Deep Learning-based Intrusion Detection for IoT Networks

Mengmeng Ge*, Xiping Fu†, Naeem Syed‡, Zubair Baig*, Gideon Teo§, Antonio Robles-Kelly*
*School of Information Technology, Deakin University, Geelong, VIC, Australia
† Telstra Network Services NZ Limited, Christchurch, New Zealand
‡School of Science, Edith Cowan University, Joondalup, WA, Australia
§School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand
mengmeng.ge@deakin.edu.au, fxpfxp0607@gmail.com, n.syed@ecu.edu.au, zubair.baig@deakin.edu.au,
teogideon@gmail.com, antonio.robles-kelly@deakin.edu.au

***Abstract*—Internet of Things (IoT) has an immense potential for a plethora of applications ranging from healthcare automation to defence networks and the power grid. The security of an IoT network is essentially paramount to the security of the underlying computing and communication infrastructure. However, due to constrained resources and limited computational capabilities, IoT networks are prone to various attacks. Thus, safeguarding the IoT network from adversarial attacks is of vital importance and can be realised through planning and deployment of effective security controls; one such control being an intrusion detection system. In this paper, we present a novel intrusion detection scheme for IoT networks that classifies traffic flow through the application of deep learning concepts. We adopt a newly published IoT dataset and generate generic features from the field information in packet level. We develop a feed-forward neural networks model for binary and multi-class classification including denial of service, distributed denial of service, reconnaissance and information theft attacks against IoT devices. Results obtained through the evaluation of the proposed scheme via the processed dataset illustrate a high classification accuracy.**

***Index Terms*—Internet of Things; Intrusion Detection; Feed-Forward Neural Networks; Denial of Service Attacks;**

I. Introduction

Cyber security for the Internet of Things (IoT) paradigm is of high significance in modern-day Information Technology/-Operations Technology (IT/OT) systems. The two factors that have led to an exposure of IoT vulnerabilities to cyber threats are: large-scale proliferation of IoT devices from within households all the way to critical infrastructures including smart electricity grids and smart vehicles, and the heterogeneity in the communication protocols of IoT devices. The number of IoT devices in use have seen a steep rise from 15.4B in 2015 to 26.7B in 2019, with numbers continuing to climb as households and businesses alike are increasingly becoming dependent on Internet-based services for their routine activities.

The global IoT market share is illustrated through the following figures: smart cities account for 28.6%, industrial IoT at 26.4%, connection health at 22%, smart homes at 15.4% and connected cars at 7.7% [1]. The increasing diversity in the types of IoT devices that are manufactured for various application domains (as given above), implies that the IoT manufacturing industry is fast progressing to advance the IoT technology and to reduce the time to market for their products. Consequently, the standards that dictate the IoT device communication are diverse, requiring a common platform to facilitate the inter-device communication. In addition, the numbers of tools and experimental platforms that are available to run the IoT device and network-level simulations and experiments, are also ever-increasing [2]. One such protocol, namely, the Message Queuing Telemetry Transport (MQTT), is an IoT connectivity protocol, which is a lightweight publish-subscribe protocol to facilitate the remote connectivity between IoT devices and centralised brokers or base stations [3]. Due to its lightweight nature, that facilitates low bandwidth, low power and low storage on IoT devices, the MQTT protocol can be easily deployed on mobile applications as well. The MQTT protocol suffers from many vulnerabilities that can be exploited by the adversary. Some of the threats against the protocol include: device compromise, data privacy, Denial of Service (DoS) against MQTT service, and Man-in-The-Middle (MiTM) attacks against MQTT messages.

The application of artificial intelligence (AI) for the timely detection of malicious attacks, is both necessary as well as effective. The range of AI techniques applicable for network intrusion detection (i.e., for detecting malicious attacks through analysis of network traffic) is very broad. Various AI techniques such as support vector machines (SVM) [4], Bayesian networks [5], principal component analysis (PCA) [6] and genetic algorithms (GA) [7] have been popularly deployed for the detection of patterns of anomalous behaviour in an IoT network. However, deep learning based intrusion detection for IoT networks is still somewhat under-researched.

In this work, we propose a feed-forward neural network classifier for detection of cyber attacks in IoT networks, with four categories of attacks assessed through our experiments, namely, distributed denial of service (DDoS), denial of service (DoS), reconnaissance, and information theft, whilst differentiating these from the legitimate network traffic. The main contributions of the paper are as follows.

- Adoption of a newly published IoT dataset with realistic attack traffic and simulated IoT traffic in a smart home

2473-3105/19/$31.00 ©2019 IEEE
DOI 10.1109/PRDC47002.2019.00056

scenario;
- Generation of generic features directly based on the information from header fields in individual IP packets to avoid the attack oriented feature selection procedure;
- Development of the deep learning based intrusion detection approach including both binary and multi-class classifications for IoT networks with high accuracy.

The rest of the paper is organised as follows. In Section II, we present the related work on intelligent detection of cyber attacks for both legacy as well as IoT networks. In Section III, we present the proposed framework for the intrusion detection in IoT networks through the application of feed forward neural networks. In Section IV, we describe the experimental testbed, data preprocessing steps and the analysis of results obtained. We present a pathway for future work and also discuss our findings in Section V. The paper is concluded in Section VI.

II. Related Work

In this section, we discuss schemes that have been developed to secure the MQTT protocol and existing techniques that have been proposed in the literature for detecting malicious cyber attacks in both traditional and IoT networks through the application of AI-based techniques.

