**Generative Adversary Network for Intrusion Detection System**

# Introduction:

Since the sudden growth of the interconnected digital world [1], such as the Internet of Things (IoT) and Software Defined Networks (SDNs), cyber-security assaults and their related risks have increased substantially. The IoT is a network of interconnected digital devices and systems known as 'things' [2]. Sensors, computing chips, and other technologies are integrated in them, allowing them to gather and share data through the internet. The goal of IoT networks is to enhance the productivity of the cloud platform, such as industrial systems and smart buildings. The number of IoT devices is predicted to touch fifty billion by the end of 2020 [2]. As a result of this expansion, the number of cyber-attack events and the related risk with them has increased. As a result, corporations and organizations are exploring for innovative ways to secure personal and corporate data held on network nodes. Unfortunately, current IoT system security mechanisms have revealed unreliable in the face of unprecedented threats [3]. In 2017, for example, attackers used an IoT fish tank thermometer to infiltrate a casino's sensitive information. In 2018, more than 2.4 million new malicious types were developed, referring to the Symantec Internet Security Threat Report [4]. As a result, there has been a surge in interest in enhancing the ability of NIDSs to identify novel attacks. As a result, new novel ways are necessary to increase the efficiency of Network Intrusion Detection Systems in detecting attacks (NIDSs).

In a network, an NIDS is used to monitor traffic flows in order to detect potential threats and secure digital assets [5]. It aims to preserve the three security principles of information systems, namely confidentiality, availability, and integrity [5]. It is developed to give strong cyber security protection in operating infrastructures. For a long time, the major purpose of NIDSs has been to detect cyber-attacks and threats. NIDSs are divided into two categories: The goal of signature-based attacks is to match and compare signatures from incoming communications to a database of predefined signatures from previously known assaults [6]. They normally give good detection accuracy for previously detected assaults, but they fail to identify latest or modified threats that are not in the database. NIDSs must be adaptable to new detection tactics, as attackers constantly vary their concepts and methods for executing attacks in order to avoid current security measures. The existing mechanism for tweaking signatures to keep up with changing attack vectors is unstable. Anomaly based NIDSs strive to overcome the drawbacks of signature NIDSs by employing advanced statistical approaches that have allowed researchers to discover behavioural trends of network traffic. Intrusion detection is accomplished using a variety of ways, including statistical knowledge and Machine Learning based algorithms [6]. They can achieve significant accuracy and Detection Rate (DR) for zero-day attacks because they match attack behaviours rather than signatures [7]. Anomaly NIDSs, on the other hand, have high False Alarm Rates (FARs) because they can label any innocuous traffic that differs from secure behaviour as an anomaly.

Existing signature NIDSs have shown to be ineffective at identifying zero-day attack signatures as they move across IoT networks [8]. This is due to the system's registry lacking known attack signatures. Many strategies, including machine learning, have been developed and implemented with some effectiveness to prevent similar occurrences from reoccurring. Machine Learning is a modern technology that has the ability to learn and extract hazardous patterns from network data, which can aid in the detection of security problems [9]. Deep Learning is a new field of machine learning that has shown to be particularly effective in detecting complex data patterns [10]. Its algorithms are based on biological brain systems, which convey data signals through a network of connected layers. A computational activation function in each unit translates input to output. Hidden layers in all of these algorithms can extract even more complicated patterns in network activity. Network attack vectors, which can be derived from numerous features communicated by network traffic, such as packet services, protocols, count/size, and signals, are used to understand these patterns. Each attack type has a distinct identification pattern, which is defined as a series of actions that, if left undetected, can weaken network security standards.

Researchers have created and tested a variety of machine learning (ML) models, which are frequently paired with Feature Reduction (FR) techniques to increase their efficiency. Although encouraging results for ML's detecting skills have been achieved using a set of assessment parameters, these models are not yet reliable for real-world IoT networks. Rather than acquiring insights into an ML-based NIDS application, the trend in this discipline has been to outperform state-of-the-art outcomes for a given dataset [11]. As a result, the vast volume of academic research undertaken far outnumbers the number of real deployments in the real life. Although this could be attributed to the high cost of errors in this sector compared to others [11], it's also possible that these strategies are unreliable in practise. This is due to the fact that they are frequently evaluated using a single dataset including a list of features that may not be viable to collect or store in a live IoT network stream. Furthermore, because of the architecture of machine learning, there is frequently room to improve in its hyper-parameters when applied to a specific data. As a result, the goal of this work is to assess the generalisability of Feature Extraction (FE) techniques and Machine Learning (ML) model combinations on various NIDS datasets.

# Literature Survey

Usually, three main methods are popular for network security. In the first method, few triggers are setting up to detect the attack on network like the threshold value exceeded. This detection approach informs the administration upon the threshold level but does not prevent the network from attack. The second option is to avoid an attack by putting in place defence policies that prevent the attack from happening, but this poses an issue if a valid operation is judged illegitimate, resulting in a denial service. The last method to block an attack by setting up the protection in the review of the attack and avoiding the attack when it happens in future. In the last two methods, the intrusion detection system (IDS) is setting up intrusion prevention system (IPS) [12]. The IDS can be configured on two locations base on the source of information. Firstly, the sensor can be fix on Host system (HIDS) or finally can be setting up on the network (NIDS) [13][14]. The Network Intrusion Detection Systems are mainly responsible to observer the network traffic by inspecting the different parameters like protocol usage, packet inspection, and the checking of IP addresses [13][14][15]. An IDS is a crucial tool for the security of the network and its success is mostly calculated by the accuracy of predicting the legitimate and illegitimate events.

