National College of Ireland

Detection of FTP and SSH Bruteforce Attacks using Deep Belief Network Model

MSc Research Project
Cybersecurity

Loveth Ukamaka Diwe
Student ID: x21180211

School of Computing
National College of Ireland

Supervisor: Imran Khan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	LOVETH UKAMAKA DIWE
Student ID:	X21180211
Programme:	MSc Cybersecurity **Year:** 2023
Module:	Research Project
Supervisor:	Imran Khan
Submission Due Date:	18/09/2023
Project Title:	Detection of FTP and SSH Bruteforce attacks using Deep Belief Network Model
Word Count:	7720 **Page Count** 26

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: LOVETH UKAMAKA DIWE

Date: 15/09/2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Detection of FTP and SSH Bruteforce Attacks using Deep Belief Network Model

Loveth Ukamaka Diwe
x21180211

Abstract

Cyber-attacks on computers and internet have been on a major increase since every organization has to communicate and run business operations on the network. The security awareness and protections implored by individuals and organizations daily is still not enough to fully eradicate the constant attacks. Bruteforce attack is a major high-level attack on the network connected via protocols like FTP, SSH or via the WEB. Brute Force attacks use methods like credential stuffing, DNS spoofing, multiple trials, and different consistent methods to access the targeted system or network. This attack type can be unpredictable and is often a gateway to many other attacks once successful hence, the need for a robust detection method arises. In this research, I investigated the application of Deep Belief Network (DBN) to detect Bruteforce attacks using the CSE-CICIDS2018 dataset. For additional performance comparison, we also trained three other algorithms: Decision Tree, Random Forest, and Logistic Regression. Experiment results showed that DBN model achieved a higher accuracy score of 99.3% while Random Forest had accuracy of 98.2%, Decision Tree 85.8% and Logistic Regression 99.2%.

1 Introduction

With the increase in the dependence of individuals and organizations on technology and digital infrastructures for day to activities, there is also an increase in the security threats and risks with these improvements. Cyber-attacks and Network intrusions continue to increase posing a constant threat to the required security needed for communications and data management. Hence, there is a paramount need to provide effective intrusion detection systems to be used in real time.

Bruteforce network attacks are attacks that can compromise the system security and infiltrate vulnerable machines. Bruteforce attacks involve repeated attempts to guess login credentials by systematically trying all combinations until the correct one is found (Roger, 2020). These attacks target a wide range of services and protocols, such as File Transfer Protocol (FTP), Cross-Site Scripting (XSS), Secure Shell (SSH), and web applications, thereby posing a substantial risk to network security.

Traditional approaches to intrusion detection frequently rely on approaches based on signatures that correspond to known attack patterns (Bo & Xue, 2016). These techniques, however, have trouble spotting new or developing attacks for which there are not any signatures yet coupled with it taking longer processing time for large data. Whereas the Intrusion Detection Systems (IDS) can adapt to new attack patterns thanks to machine learning techniques, which have

emerged as a solution to this restriction and offers a statistical and mathematical algorithm needed for effective detection of attacks (Ahmad, et al., 2020).

Recent studies on the use of deep learning algorithms have successfully proven to be a successful method to detect network intrusion attacks, speech recognition and language processing. There have been several studies focusing on CNN (Convolutional Neural Network) algorithm for detecting network intrusion attacks, DDoS attacks and Bruteforce attacks using different datasets. From my checks on studies published, there has been no recent study on the performance of Deep Belief Network using CSE-CIC-IDS2018 dataset. This dataset is more recent and has more network features that capture patterns and relationships which can be effective in recognizing attack vectors. This study explores an intrusion detection algorithm for brute force attacks detection leveraging on Deep Belief Network (DBN) algorithm targeting SSH and FTP.

Listed below are the contributions of this research.

- Collect a full network traffic dataset (CSE-CIC-IDS2018) which comprised of Benign, FTP, and SSH attack.
- Conduct feature extraction and pre-process the dataset to achieve a more comprehensive data required for analysis.
- Train the DBN model to capture the complex pattern within the network data.
- Evaluate the result and compare it to other models.