Securing MQTT: in [8], a scheme based on the AugPAKE protocol was presented for securing the MQTT protocol. Unlike TLS/SSL protocols, AugPAKE is not reliant on certificate validation and revocation checks, simplifying the preliminary setup of the MQTT protocol, that requires the establishment of publishers and subscribers in an IoT network, to enable subsequent communication. The scheme helps to prevent passive attacks against the MQTT security, that involve password guessing over the communication channel, in order to gain an illicit system access. The overall performance of the security of the scheme was found to be better than public key-based schemes.

In [9], MQTT vulnerabilities exposed through the adversarial capability of sniffing the broker messages that are transmitted in the same network as the adversary, were presented. As the publisher issues these messages with usernames and passwords transmitted in plaintext, sniffed messages can reveal authentication strings, thus allowing the adversary to gain an illegitimate system access. The authors also highlighted the need for TLS to ascertain transport layer security and to prevent disclosure of authentication strings to the adversary.

In [10], the authors presented a range of models of DoS attacks against the MQTT protocol. The TCP SYNC attack was modelled and tested for a MQTT-based IoT network, and the effectiveness of the four DoS attack types against IoT resources, was reported. The authors were able to conclusively state that the effect of the DoS attack was more on the network bandwidth than on the end-node (i.e., the victim IoT node). Moreover, the vulnerability of the MQTT broker node, exploitable by the adversary, was highlighted.

AI-based detection techniques for traditional networks: feed forward neural networks [11] comprise several neurons that are structured in layers; most commonly into an input, a hidden and an output layer. The training phase of the neural network incorporates various strategies and mathematical operations to find the most optimal weights of the interconnections between the three layers, so as to find the desired output. The algorithm that is adopted to find these interconnection weights is the back propagation algorithm. The authors in [11] presented a feed forward neural network-based scheme to detect four attack types of the DARPA dataset [12], namely, DoS, root to local (R2L), user to root (U2R) and probe attacks; with results indicating a high attack detection rate of more than 96% for three out of four attack types.

In [13], an ensemble of classifiers was presented for detecting DDoS attacks. The dataset of network traffic was split into various subsets; each subset was trained with an ensemble of classifiers, and the results were combined through the application of weighted majority voting. The resilient back propagation algorithm was adopted as the classifier. An accuracy of 99.4% was reported for the proposed scheme when tested on several publicly available datasets.

In [14], a radial basis function based scheme to detect DoS attacks was presented that comprises several layers. The radial basis function is the first computational layer, that works alongside the cumulative incarnation of the weight optimization technique, so that the training bias is reduced. An accuracy of 99.69% was reported when the scheme was tested on the NSL KDD [15] and the UNSW-NB 15 [16] datasets. In a related work, Shone et al. [17] have applied a deep network composed of a stack of deep auto encoders for unsupervised feature learning combined with a random forest classifier for the detection of network intrusion in the NSL KDD [15].

In [18], a deep learning model was presented for the detection of network intrusion on the UNSW-NB 15 dataset. The model consists of five hidden layers with ten neurons across each hidden layer. The training was performed on the full dataset in 10 epochs with 10-fold cross validation, and the importance of features was ranked based on the Gedeon method. An accuracy of close to 99% was reported for the proposed scheme.

AI-based detection techniques for IoT networks: in [19], the authors generated an IoT dataset, namely Bot-IoT, which consists of legitimate and simulated IoT traffic and attack traffic (including DDoS, DoS, reconnaissance and information theft) based on a real testbed and compared the dataset with other publicly available datasets. Among the comparison, the proposed dataset was claimed to be the only dataset containing IoT traces. The authors also developed new features using correlation coefficient and joint entropy techniques and proposed three machine learning and deep learning algorithms to detect attacks in the extracted dataset (i.e., 5% of the original dataset). Evaluation results of the classifiers were presented which demonstrate good accuracy. However, The extracted dataset contains imbalanced normal and attack traffic because the number of attack packets is much higher than that of normal packets for three attack types (DDoS, DoS and reconnaissance).

In summary, there is very little research focusing on deep learning based intrusion detection solutions for IoT networks. One critical reason could be the lack of reliable IoT datasets with the inclusion of attack traffic. In this paper, we adopt the newly published Bot-IoT dataset in [19] and develop a new deep learning based detection approach with generic features in packet level to classify normal traffic and different types of attack traffic in the dataset.

III. Proposed Framework

We provide a high-level overview of the framework for the intrusion detection in IoT networks. The framework is shown in Figure 1 which mainly includes four phases: feature extraction, feature preprocessing, training and classification. Each phase is explained in details in the following.

Feature extraction: at first, we collect the raw traffic from the network analyzer tool. Afterwards, we extract relevant fields from individual packets. Each extracted field refers to a feature. We do not calculate features based on aggregated packets but use the field information of the individual packet as features. Therefore, we try to capture generic features of the traffic instead of generating attack oriented features which are intended for capturing specific attack behaviours. In specific, we mainly focus on header fields in the IP packet, which include frame, IPv4/IPv6 and TCP/UDP related information.

Feature preprocessing: we then preprocess the extracted field information by merging, dropping and encoding column values and store the results in arrays. We split the processed data into training traffic (used in the training phase) and testing traffic (used in the classification phase). Packets in both training and testing traffic contain labels of being either malicious or normal; for malicious packets, each packet is also labelled as the specific attack it belongs to. Labels in testing traffic are only used for checking the prediction accuracy.

Training: in the training phase, we take the processed data (i.e., arrays) from the training traffic into the deep neural networks model for training. The trained neural networks model is used as the classifier when new traffic comes in. The classifier could be used for both binary and multi-class classification.