Numerous ML and DL models had been showed significant accuracies but with their limitations and flaws. Due to the misclassification of attacks, ML models are usually vulnerable to them [16][17]. By inaccurately classify the data, ML model can enable the hacker to dodge the IDS that stand the organization on risk of undetectable attacks. Influential assaults, security violation attacks, and specificity attacks are the three main categories of offences [18][19]. In classical machine learning, adversarial assaults are more effective during the training phase than rather the validation phase. Attacking the model during the training phase will result in significant modifications to the classification result of the ML model. An attack at the validation or deployment phase, on the other hand, can only exploit the model's intrinsic flaws [20]. An attacker can influence either the dataset's features (feature noise) or its labels (label noise) [20]. Label noise, also known as label reversing or label flipping, is the practise of shifting the labels for the training of a model. Label flipping [21][22] appears to be a more successful approach of misleading a ML algorithm. A label flipping attack's threat model is to effect the model by delivering false labels during its training. If the labels are incorrect, the model cannot be fully and correctly trained [23]. SVMs are frequently more resistant to adversarial machine learning and an appropriate candidate for deployment in such situations [24]. If phisher have access to supply inaccurate labels to the ML model during training, they will be able to achieve their goal the most effectively [18].

Deep learning demonstrated to be effective at classifying data with correct labels. As a result, if labels are wrongly allocated, the model accuracy will be incorrect as well [23]. Because the input in deep learning models is separated into batches, traditional adversarial machine ML such as label reversing do not apply because labels would have to be reversed inside each batch [16]. The FGSM (Fast Gradient Sign Method) was proposed by the authors of [25] to produce new altered data. Because the Fast Gradient Sign Method is a linear data manipulation, it is fast to create fresh data [25]. To produce a new image, the FGSM seeks to make the DL model more expensive than the gradient of the loss on the given image. The Jacobian-based Saliency Map Attack was proposed in [26], and it is a family of assaults (JSMA). Although JSMA is more computationally intensive than FGSM, it can provide adversarial instances that are more similar to the original sample [26]. A study proposed in [27], improved the JSMA by adding targeted and non-targeted ways for generating adversarial samples. JSMA, on the other hand, consumes more computational resources than FGSM [28]. With their neural network model, ref. [29] reported comprehensive success in employing the FGSM to acquire on average 26% inaccurate classification per class. The study [29] used the NSL-KDD dataset and disturbed it with an epsilon value of 0.02. The studies in this research had a larger rate of wrong classifications, despite the fact that the epsilon value was at least 0.1. The authors of [18] used the UNSW-NB15 and NSL-KDD datasets to graphically evaluate the effects of applying PCA and Auto Encoder. They also used the classifiers K Nearest Neighbour (KNN), Deep Feed Forward, and Decision Trees in a binary and multi-class classification environment with a set of varied numbers of dimensions (ranging between 2 and 30). In the study, Auto Encoder outperformed PCA for KNN and Deep Feed Forward, but both performed similarly for DT. The UNSW-NB15 dataset has found an appropriate number of dimensions (20), whereas the NSL-KDD dataset did not. The Bot-IoT dataset and NN (neural networks) appear to be more resistant to adversarial FGSMs than the NSL-KDD dataset.

Nonetheless, an issue raised by ref. [30] is that the FGSM perturbs the entire dataset to which it is applied. Complete data perturbation has a two-fold effect. To begin, the features predominantly impacted by perturbing the Bot-IoT dataset are those found in network packet headers. Depending on the technique utilised, a seven-layer firewall could likely identify these types of assaults. Second, rather than using FGSM, study [30] recommends using the JSMA approach since it perturbs only a section of the dataset rather than the full dataset. The study [28] recommended that FGSM is better suited to experiments due to JSMA's computational power limitation. The employed FGSM against the NSL-KDD and discovered that perturbing the decreased the accuracy of their CNN from 99.05% to 51.35% [31]. Furthermore, while employing the JSMA approach on the NSL-KDD dataset, ref. [31] discovered that the accuracy dropped from 99.15% to 50.58%. Numerous machine learning model have been trained using big data to learn the pattern in data, allowing them predict whether or not a behaviour is benign [32]. The KDD-CUP-1999 was released in 1999 and commonly used in research field of Intrusion detection system using machine learning. Later, the KDD-CUP-1999 was updated and labelled as NSL-KDD dataset. From last two decades, these two datasets have been outdated for intrusion detection system (IDS) due to the absence of modern attack and new emerging protocols. Several studies [33]–[35] have declared these datasets to be biased and unrealistic.