Research Question. How well does Deep Belief Network model perform at detecting FTP-Bruteforce attacks and SSH-Bruteforce attacks?

The layout of the research is arranged as follows. Literature reviews are presented in section 2. A brief introduction of the DBN model and method of training is presented in section 3. Description of the dataset content and Evaluation metrics were also presented. In section 4, the design specification of the Model is presented. Section 5 presents the implementation of the work and all necessary details such as research tools and applications installed to bring the work to fruition. The Section 6 of the report details the evaluation and result of the model together with the comparisons of other algorithms existing. A general discussion of the work was stated here. Finally, the conclusion of the work and future work is reported accordingly in Section 7.

2 Related Work

This section presents an overview of some previous studies done on detection of brute force attacks and network intrusion attacks under the four categories of attack detection techniques.

2.1 Signature-based detection

Signature-based detection method for brute-force attacks involves developing and storing some guidelines and known signatures which recognize the patterns of brute force attacks and intrusion traits (Quincozes, et al., 2021). These known patterns and signatures are then looked out for during network scanning to identify attacks.

(Bronte, et al., 2016) developed a signature-based detection method for detection of application layer intrusion. This was done using a dataset which they generated from a web application with malicious scripts.

Another study by (Jing, et al., 2017) proposed the use of signature -based detection and introduced some other detection methods like dynamic detection and integrity detection which together can help for better catch. The implementation of the signature technique suggested in the study is easy but might not be effective for more advanced attacks and unknown threats with no pattern.

The limitation with the use of signature-based detection method is that new attack trends cannot be detected with it since only pre-stored patterns in the database can be detected. (Khraisat, et al., 2019). Hence, a need for an improved detection method is necessitated.

2.2 Anomaly-based detection

Anomaly-based detection technique involves checking for any disparity from the baseline of the established system's normal behaviour (Montoya, et al., 2022). It also involves observing activities which are related to brute force attacks in system logs, traffic, and network activities.

(Deris, et al., 2019) in their study explored brute force attacks against IoT network FTP servers adopting a time-sensitive and a thorough statistical relationship for detecting anomalies, unauthorized activities, and visualizing attack patterns in network configurations. This was conducted using an IoT network testbed which replicated the scenario for an internal attack with the objective of establishing and identifying the attack pattern which eventually resulted in a positive result.

A more improved study on brute force attack detection on IoT Network proposed using a deep learning method for accurate detection of the attack on IoT using MQTT-IoT-IDS2020 dataset (Ahmed, et al., 2023). To train the proposed model, Deeplearning4j java library was used to achieve the result of Bi-flow Accuracy of 99.56% and Uni-flow Accuracy of 99.67%.

Another study by (Liu, et al., 2021) proposed a feature engineering technique which could create a block-based packet payload to detect anomalies and help to spot brute force attacks.

(Jeonghoon, et al., 2021) in their study explained some of the limitations with regards to lack of a standard procedure required for log files analysis and detection of security of alerts in time. He proposed a model to detect SSH brute force by extracting some kinds of data such as IP address. The result showed that the proposed approach can prevent unauthorized IPs and prevent access to malicious sites owned by the attackers.

(Günter, 2022) created Condition Monitoring System (CMS) in his study to monitor in real time SSH-brute force attacks observed from anomalies, and fluctuations. This experiment was done using a generated dataset.

One of the core limitations of the anomaly-based detection method is that there is always a high chance of false alert triggers.

2.3 Machine Learning-based detection

In this approach, machine-learning models are used to recognize patterns of attacks in a system. They are used to detect anomalies in system logs, activities and network traffic which could indicate an intrusion or a brute force attack. With historical data, these algorithms may be

trained with standard libraries to recognize well-known attack patterns, and they can also be changed to recognize emerging patterns and attack traits (Alatwi & Morisset, 2022).

(Yin, et al., 2017) created an RNN model for intrusion detection using KDDTrain and KDDTest & KDDTest21 as the training set and the testing set, respectively. Result of the model trained shows that the RNN model has higher accuracy when the training set was used.

