

**Table 3:** Chosen model's hyperparameters

Hyperparameter	Chosen
Number of hidden layers	2
Number of nodes in each hidden layer	1600
Input Length	50 bytes
Padding Style	Zero Padding
Category Encoding	1-of-C
Training/Optimization algorithm	Adam
Threshold functions	ReLU and Softmax at output layer

## 7.1 Application results

Figure 9 shows the individual class classification accuracy achieved by the model. The model achieved a high accuracy classification for each of the chosen protocols. As FTP-DATA was the least represented protocol within the dataset it is speculated that the lower accuracy seen could be improved if more samples were present within the dataset.

		Predicted										
		DNS	FTP	FTP-DATA	MAIL	SSH	P2P	NFS	HTTP	BGP	OSCAR	OTHER
Actual	DNS	100.00%	0.00%	1.38%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	FTP	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	FTP-DATA	0.00%	0.00%	91.28%	0.05%	0.00%	0.00%	0.00%	0.57%	0.00%	0.00%	25.90%
	MAIL	0.00%	0.00%	0.78%	98.75%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	SSH	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
	P2P	0.00%	0.00%	1.16%	0.00%	0.00%	99.81%	0.00%	0.37%	0.00%	0.07%	0.00%
	NFS	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.39%	0.00%	0.00%	0.00%	0.00%
	HTTP	0.00%	0.00%	0.27%	1.21%	0.00%	0.00%	0.00%	98.70%	0.00%	0.00%	5.58%
	BGP	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%	0.00%
	OSCAR	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	96.31%	0.00%
	OTHER	0.00%	0.00%	5.14%	0.00%	0.00%	0.19%	0.61%	0.36%	0.00%	3.63%	68.52%

Figure 9: Confusion matrix application classes

## 7.2 Malicious results.

In a similar fashion to the application results, the classification accuracy of the malicious classes can be seen in the confusion matrix in Figure 10. Four malicious classes, Shellcode, Analysis, Backdoor and Worm, were combined in this analysis as they were heavily under represented within the dataset. It is seen that this method of traffic classification produces a subpar class classification accuracy. As malicious content often tries to obscure its intent, it is speculated that statistical data about the flow would achieve a greater classification accuracy when used as inputs to a similarly designed neural network. The major limitations with deep learning stem from access to a large and compressive dataset. It is speculated that with access to such a dataset these accuracies could be greatly improved upon.

		Predicted						
		EXPLOITS	FUZZERS	GENERIC	DOS	RECONNAISSANCE	OTHER MALICIOUS	NON-MALICIOUS
Actual	EXPLOITS	55.27%	6.43%	53.41%	43.06%	4.18%	8.86%	1.05%
	FUZZERS	2.84%	59.05%	3.91%	11.16%	0.00%	0.00%	2.40%
	GENERIC	12.26%	3.21%	23.10%	2.96%	9.09%	10.88%	1.22%
	DOS	11.47%	11.22%	7.83%	21.31%	3.03%	0.53%	1.07%
	RECONNAISSANCE	5.59%	1.48%	3.91%	0.00%	75.26%	0.00%	0.05%
	OTHER MALICIOUS	6.40%	2.89%	3.91%	5.92%	3.03%	35.30%	1.31%
	NON-MALICIOUS	6.18%	15.71%	3.91%	15.58%	5.41%	44.43%	92.89%

Figure 10: Confusion matrix malicious classes

While this method did not prove favourable for detecting the specific malicious flows, it did show promising results for detecting whether a flow was malicious or not. This result is shown in Figure 11.