Classification: in the classification phase, we apply the classifier on the processed data from the testing traffic which outputs either normal or malicious label for the individual packet. For the multi-class classification, the classifier labels the malicious packet as the specific attack it belongs to.

IV. Evaluation and Analysis

We provide a detailed description of the dataset and discuss the processing of the dataset in terms of label mapping and dataset extraction, feature extraction and preprocessing, experiment setup and analysis of results.

A. Dataset description

We use the BoT-IoT dataset produced by Koroniotis et al. [19] in the Cyber Range Lab of The center of UNSW Canberra Cyber. The authors developed a testbed to collect legitimate and malicious traffic from simulated IoT devices. The testbed includes attack and target virtual machines (VMs), network devices (e.g., pfSense firewall), simulated IoT devices running on normal VMs and being connected to the IoT hub of the Amazon Web Services (AWS). There are five IoT scenarios forming a smart home configuration, including a weather station, a smart fridge, a smart thermostat, a remotely controlled garage door and smart lights. Simulated IoT devices publish data via the MQTT procotol to the MQTT broker where MQTT is a lightweight protocol for IoT solutions. Malicious activities generated in the dataset are listed as follows with four categories: probing attacks (port scanning and OS fingerprinting), DoS and DDoS over TCP, UDP and HTTP and information theft (data exfiltration and keystroke logging). In total, there are 10 subcategories of the attacks. We present an overview of the dataset in terms of the number of packets in each subcategory in Table I. The dataset contains a good range of attacks and a reasonable amount of both normal and attack traffic. The detailed description of the testbed setup and statistics of attacks could be found in the paper [19].

TABLE I: Statistics of the dataset.

Category	Subcategory	Number of packets
DDoS	HTTP	194417
	TCP	95643746
	UDP	174344218
DoS	HTTP	287480
	TCP	59977444
	UDP	191407421
Reconnaissance	OS fingerprinting	827940
	Service scanning	3850942
Information theft	Data exfiltration	251108
	Keylogging	11389
Normal	Normal	31965724
Total		558761829

The reasons to adopt the BoT-IoT dataset include the configuration of real testbed, collection of realistic attack traffic, generation of simulated IoT traffic in a smart home scenario and inclusion of labelled data.

B. Dataset processing

In the BoT-IoT dataset, the original traffic was collected in the PCAP files. The labelled traffic is stored in two formats: Argus files processed by the Argus tool from the PCAP files and CSV files converted from the Argus files. However, the Argus tool is applied to generate network flows which are aggregated packets in the PCAP files. Thus, in order to perform the packet-level detection in our framework, we need to retrieve labels on individual packets.

We started with the collection of the PCAP files. A total of 344 PCAP files from the BoT-IoT dataset were obtained. The PCAP files were organised according to 10 subcategories of attacks namely: DDoS/DoS over HTTP, TCP and UDP, service scan, OS fingerprinting, Keystroke logging and data exfiltration.

Feature extraction: individual packet information was extracted from the PCAP files using the TShark tool and output into the CSV files. Packet fields extracted from the PCAP files are shown as follows. We consider IP and ARP packets. These

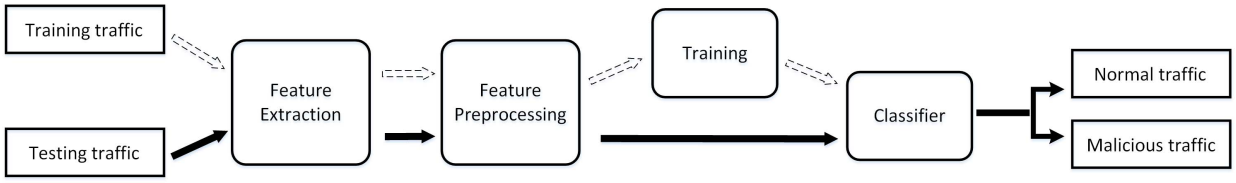


Fig. 1: Workflow of the framework.

fields convey different information of an individual packet. Additional field information could be further investigated and extracted to be used as features.

- Frame related fields: frame.time_epoch, frame.len
- ARP related fields: arp.proto.type, arp.src.proto_ipv4, arp.dst.proto_ipv4
- IP related fields: ip.proto, ip.src, ip.dst, ip.ttl, ipv6.src, ipv6.dst
- TCP related fields: tcp.srcport, tcp.dstport, tcp.stream, tcp.analysis.initial_rtt, tcp.time_delta, tcp.len, tcp.flags, tcp.window_size_value, tcp.checksum
- UDP related fields: udp.srcport, udp.dstport, udp.stream, udp.checksum, udp.length

Label mapping: after the feature extraction, the labelling process was completed by analysing flows in the original labelled dataset and replicating same labels for all packets within individual flows. In specific, packets belonging to different flows were generated by four bot machines as described in the BoT-IoT dataset and were labelled according to the attack subcategory. A script was developed to match the IP address of the bot machine and label packets according to the attack subcategory. Labelled packets are stored in the CSV files for further processing. As the attacks were launched at different times with normal traffic generated in the background, there is no overlapping of two different attacks during the same time period. Generated CSV files with labelled traffic are placed in different folders based on the subcategory of the attacks.