Our work stands out from the others because it is the first to leverage the Bot-IoT dataset for adversarial instances generation [36]. In comparison to the NSL-KDD dataset, our findings show that the Bot-IoT dataset paired with NN creates a more robust configuration.

# Methodology

## Dataset Overview

There are many trained machine learning model available using dataset of the literature [37]. But the Bot-IoT dataset is the new dataset and in its early days. The Bot-IoT dataset was develop by attack and victim machines in stimulated environment [36]. In this environment, the network traffic is captured in dot pcap files and exporting in comma delimited (CSV) files after the proper processing and analysing. The Bot-IoT dataset was consist of more than 73 million instances with 43 feature values and 3 classification features. The dataset was categorized into 5 types of classes (Normal, Reconnaissance, DDoS, DoS, Information Theft) and extract the 5% sample from each class randomly [36]. The total counts for each category is presented in Table 1 [38].

Table 1: The count of each category in Bot-IoT dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Total Samples | 5% Samples | Training Samples | Testing Samples |
| DoS | 33,005,194 | 1,650,260 | 1,320,148 | 330,112 |
| DDoS | 38,532,480 | 1,926,624 | 1,541,315 | 385,309 |
| Reconnaissance | 1,821,639 | 91,082 | 72,919 | 18,163 |
| Theft | 1587 | 79 | 370 | 14 |
| Normal | 9543 | 477 | 370 | 107 |
| Total | 73,370,443 | 3,668,522 | 2,934,817 | 733,705 |

For the feature selection, the processing steps of the study of *Koroniotis et al.* [36] was followed. The joint entropy and correlation coefficient methods were implemented for feature selection from Bot-IoT dataset. The top 10 score features were extracted by using the score metrics of both algorithm Table 2. The selected features were highly depended on three classification features (attack, category, subcategory) of the dataset. The attack is the first classification feature proposed for binary classification. The label of attack feature was either true (1) or false (0) directly map with malicious and normal traffic respectively. All the classes of malicious traffic are labelled as True. The subsequent classification feature of Bot-IoT dataset is five class (DoS, DDoS, Normal, Information Theft, Reconnaissance) multi classification feature labelled as category. The category classification feature made up five string value (class) show in Table 1. The last classification attribute is subcategory that is the most described form of attack. The subcategory is made up of ten classification value by dividing each category into subcategory. Like the DoS category is further divided on the basis of protocol (TCP, UDP, HTTP), reconnaissance is categorized into OS finger printing and service scanning, information theft is split into data theft and key logging.

Table 2: Top 10 selected attributes of Bot-IoT dataset with feature selection method.

|  |  |
| --- | --- |
| **Top-10 Selected Features** | **Description** |
| seq | sequence number |
| stddev | Standard deviation of records |
| max | Average duration of aggregated records |
| srate | Source-to-destination packets per second |
| state\_number | Numeric representation of transaction state |
| mean | Average duration of records |
| min | Minimum duration of records |
| N\_IN\_Conn\_P\_DstIP | Number of inbound connections against destination IP |
| drate | Destination to source packets per second |
| N\_IN\_Conn\_P\_SrcIP | Number of inbound connections per source IP. |

## Machine Learning (ML) Overview

ML is an area of AI that enables machines to learn and progress on their own without having to be statically programmed. ML is concerned with the formation of automated models that can retrieve data and understand on their own. The training process begin with observations or data, such as examples, direct experience, or instruction, so that we can find the patterns in data and mark well forecast in the future based on the learning examples. The ultimate goal is for computers to learn on their own, without the need for human participation, and to change their behaviour accordingly. The first stage of this study is to replicate the implementation of random forest model proposed by the study [36]. The purpose of replication is to enable the fair comparison between machine ML model and DL model using evaluation measures. The structure and activation function for normal and adversarial data was same for fair evaluation. The second and third section focused on the generation of adversarial data for random forest and ANN model. Deep learning is an area of machine learning that deals with artificial neural networks, which are algorithms inspired by biological nervous system and function of the human brain. AI algorithms are usually divided in the following categories.

**Supervised ML Models**: Supervised ML models can use labelled samples to apply what they've observed in the historical to subsequent data and make predictions. The learning model generates an inferred function to make predictions about the target based on the examination of a known training examples. After sufficient training, the system can predict targets for any new input. The learning algorithm can also match its output to the correct, intended output and detect faults, allowing the model to be improved as needed.

**Unsupervised ML Models:** Unsupervised ML techniques, on the other hand, are utilised when the data being trained is not categorized/labelled. Unsupervised learning inspects how computers might infer a function from unlabelled data to explore an underlying pattern. The system doesn't figure out the appropriate result, but it investigates the data and can infer hidden patterns from unlabelled data using datasets.

**Semi-supervised ML Models:** Because they use both labelled and unlabelled data for learning – often a small amount of classified data and a big amount of unclassified data. Semi supervised ML algorithms fall in between supervised and unsupervised learning. This approach can significantly enhance learning efficiency in systems that adopt it. Semi-supervised learning is normally used when the obtained labelled data necessitates the use of expert and suitable resources to train / learn from it. Obtaining unlabelled data, on the other hand, usually does not necessitate additional resources.