Another study (Kim, et al., 2019) developed a CNN model for intrusion detection. In this study, the dataset was converted into images to improve the performance of the study by (Yin, et al., 2017). The accuracy score increased after the dataset was pre-processed.

A study by (Karel, et al., 2020) explored ways to fix high FP rate which is experienced in some detection methods. The study proposed refined machine learning for detection with extended IP Flow extended specifically using a generated dataset. The result showed that the Ada-Boosted tree had the highest accuracy of 99.47% followed by C4.5 Decision tree: 99.46%, 5-NN: 99.39%, Random Forest: 99.38%, Naive Bayes: 92.09%.

Using the approach of detection via packet level feature set with a generated dataset. This dataset consists of 1.8 million number of IP flows, (Jan, et al., 2021) in their study was able to identify a distinctive packet-level feature set after training some classifiers and achieved a false positive rate of 10^-4. Also, the true positive rate increased resulting to 0.938. The true positive rate was higher than the state of art solution, thus can be used as a detection method.

(Hossain et al., 2020) also proposed using LSTM algorithm (Long Short-Term Memory) for detection of brute-force attack with focus on protocols SSH and FTP. The analysis was done with CICIDS2017 dataset. The model result when compared to other classifiers such as k-nearest-neighbour (KNN) and naive Bayes (NB) showed that the LSTM model delivers a superior accuracy of 99.88%.

Another study by (John, et al., 2021) proposed that a simpler method can be used for the detection of brute force attacks using the CIC IDS2018 dataset which is a more recent big dataset for network intrusion. The study trained and tested simple Decision Tree models with two independent variables to classify the dataset for detection of SSH and FTP brute force attacks using the Scikit-learn decision tree implementation (F. Pedregosa, 2011). The result was evaluated based on AUC and AUPRC and scored greater than 0.99 and with that they concluded using a simple decision tree algorithm can equally be used for the detection of the attack.

A study by (Wanjau, et al., 2021) proposed using a CNN model with CSE-CIC-IDS 2018 dataset for detecting SSH Brute force attacks and implemented with the use of Python's libraries like TensorFlow and Keras. The performance of the model when compared to the past results of some machine learning classifiers like Naive Bayes, Decision Tree and KNN showed that the model has a higher accuracy rate and higher precision rate when all features are used.

2.4 Hybrid detection

Hybrid detection technique is a combination of two or more detection techniques for effective brute-force attack detection. This can be a combination of old and new trends of attacks or a combination of signature-based and anomaly detection methods (Gupta & Srinivasagopalan, 2020).

(Fernandez & Xu, 2019) in their study proposed both supervised network intrusion and unsupervised network intrusion detection using deep learning method. The intrusion detection system was trained with a supervised Deep Neural Network (DNN). An auto encoder was also used to categorize and detect attack traffic with unsupervised learning. The results from the study showed the intrusion detection systems performed better than the traditional machine learning models.

A study by (Luo, et al., 2021) also developed a hybrid learning technique which consists of self-organizing map (SOM) to detect SSH Brute force attacks. The network flow clusters were recognized using the SOM. The Association Rule Mining (ARM) algorithm was deployed for analysing and interpreting traffic behaviour.

Another study by (Wu, et al., 2020) proposed a method for threat intelligence detection from SSH brute force attacks using a honeypot dataset for experimentation. The study distinguished between cross-country and persistent attacks and warned IoT vendors cloud providers about stolen SSH keys, allowing them to check the effectiveness of software fixes and discourage the evasion strategies with anomaly detectors. The implementation was human supervised, and the researchers were able to detect ways by which attackers launch persistent large-scale attacks.

(Saumya & Tamanna, 2021) proposed a combination signature-based detection algorithms and an unsupervised anomaly-based detection method to detect intrusion attacks with the aim to lower false alarm rates and improve the detection rate.