Schema unifying: we converted the CSV files to Apache Parquet files which reduce the total size of the dataset from 16GB to 300MB and improve the processing speed. We unified the schema of all files regarding the number and type of fields in the schema. In specific, for files containing the mix traffic from DoS attacks over TCP and UDP and normal communication, there are no IPv6 related columns. We added two columns when merging the Parquet files and filled in NULL values. In addition, some ICMP packets contain the UDP packet inside the ICMP payload due to the recognition of the embedded packet by the network protocol analyzer. Thus two columns (i.e., ip.proto and ip.ttl) could be either string type (i.e., two values separated by comma) or double type. We decided to keep both values as this could be a feature of the attack. We adjusted all values in those two columns to string type for consistency. Finally, we combined all Parquet files to one single Parquet file.

Dataset extraction: by using the Parquet file with unified schema, we extracted roughly 2% of the dataset. For each subcategory of attacks, we used 1 million packets if the number of packets is larger than 1 million or used all packets when the number of packets is lower than 1 million. For the normal traffic, we calculated the number of packets for 2% of the full dataset and deducted that number by the total number of malicious packets. In specific, for each attack with more than 1 million packets, the random selection of 1 million packets was implemented through choosing a fraction of total packets (i.e., the fraction is 1 million divided by the total number of packets). Therefore, the number of the chosen packets fluctuates slightly either above or below 1 million. The number of packets extracted is 11174616 including mix of normal and malicious traffic.

C. Feature preprocessing

After the processing of the dataset and feature extraction, we could preprocess the features to be used in the deep learning model.

We uploaded the Parquet file with extracted features to Google's Colaboratory which allows us to use one free Tesla K80 GPU wit 12GB of RAM. From the observation of the packets, we selected fields that can be used as inputs into the deep neural networks model. We dropped columns of frame.time_epoch, IPv4/IPv6 address related columns (i.e., ip.src, ip.dst, ipv6.src, ipv6.dst, arp.src.proto_ipv4 and arp.dst.proto_ipv4), and checksum columns (i.e., tcp.checksum and udp.checksum). In order to reduce the size of the inputs, we merged five pairs of columns respectively as values in those two columns of each pair do not overlap with each other. The five pairs of the columns include: ip.proto and arp.proto.type; tcp.srcport and udp.srcport; tcp.dstport and upd.dstport; tcp.len and udp.length; tcp.stream and udp.stream. New columns are named as proto, src.port, dst.port, length and stream.

We identified columns with categorical data and processed the data to be used for one-hot encoding. In specific, for the protocol column (i.e., proto), due to the comma separated values for embedded packets, we mapped each unique value in string format to an integer value. For the tcp.flags column, We converted hexadecimal values to float values. For the src.port, dst.port columns, we used a list of common ports (i.e., 20, 21, 22, 23, 25, 42, 43, 53, 80, 161, 443) to map uncommon ports to new columns. This list was constructed based on well-known ports for various services, including FTP, SSH, Telnet, DNS, SMTP, HTTP/HTTPS, SNMP. More options of the common ports could be investigated in the future and used for the encoding. For either source or destination port column, we

generated two new columns: one for the port number between the range of 0 and 1023 but does not belong to the list; another for the port number larger than 1023 (which belongs to registered ports or private ports).

In addition, for the tcp.ttl column (ttl means time to live), due to the comma separated values, we split the values into two columns (i.e., tcp.ttl_1 and tcp.ttl_2) while non-embedded packets are filled with NaN values in the second column.

Due to the existence of NaN values, we filled NaN values with numeric values. In specific, for the tcp.flags and stream columns, we obtained the maximum value in each column and filled NaN values with the maximum value plus 1. For other columns, we filled NaN values with 0.

Due to the above processing (i.e., column dropping and merging), duplicated rows were introduced. We dropped the redundant rows because the testing traffic should not have duplicates from the training traffic. Without redundancy, we will be able to check how the model will perform on unseen data in the prediction phase under testing traffic. The statistics of the dataset before and after dropping duplicates rows are shown in Table II.

TABLE II: Statistics of the dataset before and after dropping duplicated rows.

Category	Subcategory	Drop duplicated rows	
		Before	After
DDoS	HTTP	194417	194407
	TCP	1000853	1000850
	UDP	999517	974617
DoS	HTTP	287480	287464
	TCP	1000107	1000103
	UDP	999752	970451
Reconnaissance	OS fingerprinting	827940	823368
	Service scanning	1000932	996110
Information theft	Data exfiltration	251108	33209
	Keylogging	11389	11189
Normal	Normal	4601121	1018778
Total		11174616	7310546

From Table II, we can see that the total number of packets being kept is roughly 65.42% of the extracted dataset while normal packets are being dropped by 77.86%.

Finally, we applied one-hot encoding for the src.port, dst.port, tcp.flags, proto columns and dropped these original columns. We also encoded label columns. We considered binary classification (i.e., classify normal packets and malicious packets from each subcategory of attacks separately) and multi-class classification (i.e., classify normal packets and malicious packets from DDoS/DoS, reconnaissance and information theft attacks). In the binary classification, a normal packet is labelled as 0 while a malicious packet is labelled as 1; in the multi-class classification, packets from each class are labelled from 0 to 3 respectively. Results were stored in arrays, which will be taken as inputs into the deep neural networks model.

D. Experimental Setup

We used TensorFlow library and Keras. All our experiments were effected on Google's Colaboratory with a 64%-16%-20% training, validation and testing split on our dataset. As a preprocessing step, we transformed the inputs by scaling each feature into a given range between (-1, 1) to avoid extreme values. Also, we introduced class weights to the training data due to the imbalanced samples of different classes. The weight for each class was calculated by the inverse of the quotient value from dividing the packet count in a particular class by the maximum packet count among all classes. In this manner, the under-represented class with a smaller number of samples will get a higher weight value.