**Reinforcement ML Models:** Reinforcement ML techniques are a category of machine learning approach that interacts with its environments by generating actions and identifying failures or rewards. The most essential elements of reinforcement learning are test and failure search and late compensation. This technology enables robots and software agents to autonomously choose the best behaviour in a given situation in order to increase their efficiency. For an agent to learn which achievement is better, simple reward response is required; this is describing as the reinforcement signal.

As the proposed study is based on BoT-IoT dataset in which all the samples are classified. So, our problem is fall in supervised machine learning category. The detail and methodology of different ML and DL algorithm for the execution of proposed problem is following.

**Random Forest Model**

Random forest (RF) is a convenient and flexible ML technique that, in most situations, provides tremendous results even without hyper-parameter optimization. Because of its flexibility, simplicity and adaptability, it is also one of the most extensively used machine learning algorithm (it can be used equally for regression and classification problems). RF is a supervised ML model and build ‘forest’ that is the ensemble of decision trees usually trained on ‘bagging’ method. In the first stage the random forest model was train on attack and category classification feature for binary and multi class classification respectively. The default parameter of random forest in scikit-learn package were used as hyper parameter except the n-estimator=20 and random-state=0. The cross validation technique was used to train the model rather than the train test split approach. The four cross validation were used that split the 0.75% data for training and 0.25% data for testing in each fold. The cross-fold validation technique iterated the 4th, 3rd, 2nd, and 1st, quarter as testing set in each fold respectively. Confusion matrix and evaluation measures ware also generated by the combination of training and testing data as proposed in [36]. The accuracy, f1-score, precision and recall were also calculated by the confusion matrix. For the multiclass classification RF was also trained with category feature as the label. The cross fold technique also used rather than train test split approach. As the category feature was based on the five string value (classes), so before the training of the model, the feature was encoded with OneHotEncoder function of scikit-learn. The hyper parameters were tuned with same values of random forest classification model. The evaluation of the model was done by calculating evolution measures using confusion matrix. The artificial neural network (ANN) model of deep learning model were also train for binary and multiclass classification on attack and category feature respectively.

**ANN Model**

Artificial neural networks (ANN) are used to simulate complex systems and predict target value for associated with the input parameters based on training experiences. ANN basically base on the biological nervous system of brain but it uses the reduce form of biological neural network. ANN specially simulate the electrical activity of nervous system and brain. An ANN is composing up of a massive number of cores that work in parallel and are stacked in tiers (layer). The raw input samples are obtained by the first layer, which is corresponding to the optic neurons in human visual perception. In the similar way as neurons further move from the optic nerve receive signals from those nearer to it, each subsequent layer takes the information from the layer before it, rather of the raw input. The system's output is produced by the last layer.

The ANN model had a five layer including one input layer and one output layer. The input layer was map to the total sum of features and had the total ten nodes. The three hidden layers between input and output layer are composed of 20, 60, 80 and 90 nodes respectively[38]. All the hidden layers were fully connected convolutional layers. The input layer was directly map with length of input features. The nodes on output layer was either two or five for binary and multi class classification. The activation function of the hidden layers are ‘tanh’ and Sigmoid on output layer for binary classification as sigmoid base models are more robust [25]. Softmax is the activation function used on output layer for multi-class classification and the architecture of both model is shown in Fig 1a and 1b. Both models were evaluated on the basis of accuracy and loss function graph.

Table

Description automatically generated

Figure 1:Structure of ANN model for binary classification.

Table

Description automatically generated

Figure 2: Structure of ANN model for multi-class classification.

**GAN Model**

Generative Adversarial Network, are a type of modelling in which samples are generated by employing the DL techniques such as CNN (convolutional neural network) and ANN (artificial neural networks. In ML, GAN modelling is an unsupervised learning job that entails automatically detecting and learning constancies or patterns in given data so that the system may be used to produce or output training examples that could have been taken from the given data. GANs are a smart way of training a generative model by describing the problem as a supervised learning problem with 2 different models: the generator model, which we prepare to generate new instances, and the discriminator model, which tries to categorise examples as real (from the domain) or fake (not from the domain) (generated). Both models are trained in an adversarial zero-sum field awaiting the discriminator model is tricked around half of the time, indicating that the generator model is providing credible examples.

The adversarial generative model (GAN) was used to generate adversarial data by adding noise in malicious real data. GAN model was train using generator and discriminator functions to generate fake malicious data. The above described CNN structure is also used in generator and discriminator functions of GAN. The generator model generates the new instances for malicious attack and the generator function predict them as these are malicious instances or benign instances. The combination of training and testing data was used in GAN and then 750 fake malicious samples were generated for binary classification. The purpose of generated samples is to test the robustness of RF and ANN binary classification models. The generated data was passed to both model as testing set and evaluate the performance of both by seeing how much generated malicious samples deceive the model.

## Evaluation Measures

The fourth and last stage of this study is to evaluate the results of trained model using by comparing real data with generated samples. Numerous evaluation measures including precision, recall, F1-score and accuracy were used (Eq. 1-4) to evaluate the both RF and ANN model. These evaluation measures were also calculated by generating graphical representation of confusion metrics rather than numeric values. The recall score is considered as key evaluation measure. As the decrease showed that the model increase false negative and increase in false negative allow hacker to launch cyber security attack which the trained model would not detect. For the ANN binary and multiclass classification accuracy and loss function graphs were also plotted to evaluate both models. For the generated sample same evaluation criteria was used for fair comparison.