Table 1: Related Studies on Brute Force attack detection

Study/Year	Dataset	Method	Performance
(Bronte, et al., 2016)	Generated Dataset	Genetic Algorithm	False Positive: 0% False Negative: 0%
(Jing, et al., 2017)	NA	Signature-based	The method is simple and easy to implement.
(Jeonghoon, et al., 2021)	Generated internet router dataset	Network data log.	It can prevent unauthorized access, but it does not respond to real-time attacks.
(Yin, et al., 2017)	NSL-KDD dataset	RNN	Accuracy:81.29% (KDDTest) Accuracy: 64.67% (KDDTest-21)
(Kim, et al., 2019)	CIC-CSE-IDS2018	CNN	Improved accuracy results for all attack types.
(Hossain, et al., 2020)	CICIDS2017	LSTM	Accuracy of 99.88%
(Wanjau, et al., 2021)	CIC-IDS 2018	CNN	Accuracy = 94.3%, Precision = 92.5%, Recall = 97.8% F1-score = 91.8%
(Fernandez & Xu, 2019)	CIC IDS 2017, ISCX IDS 2012	DNN	TPR = 0.9999 TFR = 0.0003
(Jan, et al., 2021)	Generated dataset	HistGB AdaBoost Random Forest SVM Decision Tree KNN and GaussianNB	TPF = 0.938

<table>
<tr><td>(Wichmann, et al., 2021)</td><td>Real-world traffic from a Tor exit node and via synthetic.</td><td>Based on TCP packets.</td><td>FTPS: F-measure between 85% and 93%,
IMAPS: F-measure between 75% and 86%</td></tr>
<tr><td>(Ahmed, et al., 2023)</td><td>MQTT-IoT-IDS2020</td><td>DNN</td><td>Bi-flow Accuracy= 99.56%
Uni-flow Accuracy = 99.67%</td></tr>
<tr><td>(Deris, et al., 2019)</td><td>Generated dataset</td><td>Snort</td><td>Successfully detected Snort</td></tr>
<tr><td>(Karel, et al., 2020)</td><td>Generated dataset</td><td>Ada-Boosted tree,
5-NN,
Naive Bayes,
RandomForest
C4.5 Decision tree</td><td><table><tr><th>Algorithm</th><th>Accuracy</th></tr><tr><td>Ada-Boosted tree</td><td>99.47%</td></tr><tr><td>Naive Bayes</td><td>92.09%</td></tr><tr><td>5-NN</td><td>99.39%</td></tr><tr><td>C4.5 Decision tree</td><td>99.46%</td></tr><tr><td>Random forest</td><td>99.38%</td></tr></table></td></tr>
<tr><td>(Gunter, 2022)</td><td>Generated dataset</td><td>NA</td><td>Provided a successful monitoring system for SSH-brute force attacks happening in real time</td></tr>
<tr><td>(John, et al., 2021)</td><td>CSE-CIC-IDS2018</td><td>Decision Tree classifier</td><td>AUC & AUPRC scores greater than 0.99.</td></tr>
<tr><td>(Wu, et al., 2020)</td><td>Honeypot dataset</td><td>Human supervised</td><td>The researchers were able to detect ways by which attackers launch persistent large-scale attacks.</td></tr>
<tr><td>(Saumya & Tamanna, 2021)</td><td>CSE-CIS-1DS2018 Dataset</td><td>Decision tree
Logistic regression
Adaboost
Naive bayes Nearest neighbour.
+ Isolation tree</td><td>HYBRID IDS MODELS<table><tr><th>Hybrid model</th><th>Precision</th><th>Recall</th><th>F1-score</th></tr><tr><td>Decision tree + Isolation tree</td><td>0.92</td><td>0.78</td><td>0.84</td></tr><tr><td>Logistic regression + Isolation tree</td><td>0.72</td><td>0.75</td><td>0.73</td></tr><tr><td>Adaboost + Isolation tree</td><td>0.25</td><td>0.67</td><td>0.36</td></tr><tr><td>Naive bayes + Isolation tree</td><td>0.25</td><td>0.56</td><td>0.35</td></tr><tr><td>Nearest neighbour + Isolation tree</td><td>0.30</td><td>0.75</td><td>0.43</td></tr></table></td></tr>
</table>

Overall, more studies are being done for efficient and easier detection of brute force attacks with the major work being on machine learning approach. This is because it is currently the most successful approach of detection when applied to other attacks intrusions (Bo & Xue, 2016).