Here, we consider a feed-forward neural networks (FNN) model. To this end, we use initial random weights for the layers in the FNN model, which are sampled from a truncated normal distribution. For the solver, we choose Adam to update weights and apply a sparse categorical cross-entropy loss function. The architecture of the FNN model along with interactions with the loss function and the solver is shown in Figure 2.

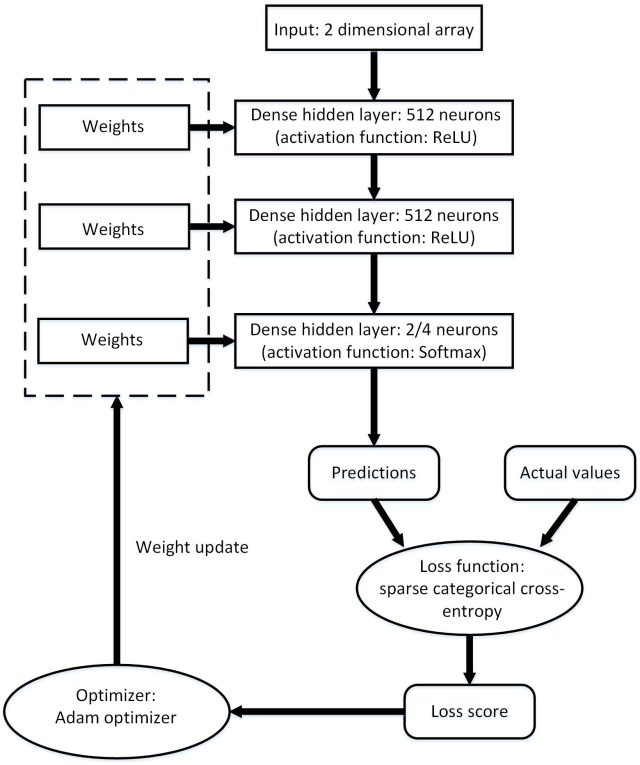


Fig. 2: FNN model with the loss function and optimizer.

The FNN model in Figure 2 takes a two dimensional array as input and consists of three dense hidden layers with each input neuron connected with each output neuron in each hidden layer. The first two hidden layers have 512 neurons in each layer and the last hidden layer has either 2 or 4 neurons depending on the classification type (i.e., binary or multi-class). A ReLU activation function is used for the first two hidden layers and Softmax function is used for the last hidden layer which turns numbers into probabilities that sum to one.

We tuned the model with different parameters and hyperparameters and optimized the model with different regularization techniques using the IoT traffic with multi-class labels. We used the same FNN model (i.e., same hyperparameters and parameters except the number of neurons in the last hidden layer of the neural networks model is 2) for the binary classification because we want to apply the generalization of the model to different cases. We discuss the regularization and tuning procedure for the multi-class classification in details in the following paragraphs.

Regularization: we first experimented with different regularisation techniques including L1, L2 and dropout. However, the classification accuracy does not improve. That said, regularisation is proven effective for preventing over-fitting. Since our model is trained on the dataset with millions of samples, the over-fitting problem is a less concern.

Parameter tuning: we then experimented with different architectures of the neural networks, including adding/reducing one hidden layers (ranging from one to four hidden layers which exclude the last hidden layer as it should contain 4 neurons for the multi-class classification), increasing/decreasing the number of neurons in each hidden layer (including 512 neurons in each hidden layer for one to three hidden layers; or 256, 128, 64, 32 neurons in each hidden layer for a total of four hidden layers; all excluding the last hidden layer). We found the neural network model with three hidden layers and 512 neurons in the first two layers has the best performance. The different variants of the neural network model, however, do not increase the classification accuracy. The possible reason could be that the complexity of the model is either too small or too large. When the complexity of the model is too small, the model is not able to capture the variation of different classes inside the training dataset. When the complexity of the model is too large, the model tends to over-fit different classes inside the training dataset. Therefore, the too large/small complexity of the model would make the performance of the model on specific classification problems worse.

Hyperparameter tuning: lastly, we experimented with different learning rates using Adam, including 0.01, 0.001 and 0.0001. We chose the default learning rate of 0.001 which gives higher accuracy. The larger learning rate tends to oscillate over training epochs, causing weights to diverge, while a smaller learning rate often gets stuck on a sub-optimal solution.

We also applied an early stopping technique with a patience of 5 iterations to terminate the training as soon as the validation error reaches a minimum. This has the advantage of reducing over-fitting. The FNN model was trained with a batch size of 500 and the maximum epoch was set to be 20. We evaluated the model via the test set.

E. Analysis of Results

Here, we discuss results for the binary classification and multi-class classification. We also present confusion matrices for both classification settings. We use following metrics to evaluate the model performance in the binary classification: accuracy, precision, recall and F1 score. Accuracy is defined as the number of correct predictions, including true positive (TP) and true negative (TN) predictions, divided by the total number of predictions; precision is the number of positive predictions divided by the total number of positive class values predicted; recall is the number of positive predictions divided by the number of positive class values in the test data; F1 score is weighted average of precision and recall. A low precision could indicate a large number of false positive (FP) predictions while a low recall could indicate a large number of false negative (FN) predictions. As F1 score combines precision and recall, a good F1 score could indicate low FP and low FN predictions. Definitions of the metrics in terms of positives and negatives are shown in Table III.

TABLE III: Definitions of metrics.

Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1 score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$

Binary classification: we show confusion matrices for the binary classification in Table IV and evaluation metrics in Figure 3. For the evaluation metrics shown in Figure 3, we use the range between 0.9 and 1.0 on the vertical axis, where all values are above 0.9.

We can see the classification for normal traffic and malicious traffic from each subcategory of DDoS/DoS achieves a high accuracy being above 0.999 as well as high precision and recall values (all above 0.99) and high F1 score being above 0.999 which indicate low FP and FN predictions. In specific, detection of DDoS over HTTP/UDP attacks achieves 100% accuracy in the current extracted dataset. In the prediction on DoS over UDP attacks, 222 over 194480 (0.1%) malicious packets ware classified as normal packets. The reason could be that some ICMP packets contain the UDP packet inside the ICMP payload and these packets could be either normal or malicious thus causing confusion to the classifier based on existing generic features.

In addition, the classification for reconnaissance attacks also achieves a high accuracy as well as high precision and recall values and a high F1 score (all above 0.99). In the prediction on service scan attacks, 1201 over 203278 (0.6%) normal packets ware classified as malicious packets while 694 over 199700 (0.3%) malicious packets were classified as normal packets. The reason for this could be the embedded UDP packet inside the ICMP payload and these packets may either be normal or malicious, thus confusing the classifier based on current features.

For attacks under the category of information theft, the classification for data exfiltration and keylogging attacks

TABLE IV: Confusion matrices for the binary classification

Actual/Predicted	Normal	DDoS over HTTP
Normal	204007	0
DDoS over HTTP	0	38630

Actual/Predicted	Normal	DDoS over TCP
Normal	203331	2
DDoS over TCP	0	200593

Actual/Predicted	Normal	DDoS over UDP
Normal	203308	0
DDoS over UDP	0	195371

Actual/Predicted	Normal	DoS over HTTP
Normal	203492	1
DoS over HTTP	0	57756

Actual/Predicted	Normal	DoS over TCP
Normal	203251	1
DoS over TCP	0	200525

Actual/Predicted	Normal	DoS over UDP
Normal	203366	0
DoS over UDP	222	194258

Actual/Predicted	Normal	Service scan
Normal	202077	1201
Service scan	694	199006

Actual/Predicted	Normal	OS fingerprint
Normal	203013	108
OS fingerprint	78	165231

Actual/Predicted	Normal	Data exfiltration
Normal	203207	519
Data exfiltration	4	6668

Actual/Predicted	Normal	Keylogging
Normal	203630	75
Keylogging	7	2282

achieves high accuracy and recall which indicate the capability of the classifier to detect true malicious packets as intrusion. Prediction on both attacks has more than 90% precision (i.e., 92.78% for data exfiltration and 96.82% for keylogging). There are two possible reasons that the model behaved worse in classifying normal traffic and attack traffic from information theft. Firstly, we used all malicious packets belonging to these two attacks. However, the sample size is still small compared with the number of packets from other attacks. Besides, the traffic in these attacks could closely resemble normal network traffic thus making the detection harder. We will investigate and include more relevant fields as features in the future work.

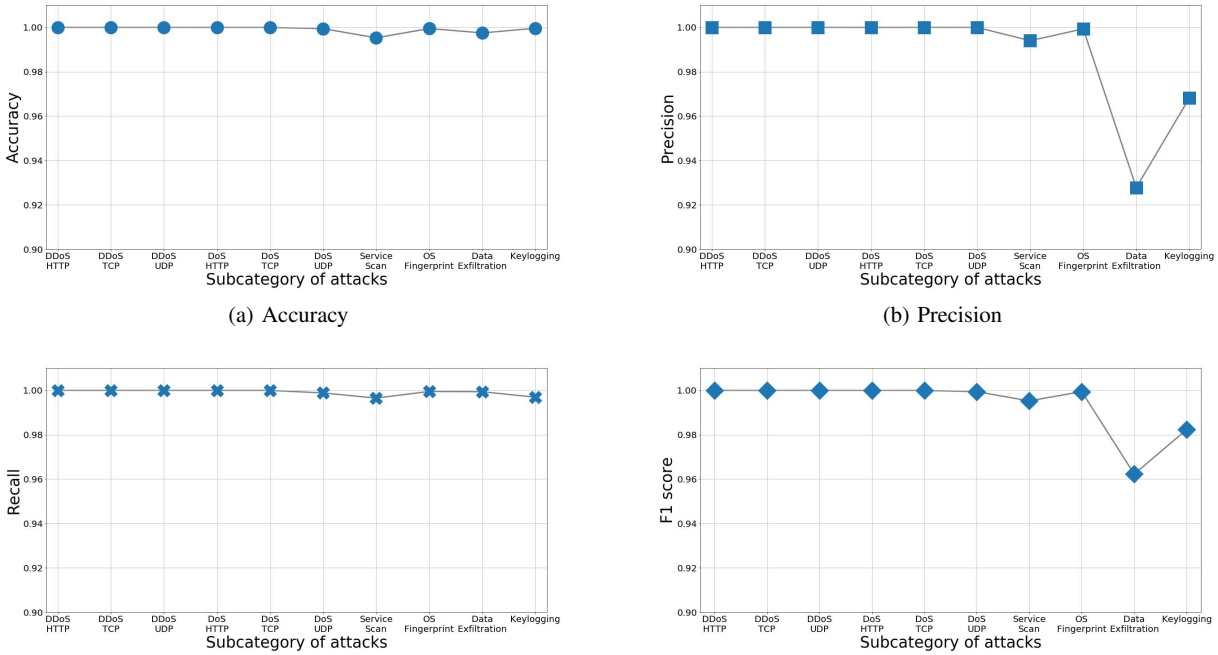
Multi-class classification: we show the confusion matrix for the multi-class classification in terms of packet count and percentage value in Table V. The FNN model achieves a high accuracy of 0.99 for classifying normal packets and malicious packets from three attack categories. For the normal traffic, the model is able to identify 98% of the true normal packets. For DDoS/DoS and reconnaissance attacks, the model is able to detect more than 98% of the real malicious packets as intrusion of correct category. For the information theft attack, the model classifies 88.9% of the packets correctly while fails to differentiate packets from the reconnaissance attack (i.e., 10.9% of the theft traffic was classified into the reconnaissance attack). The reason could be that the number of training packets for information theft is not large enough. Therefore, one approach to increase the prediction accuracy is through increasing the number of packets within this class. Another approach is to increase the corresponding weight of the class. However, this could introduce the risk of over tuning weights for this specific dataset (i.e., weights are designed too specific for the dataset). Besides, there is no misclassification for DDoS/DoS and information theft attacks.