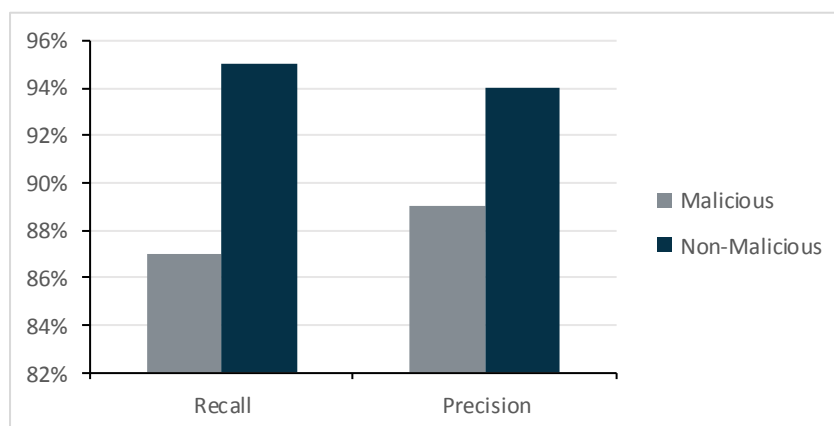


Figure 11: Binary classifier

### 8. Conclusions and future work

Achieving classification accuracies of over 90% for applications protocols and a malicious detection of rate of 87%, our results signify that deep learning can provide an effective and efficient method for the classification of general network traffic and shows promise for

detecting malicious traffic. In this paper, several hyperparameters were analysed to acquire an optimal deep learning architecture, specifically:

- Two hidden layers were used to decrease training time while still meeting the complexity threshold.
- Padding flows that did not meet the required input length with zeros rather than random values resulted in a consistently higher classification accuracy.
- An input length of 50 bytes, instead of the 1000 used in Wang (2015), granted an increased classification accuracy and a reduction in the training time for the used dataset.
- It is speculated that given a more extensive dataset, the classification accuracies could be improved on and extrapolated onto a wider range of network traffic protocols.

In the future, supplementary network statistical data shall be utilised to improve the classification accuracy for malicious flows. Another avenue of research involves using convolutional neural network (CNN) to further enhance deep neural network traffic classifiers. With additional research, the authors of this paper believe that deep learning will become the pinnacle of network traffic classification in the future.

## References

- [Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., . . . Isard, M. \(2016\).](#) *TensorFlow: A System for Large-Scale Machine Learning*. Paper presented at the OSDI.
- [Auld, T., Moore, A. W., & Gull, S. F. \(2007\).](#) Bayesian neural networks for internet traffic classification. *IEEE Trans Neural Networks*, 18(1), 223-239. doi:10.1109/TNN.2006.883010
- [Dong, B., & Wang, X. \(2016\).](#) *Comparison deep learning method to traditional methods using for network intrusion detection*. Paper presented at the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN).
- [LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. \(1989\).](#) Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4), 541-551. doi:10.1162/neco.1989.1.4.541
- [Nour, M., & Slay, J. \(2015\).](#) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Military Communications and Information Systems Conference (MilCIS)*.
- [Nour, M., & Slay, J. \(2016\).](#) The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 1-14.
- [Singh, K., Agrawal, S., & Sohi, B. S. \(2013\).](#) A Near Real-time IP Traffic Classification Using Machine Learning. *International Journal of Intelligent Systems and Applications*, 5(3), 83-93. doi:10.5815/ijisa.2013.03.09
- [Sontag, E. D. \(1992\).](#) Feedback stabilization using two-hidden-layer nets. *IEEE Transactions on Neural Networks*, 3(6), 981-990. doi:10.1109/72.165599
- [Swingler, K. \(1996\).](#) *Applying neural networks: a practical guide*: Morgan Kaufmann.
- [Trivedi, C., Chow, M.-Y., Nilsson, A. A., & Trussell, H. J. \(2002\).](#) Classification of Internet Traffic using Artificial Neural Networks
- [Wang, Z. \(2015\).](#) The Applications of Deep Learning on Traffic Identification. *Black Hat USA*.
- [Yang, F. \(2015, 27-29 May 2015\).](#) *The tale of deep packet inspection in China: Mind the gap*. Paper presented at the 2015 3rd International Conference on Information and Communication Technology (ICICT).

**Appendix:** Number of hidden nodes across the 25 different models

Model Num.	Num. of Nodes		
	Hidden Layer 1	Hidden Layer 2	Hidden Layer 3
0	10		
1	20		
2	50		
3	100		
4	2500		
5	5000		
6	10000		
7	200	100	
8	500	500	
9	1000	500	
10	500	1000	
11	1000	1000	
12	1500	1000	
13	1500	1500	
14	200	100	100
15	500	500	500
16	700	500	300
17	1500	200	200
18	500	1000	500
19	300	500	700
20	500	500	1000
21	750	750	750
22	1000	1000	1000
23	2500	500	500
24	2000	2000	2000
25	5000	5000	5000