# Results and Discussion

The Bot-IoT dataset was downloaded and prepared for ML and DL models. The Bot-IoT dataset total have the 19 features that can be used for statistical analysis. The features saddr, daddr, proto, dport, and sport are used to uniquely identify the data points and therefor removed from the training set for model deployment. The pkSeqID feature is a sample identity key that also removed from training set. Then the feature selection had been done to reduce the number of feature to avoid overfitting and get significant results. The aim of the feature selection methods to find those feature that play significant role for the training of models. Different types of statistical feature are implemented on the features to rank the features according to their significance. Lastly the top significant features are selected as training data. Correlation and joint Entropy feature selection algorithm were used and extract top ten features of Bot-IoT dataset for training (Table 2). As the dataset is too much large and imbalance in term of classes, the selection of whole dataset for training is not suitable. For the manageable training of the models, the 5% samples of the dataset were extracted resulting to 3.6 million samples [36]. The extracted 5% sample were split into training and testing set with the ratio of 80 and 20% respectively. After splitting the dataset, the training set have the 2934817 instances and testing set have the 733705 instances.

Secondly the training features data and attack classification feature was already in numeric format and did not required the label encoding. However, the category classification feature used five classes in string format that required the label encoding. The five classes of category feature were encoded using oneHotEncoder function of scikit-learn. The oneHotEncoder function of scikit-learn replace each class category feature with one corresponding numeric value. The DoS, DDoS, Normal, Reconnaissance, Information Theft are replaced by the 0 to 4 numeric values respectively. By replicating the Scaling, the data had normalized without training and testing split to equally scale the data. The MinMaxScaling Function of scikit learn was used to normalize data between the range of -1 and 1 according to Eq. 5 and 6. After preparing the dataset, the machine learning and deep learning models were train on trusted (normal) data and then test on trusted and manipulated data. Lastly the result of trusted data compare with manipulated data using evaluation measures.

The random forest is a ML model that was train on trusted data for binary classification. The hyper parameter of random forest was same as the default parameter in scikit-learn module except the n-estimator=20 and random-state=0. For the Random Forest the four cross validation technique was used rather than the train test split approach that describe in methodology section. In four cross validations the data is split 75% for training and 25% in testing for each iteration. The evaluation measures were also calculated with cross fold validation. Random forest showed the 99% accuracy for trusted data and the graphical confusion metrics with cross validation results is show in Fig 3. By viewing the confusion metrics, the model inaccurately predicts the six malicious attack as benign. The evaluation measures in Table 3 showed that model is strongly capable of detecting the malicious traffic over normal traffic. The Appendix A show the detail implementation of random forest model.

The random forest model was also train on category feature column for multi class classification. The hyper parameters were same as for binary classification. Random forest used the top ten features in Table 2 as input and the category feature as output. The label encoding scheme was used on category feature to encode the binary value into numeric values. The oneHotEncoder encodes the DOS, DDos, Normal, Information Theft and Reconnaissance into 0, 1, 2, 3, and 4 respectively. After the training of the model, RF multi-class classification model showed the 0.99% accuracy for trusted data. The confusion matrix was plotted (Fig 4) that showed that the model inaccurately predicted the 9 malicious sample as benign instances. All the defined evolution measures for this study were calculated using confusion matrix. The evaluation measures in Table 3 show the robustness of the model and how much it is capable to determine the summary of attacks over the network. The Appendix B show the detail implementation of random forest for multi class classification.

Chart

Description automatically generated

Figure 3: Confusion Metrics of random forest binary classification on trusted data.

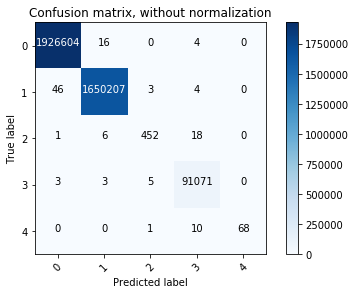


Figure 4:Confusion Metrics of random forest multi-class classification on trusted data.

The ANN model was also train for binary and multi-classification on attack and category feature respectively. Both models were made with five layers including input and output layer. The train test split on data was used for the training of both ANN models. For Binary classification, the sigmoid activation function was used with the Adam optimizer. The ‘sparse\_categorical\_crossentropy’ loss function and accuracy evaluation matrix was used with 20 epochs. The ANN showed 99% accuracy for binary classification. The loss score for the ANN binary classification model was very low. The accuracy and loss of the model is presented in Fig 5a and Fig 5c respectively. The graphical confusion matrix showed that model false negative rate is approximately zero. Hence, the model if fully capable to detect the malicious traffic over the normal traffic in real environment. For the multi class classification, the Softmax was used as an activation function on output layer with Adam optimizer. The rest of the parameters were same set for binary classification model. The multi-class classification models showed the 97% accuracy with near to the ground loss score. The accuracy and loss score of multi-class classification model is presented in Fig 5b and 5d respectively. Further, the multi class confusion metrics shows that the model inaccurately predicts the 21 attack samples as benign. The evaluation measures were also calculated using the confusion matrix of both model and showed in Table 3. The Appendix C and D shows the detail implementation of ANN model for binary and multiple class classification respectively.