3 Research Methodology

This section presents methodology proposed for effective detection of FTP & SSH brute force attack. I also explained the data collection techniques and the evaluation metrics used. This research offers the use of deep belief network algorithm to detect Bruteforce attacks, case study being on FTP and SSH protocol using CIC IDS2018 dataset as a sample data for analysis (UNB, 2023). A comparative analysis with three classical traditional machine learning models to ascertain the performance rate with the DBN was also presented.

3.1 Deep Belief Networks

Deep Belief Network is a deep neural network classifier that combines multilayer unsupervised learning networks called RBMs (Restricted Boltzmann Machines) with a supervised learning network called BP (Backpropagation) (WEI, et al., 2019). They are artificial neural networks which are made up of layers of RBMs. These RBMs are trained with an unsupervised method

to learn the input data which is compressed (Tavanaei, et al., 2019). Each layer of the RBM learns a representation of the data which is much more abstract as it receives the input from the previous layer. This way, each RBM layer as a result, learns to extract higher-level features from the input data.

3.1.1 Restricted Boltzmann Machine (RBM)

RBMs are two-layered networks of stochastic units with undirected connections between pairs of units in the two layers (Guido, 2018). The two node layers are known as visible and hidden nodes with no connections from visible to visible or hidden to hidden nodes. RBMs are among the basic blocks of deep belief network and can be used as a generative model as well as a classification task. To learn an RBM, design of the training algorithm using the maximum likelihood estimation of the parameters.

Fig.1 shows RBM which consists of visible units (m) and hidden units (n). (b) and (c) represent the bias for the visible unit and the hidden unit. (wij) is the weight between (hi) and (vj).

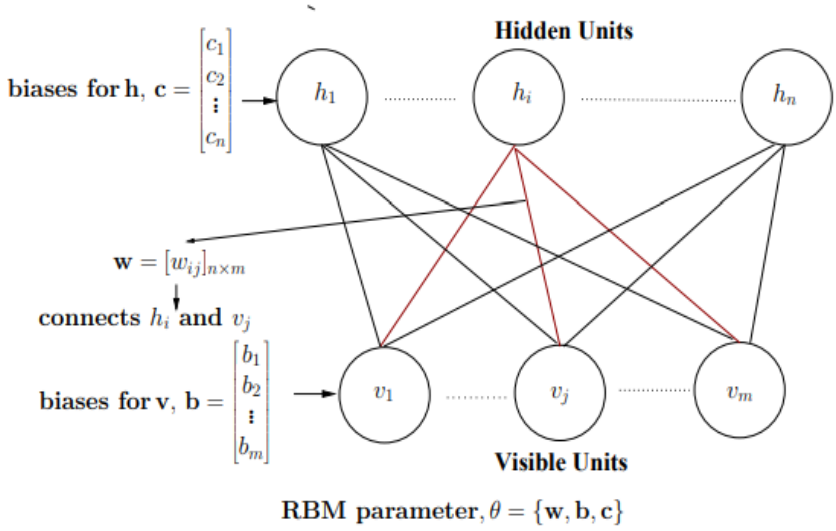

Fig. 1. Restricted Boltzmann Machine (Guido, 2018).

3.1.2 Architecture and Structure of DBNs

There are two primary layers that make up the DBN architecture. The layers are visible layers and hidden layers. These two layers form an undirected graph. The input data is directly represented by the visible layer, while the more abstract and higher-level data features are captured by the hidden layers.

In general, the DBN is made of multiple layers of networks and each pair of the connected layers are RBMs which form a stack around.