By counting the number of malicious packets which are classified as normal packets, we can see that 319 of them (0.2%) are misclassified. This means that malicious packets are rarely classified as normal packets, while normal packets have much higher chance to be classified into malicious packets and most of them were classified into reconnaissance attacks. For misclassified packets from DDoS/DoS and information theft attacks, packets from DDoS/DoS attacks are classified into reconnaissance attacks rather than information theft attacks while packets from information theft attacks are classified into reconnaissance attacks rather than DDoS/DoS attacks. From the above observation, we can see that packets from reconnaissance attacks are the main cause of the misclassification (i.e., either packets from reconnaissance attacks are classified into other attacks or packets from other attacks are classified into reconnaissance attacks). This could be due to the fact that behaviours of reconnaissance attacks reflected through the current feature set have similarity with behaviours of other attacks which makes it difficult to distinguish between reconnaissance attacks and other attacks.

Regarding the runtime, the FNN model completed the training, validation and testing procedure for each binary classification between 1 and 5 minutes and took roughly 20 minutes for the multi-class classification.

F. Comparison of Results

Finally, we applied the SVM model in [13] on the dataset in order to compare the prediction accuracy and runtime of our FNN model. It is important to note, however, that due to the long convergence time of the SVM model, we performed a random subsampling step on the processed dataset with sample sizes of 1000, 10000 and 30000 out of 7310546 instances.



(a) Accuracy

(b) Precision

(c) Recall

(d) F1 score

Fig. 3: Evaluation metrics for the binary classification

TABLE V: Confusion matrix for the multi-class classification

Actual/Predicted	Normal	DDoS/DoS	Reconnaissance	Information theft
Normal	200539 **(98.071%)**	0 (0.0%)	2910 (1.663%)	542 (0.266%)
DDoS/DoS	232 (0.026%)	882118 **(99.414%)**	2406 (0.560%)	0 (0.0%)
Reconnaissance	42 (0.009%)	5566 (1.400%)	356878 **(98.375%)**	1980 (0.217%)
Information theft	45 (0.169%)	0 (0.0%)	506 (10.914%)	8346 **(88.918%)**

We discuss the model settings and results for multi-class classification in detail in the following paragraphs.

As mentioned above, we randomly selected samples from the processed dataset so as to follow the same distribution (i.e., proportion) of data from different classes. We used a linear Support Vector Classifier (SVC) from the Scikit Learn library [1]. Due to the imbalanced nature of the different classes in the dataset, class weights were introduced in the training step. These weights were calculated following the FNN model (i.e., the weight for each class is given the inverse of the quotient value obtained by dividing the packet count in one class by the maximum packet count among all classes). A grid search was then applied to tune the SVC by cross validation of the penalty parameter values, including 0.1, 1 and 10. This cross validation was a 5-fold one applied on the 1000-sample set with a 80%-20% training and testing split. In line with our other experiments, the experiments were conducted on Google's Colaboratory. The overall runtime including training with each penalty parameter value and testing under the best estimator took roughly 168 minutes to complete and the optimal penalty parameter value retrieved from the grid search was 0.1. The SVC was applied with the same hyperparameter to train and test the other two subsets containing 10000 and 30000 samples.

The average accuracy and runtime results for the three subsets are illustrated in Table VI. The class size represents the number of selected samples in each class (normal, DDoS/DoS, reconnaissance, information theft) while the runtime is the time for training and testing under the optimal penalty parameter value.

From the table, we can see that the SVC yields a slightly higher accuracy with larger sample size. That said, the runtime also increases dramatically. This is even more important since the typical timing for our approach is of a few minutes. Compared with the runtime of the FNN model for the multi-class classification, the SVC is more than an order of magnitude slower. Thus, we could conclude that it is more time efficient to run the FNN model for classification over large dataset.

[1]Scikit is widely accessible at https://scikit-learn.org

TABLE VI: Accuracy and runtime of SVC for the multi-class classification

Sample size	Class size	Accuracy	Runtime (mins)
1000	(117, 479, 200, 4)	0.79	9.06
10000	(1100, 4810, 2029, 61)	0.80	86.18
30000	(3382, 14519, 5975, 124)	0.82	346.52

V. Discussion and Future Work

By extracting the field information of individual packets as generic features and developing the FNN model, we were able to differentiate the normal and malicious traffic in high accuracy and achieving low FP and FN predictions. However, we encountered several issues that we are planning to resolve in our future work and the details are discussed in this section.