|  |  |  |
| --- | --- | --- |
| C:\Users\BRL\Downloads\CNN-ACC.png  (a) | | C:\Users\BRL\Downloads\CNN-LOSS.png  (b) |
| C:\Users\BRL\Downloads\CNN5-ACC.png  (c) | | C:\Users\BRL\Downloads\CNN5-LOSS.png  (d) |
| (e) | C:\Users\BRL\Downloads\CM5-CNN.png (f) | |
| Figure 5: a and c display the training and validation accuracy of ANN model for binary and multiple class classification respectively. B and D shows the loss of both model. E and f show the confusion metric of CNN for binary classification. | | |

Lastly, the generative adversarial Network (GAN) was trained on combine dataset for the generation of manipulated data for binary classification. The GAN network was compose with generator function followed by the discriminator function. The purpose of generator function is to generate the samples and the discriminator aim to categorize the real and fake samples. The 100 iteration were performed on the combination of training and testing data by filtering attack records only. It firstly induced the noise in trusted samples and then start training on them. The generator generates the sample after inducing noise in each iteration and discriminator categorize them in normal and malicious classes. After the 100 iteration, the generator model was train enough to deceive the discriminator for malicious traffic. Then 750 fake sample were generated by the trained GAN model. These generated samples were passed through the pre-processing pipeline line that describe in methods section. After that these manipulated samples were predicted by the already trained random forest classifier and ANN model for binary classification to test the robustness of model. Random Forest already trained model was loaded and predicted with the 750 generated samples. RF showed the 99% accuracy for the malicious samples generated by GAN model. The model is again evaluated with predefined evaluation measures. The confusion matrix of RF for generated samples was plotted and all the evaluation measures were calculated by the graphical confusion matrix. The Confusion matrix of RF (Fig 6) showed that the model inaccurately classified the 42 generated malicious samples as benign. The FN rate was bearable for the manipulated samples to deploy it in real environment. The trained ANN binary classification model on trusted data was also tested on generated data. The ANN make the predictions for malicious generated samples and showed the 95% accuracy. The confusion matrix of ANN binary classification data showed that model inaccurately predict the 197 generated malicious samples and benign samples (Fig 7). The rest of the evaluation measures were also calculated the confusion matrix of ANN binary classification model. The FN rate for manipulated data showed the robustness of trained model.

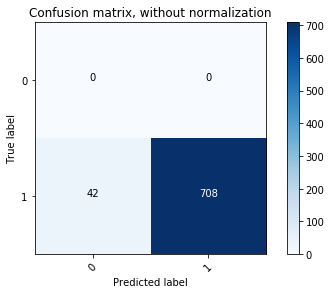


Figure 6: Confusion Metrics for RF on generated data.

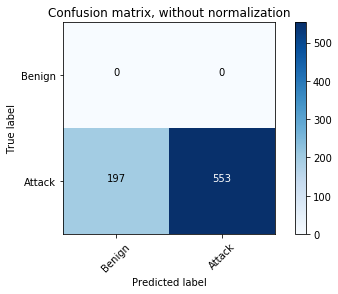


Figure 7: Confusion Metrics of CNN for generated data.

Now both binary classification, models were trained on trusted data and tested on trusted and fake generated data. The training accuracy of both models was more than 90% with low false negative rate (FNR) with significant values of other evaluation measures. For the 750 generated samples, RF and ANN model also show the robustness by classify the 708 and 553 accurately. The multiclass classification model was also train trusted data and it show feasible result on testing data (table 3). By viewing the evaluation measures of binary classification model, the proposed model can be deployed in real world network to detect malicious traffic over benign. As the hacker manipulate the data by different means to pass from security check and come in the network, the proposed binary classification model will detect the malicious traffic as it is tested on manipulated data. For future studies, there is need to a multi-class trainable model that will also capable to classify the attack traffic after the manipulation of data. The Appendix D and E showed the implementation of making prediction from RF and ANN model for binary classification respectively.

Table 3: Calculated avalutation measures for all trained models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Data Type | Classification Type | Accuracy | Precision | Recall | F1 Score |
| RF | Trusted | Binary | 0.99% | 0.99% | 0.99% | 0.99% |
| RF | Generated | Binary | 0.94% | 0.94% | 1.0% | 0.96% |
| RF | Trusted | Multi Class | 0.99% | - | - | - |
| CNN | Trusted | Binary | 0.99% | 0.99% | 1.0% | 0.99% |
| CNN | Generated | Binary | 0.73% | 0.73% | 1.0% | 0.84% |
| CNN | Trusted | Multi Class | 0.97% | - | - | - |