For the binary classification, we could see the classifier is confused about embedded packets that exist in the malicious traffic from DoS over UDP and service scan attacks (i.e., an UDP packet inside of the ICMP payload) and normal traffic. Besides, the classifier also has a relatively low precision for data exfiltration and keylogging attacks. The nature of these attacks should be carefully investigated in the future and more relevant fields could be extracted from individual packets and used as new generic features for better classification.

For the multi-class classification, we could see the model is confused about certain categories of attacks (reconnaissance and information theft). Besides, we used attack category for the multi-class classification at the current stage as the model is confused about subcategories of attacks under the same category (e.g., DDoS and DoS attacks). The reason could be that the field information for the individual packet could not capture the certain attack behaviour in a large scale (e.g., a flood of traffic generated in DDoS/DoS attacks). Therefore, We are planning to develop a classifier for differentiating each subcategory of attacks in the future work. We will include the timestamp information, introduce more fields from individual packets as generic features, process the features and develop a new deep neural networks model to incorporate these new features. Long short term memory networks could be the best option to model time series data.

The current intrusion detection works in the batch mode. We will implement the online mode for the real time detection. In addition, we use a list of well-known ports for encoding port columns. We will adopt the adaptive encoding technique for port columns. The adaptive encoding is to map ports into vectors which inherit the similarity information between original packets. The mapping function will be learnt by the data.

VI. Conclusion

Deep learning as an intelligent technique is a solution for the intrusion detection problem that exists for IoT networks. Through the proposal of an intelligent binary and multi-class classification scheme via a feed-forward neural networks model along with the extraction and preprocessing of the field information in individual packets as generic features, and by testing its efficacy for the newly published IoT dataset with realistic network traffic, we have been able to demonstrate the performance of the proposed scheme. The capability of the classifier in binary classification for DDoS/DoS and reconnaissance attacks was demonstrated by the result of close to 0.99 across all evaluation measures including accuracy, precision, recall and F1 score. In the multi-class classification, the detection accuracy of above 0.99 for DDoS/DoS attacks was reported by the classifier while the normal traffic classification also yielded an accuracy of 0.98. The results are significant and warrant further research in this domain of cybersecurity for IoT networks.

References

[1] A. Bera, "80 Mind-Blowing IoT Statisitics (Infographic)," https://safeatlast.co/blog/iot-statistics/, February 2019, accessed on 2019-07-11.

[2] M. Chernyshev, Z. A. Baig, O. Bello, and S. Zeadally, "Internet of Things (IoT): Research, Simulators, and Testbeds," *IEEE Internet of Things Journal*, vol. 5, pp. 1637–1647, 2018.

[3] "MQTT," http://mqtt.org/, April 2019, accessed on 2019-07-11.

[4] V. V. Kumari and P. R. K. Varma, "A semi-supervised intrusion detection system using active learning SVM and fuzzy c-means clustering," in *Proceedings of the 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2017, pp. 481–485.

[5] W. Alhakami, A. ALharbi, S. Bourouis, R. Alroobaea, and N. Bouguila, "Network Anomaly Intrusion Detection Using a Nonparametric Bayesian Approach and Feature Selection," *IEEE Access*, vol. 7, pp. 52 181–52 190, 2019.

[6] D. H. Hoang and H. D. Nguyen, "A PCA-based method for IoT network traffic anomaly detection," in *Proceedings of the 2018 20th International Conference on Advanced Communication Technology (ICACT)*, 2018, pp. 381–386.

[7] Y. Zhang, P. Li, and X. Wang, "Intrusion Detection for IoT Based on Improved Genetic Algorithm and Deep Belief Network," *IEEE Access*, vol. 7, pp. 31 711–31 722, 2019.

[8] S. Shin, K. Kobara, Chia-Chuan Chuang, and Weicheng Huang, "A security framework for MQTT," in *Proceedings of the 2016 IEEE Conference on Communications and Network Security (CNS)*, 2016, pp. 432–436.

[9] S. Andy, B. Rahardjo, and B. Hanindhito, "Attack scenarios and security analysis of MQTT communication protocol in IoT system," in *Proceedings of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017, pp. 1–6.

[10] S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, "Modelling and Evaluation of Malicious Attacks against the IoT MQTT Protocol," in *Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2017, pp. 748–755.

[11] F. Haddadi, S. Khanchi, M. Shetabi, and V. Derhami, "Intrusion Detection and Attack Classification Using Feed-Forward Neural Network," in *Proceedings of the 2010 Second International Conference on Computer and Network Technology*, 2010, pp. 262–266.

[12] J. McHugh, "Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations As Performed by Lincoln Laboratory," *ACM Transactions on Information and System Security*, vol. 3, no. 4, pp. 262–294, 2000.

[13] P. A. R. Kumar and S. Selvakumar, "Distributed denial of service attack detection using an ensemble of neural classifier," *Computer Communications*, vol. 34, no. 11, pp. 1328–1341, 2011.

[14] N. G. B. Amma and S. Selvakumar, "Deep Radial Intelligence with Cumulative Incarnation approach for detecting Denial of Service attacks," *Neurocomputing*, vol. 340, pp. 294–308, 2019.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6.

[16] N. Moustafa and J. Slay, "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*, vol. 25, no. 1-3, pp. 18–31, 2016.

[17] N. Shone, T. N. Ngc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, pp. 41–50, 2018.

[18] 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," in *Proceedings of the 2017 International Conference on New Trends in Computing Sciences (ICTCS)*, 2017, pp. 167–172.

[19] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset," in *eprint arXiv:1811.00701*, 2018.