# Conclusion

The numerous studies have been published based on ML models for the recognition of attack traffic on the network. But these studies faced the similar problem that these were failed in detection when label flipped data or manipulated data passed to them. The basis of the failure of these studies was mainly the dataset and robustness of trained models. In this study, the Bot-IoT dataset was used to train the models. Bot-IoT is collected in real environment and much flexible to detect the malicious traffic over the network and to our best knowledge it is used in only one study [38]. Secondly, the architecture of models was also robust to classify the malicious samples over the benign traffic. By using the Bot-IoT dataset, RF and ANN models were train for binary and multiple class classification. By the train test split and cross validation approach, all models showed remarkable accuracies with other evaluation measures. GAN models generate the manipulated data after training on the attack sample of trusted data. RF and ANN also classify the manipulated data generated by the GAN model. After testing the RF and ANN on manipulated data for binary classification, it is concluded that the both models are robust enough to deploy in real world environment and also capable to classify the manipulated data after deployment.

**Appendix**

|  |
| --- |
| **A. Random Forest for binary Classification.** |
| def RandomForest(X=input\_attributes, y=output\_attribute):  RF-model = RandomForestClassifier(random-state=0, n-estimators=20)  predictions = cross-val-predict(RF-model, X, y, cv=4)  confusion-matrix-summary = confusion-matrix(y, predictions) |
| **Appendix B. Random Forest for multi-class Classification.** |
| def RandomForest(X=input\_attributes, y=output\_attribute):  RF-model = RandomForestClassifier(random-state=0, n-estimators=20)  predictions = cross-val-predict(RF-model, X, y, cv=4)  confusion-matrix-summary = confusion-matrix(y, predictions) |
| **Appendix C. ANN for binary Classification.** |
| def ANN\_Model(X\_train, X\_test, y\_train, y\_test):  model1 = Sequential()  model1.add(Dense(20, input\_dim=10, activation='tanh'))  model1.add(Dense(60, input\_dim=20, activation='tanh'))  model1.add(Dense(80, input\_dim=60, activation='tanh'))  model1.add(Dense(90, input\_dim=80, activation='tanh'))  model1.add(Dense(5, input\_dim=90, activation='sigmoid'))  model1.compile(loss='sparse\_categorical\_crossentropy',optimizer='adam', metrics=['accuracy'])  model1.fit(X\_train, y\_train, validation\_data=(X\_test,y\_test), epochs=10, batch\_size=256) |
| **Appendix D. ANN for multi-class Classification.** |
| def ANN\_Model(X\_train, X\_test, y\_train, y\_test):  model2 = Sequential()  model2.add(Dense(20, input\_dim=10, activation='tanh'))  model2.add(Dense(60, input\_dim=20, activation='tanh'))  model2.add(Dense(80, input\_dim=60, activation='tanh'))  model2.add(Dense(90, input\_dim=80, activation='tanh'))  model2.add(Dense(5, input\_dim=90, activation='softmax'))  model2.compile(loss='sparse\_categorical\_crossentropy',optimizer='adam', metrics=['accuracy'])  model2.fit(X\_train, y\_train, validation\_data=(X\_test,y\_test), epochs=10, batch\_size=256) |
| **Appendix E. Testing of RF model for generated data.** |
| def RF\_test(generatedData= X):  modelName = 'trained\_RF\_model.sav'  RF\_trained\_model = pickle.load(open(modelName, 'rb'))  predictions = RF\_trained\_model.predict(generatedData) |
| **Appendix E. Testing of ANN model for generated data.** |
| def ANN\_test():  ANN\_trained\_model = load\_model('trained\_dense\_model.h5')  ANN\_trained\_model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])  model\_score = ANN\_trained\_model.evaluate(X=input\_features, y=output\_feature, verbose=0)  print("%s: %.2f%%" % (ANN\_trained\_model.metrics\_names[1], model\_score[1]\*100)) |

**References**

[1] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, “A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3453–3495, Oct. 2018.

[2] M. A. Khan and K. Salah, “IoT security: Review, blockchain solutions, and open challenges,” *Futur. Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[3] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, “Internet of Things (IoT): Taxonomy of security attacks,” *2016 3rd Int. Conf. Electron. Des. ICED 2016*, pp. 321–326, Jan. 2017.

[4] “ISTR Internet Security Threat Report Volume 24 |,” 2019.

[5] F. Y. Sattarova, “Integrating intrusion detection system and data mining,” *Proc. - 2008 Int. Symp. Ubiquitous Multimed. Comput. UMC 2008*, pp. 256–259, 2008.

[6] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Comput. Secur.*, vol. 28, no. 1–2, pp. 18–28, Feb. 2009.

[7] P. Amoli, T. Hamalainen, … G. D.-… (International J. of, and undefined 2016, “Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets,” *users.jyu.fi*.

[8] M. J. Hashemi, G. Cusack, and E. Keller, “Towards evaluation of NIDSs in adversarial setting,” *Big-DAMA 2019 - Proc. 3rd ACM Conex. Work. Big DAta, Mach. Learn. Artif. Intell. Data Commun. Networks, Part Conex. 2019*, pp. 14–21, Dec. 2019.

[9] C. Sinclair, L. Pierce, and S. Matzner, “An application of machine learning to network intrusion detection,” *Proc. - Annu. Comput. Secur. Appl. Conf. ACSAC*, vol. Part F133431, pp. 371–377, 1999.

[10] Q. Niyaz, W. Sun, A. Y. Javaid, and M. Alam, “A deep learning approach for network intrusion detection system,” *EAI Int. Conf. Bio-inspired Inf. Commun. Technol.*, 2015.

[11] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” *Proc. - IEEE Symp. Secur. Priv.*, pp. 305–316, 2010.

[12] O. Ibitoye, O. Shafiq, A. M.-2019 I. Global, and undefined 2019, “Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks,” *ieeexplore.ieee.org*.

[13] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, “From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3369–3388, Oct. 2018.

[14] N. T. Van, T. N. Thinh, and L. T. Sach, “An anomaly-based network intrusion detection system using Deep learning,” *Proc. - 2017 Int. Conf. Syst. Sci. Eng. ICSSE 2017*, pp. 210–214, Sep. 2017.

[15] E. Benkhelifa, T. Welsh, and W. Hamouda, “A critical review of practices and challenges in intrusion detection systems for IoT: Toward universal and resilient systems,” *IEEE Commun. Surv. Tutorials*, vol. 20, no. 4, pp. 3496–3509, Oct. 2018.

[16] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial Machine Learning at Scale,” *5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc.*, Nov. 2016.

[17] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, “A taxonomy and survey of attacks against machine learning,” *Comput. Sci. Rev.*, vol. 34, p. 100199, Nov. 2019.

[18] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, “Adversarial Machine Learning \*,” *Proc. 4th ACM Work. Secur. Artif. Intell. - AISec ’11*, 2011.

[19] H. Xiao, H. Xiao, and C. Eckert, “Adversarial Label Flips Attack on Support Vector Machines,” *Front. Artif. Intell. Appl.*, vol. 242, pp. 870–875, 2012.

[20] B. Biggio, B. Nelson, and P. Laskov, “Support Vector Machines Under Adversarial Label Noise.” PMLR, pp. 97–112, 17-Nov-2011.

[21] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli, “Support vector machines under adversarial label contamination,” *Neurocomputing*, vol. 160, pp. 53–62, Jul. 2015.

[22] P. W. Koh, J. Steinhardt, and P. Liang, “Stronger Data Poisoning Attacks Break Data Sanitization Defenses,” Nov. 2018.

[23] R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, A. Miri, and M. Conti, “On defending against label flipping attacks on malware detection systems,” *Neural Comput. Appl. 2020 3218*, vol. 32, no. 18, pp. 14781–14800, Jul. 2020.

[24] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, “Adversarial Support Vector Machine Learning,” *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD ’12*, vol. 12.

[25] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and Harnessing Adversarial Examples,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Dec. 2014.

[26] N. Papernot, P. Mcdaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, “The limitations of deep learning in adversarial settings,” *Proc. - 2016 IEEE Eur. Symp. Secur. Privacy, EURO S P 2016*, pp. 372–387, May 2016.

[27] R. Wiyatno and A. Xu, “Maximal Jacobian-based Saliency Map Attack,” Aug. 2018.

[28] K. Yang, J. Liu, C. Zhang, and Y. Fang, “Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems,” *Proc. - IEEE Mil. Commun. Conf. MILCOM*, vol. 2019-October, pp. 559–564, Jan. 2019.

[29] Z. Wang, “Deep Learning-Based Intrusion Detection with Adversaries,” *IEEE Access*, vol. 6, pp. 38367–38384, Jul. 2018.

[30] M. Rigaki, “Adversarial Deep Learning Against Intrusion Detection Classifiers,” 2017.

[31] J. Jeong, S. Kwon, M.-P. Hong, J. Kwak, and T. Shon, “Adversarial attack-based security vulnerability verification using deep learning library for multimedia video surveillance,” *Multimed. Tools Appl. 2019 7923*, vol. 79, no. 23, pp. 16077–16091, Jan. 2019.

[32] S. Sapre, P. Ahmadi, and K. Islam, “A Robust Comparison of the KDDCup99 and NSL-KDD IoT Network Intrusion Detection Datasets Through Various Machine Learning Algorithms,” Dec. 2019.

[33] McHughJohn, “Testing Intrusion detection systems,” *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[34] M. V. Mahoney and P. K. Chan, “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2820, pp. 220–237, 2003.

[35] N. Athanasiades, R. Abler, J. Levine, H. Owen, and G. Riley, “Intrusion detection testing and benchmarking methodologies,” *Proc. - 1st IEEE Int. Work. Inf. Assur. IWIA 2003*, pp. 63–72, 2003.

[36] 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,” *Futur. Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[37] Y. Pacheco and W. Sun, “Adversarial Machine Learning: A Comparative Study on Contemporary Intrusion Detection Datasets.”

[38] P. Papadopoulos, O. T. von Essen, N. Pitropakis, C. Chrysoulas, A. Mylonas, and W. J. Buchanan, “Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT,” *J. Cybersecurity Priv. 2021, Vol. 1, Pages 252-273*, vol. 1, no. 2, pp. 252–273, Apr. 2021.