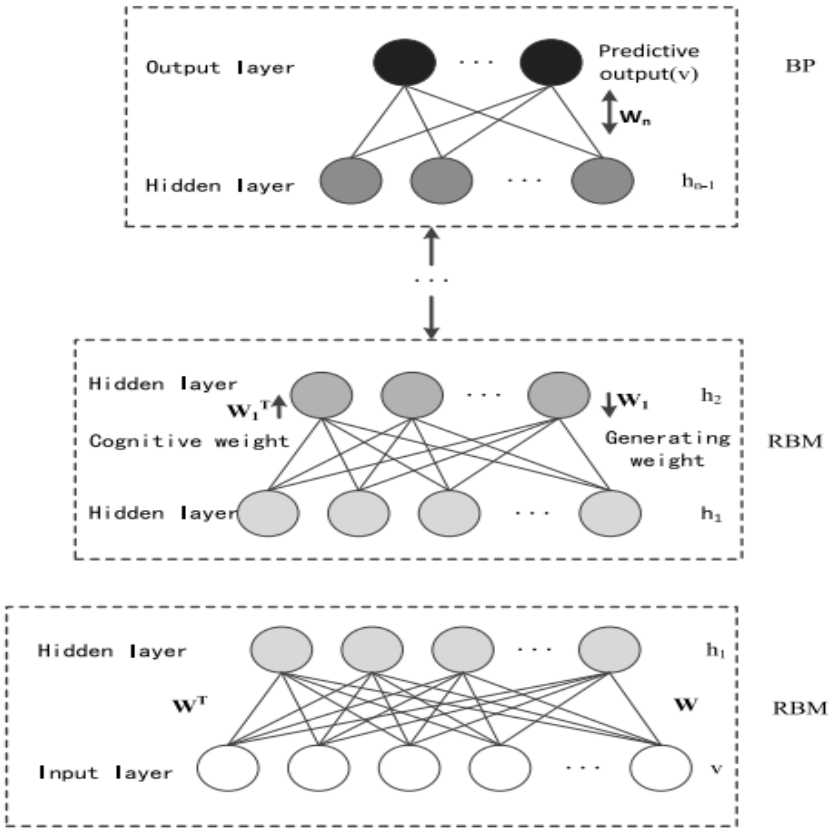

Fig.2. Architecture of Deep Belief Networks (WEI, et al., 2019)

Fig.2 above represents the multilayer generative structure of a DBN model with the undirected connections in the top two layers and directed top-down connections in the lower layers from the layer above. The recognition model is represented by the upward arrows, while the generative model is represented by the downward arrows. A DBN may be pre-trained very effectively using the layer-by-layer unsupervised, greedy learning technique contrastive divergence (CD), which is used to train the RBM of layers n in the first phase. The observable variables v are sampled by the posterior distribution p(|)1vh after the posterior p(|) hv1 is sampled from the first level RBM. Then, the hidden variables h1 are sampled one more using the same methodology. Up until a randomly approaching equilibrium distribution, the k steps of alternating Gibbs sampling were repeatedly conducted. After then, the second level RBM is learned, a sample is computed for h2, and so on through the final layer, using the optimum representation of the input vector v, h1 (WEI, et al., 2019).

The settings of the entire DBN are fine-tuned in the second phase. In the penultimate layer, the weights on the undirected connections which are located at the RBM top level are learned by adjusting and fitting the posterior distribution.

DBNs' capacity to autonomously learn meaningful representations of the data is one of their primary features. The hierarchical structure allows the network to capture both low-level and high-level features in an unsupervised manner.

3.1.3 Training of a DBN Model

The training and validation of the DBN model is a crucial step in the methodology of this study.

Training a DBN involves two major steps which are.

- Pre-training (Unsupervised learning method)
- Fine-tuning (Supervised)

A. Pre-training

During the pretraining stage, the RBMs are taught individually in a layer-wise manner. The first layer is trained, followed by the second layer on top and third until the last one in a greedy-like method (Guido, 2018). At this stage, the RBMs learn to model the probability distribution of the hidden units given the visible unit by adjusting the RBM weights iteratively using optimization methods such as contrastive divergence.

B. Fine-tuning

Fine-tuning of DBN can be done by backpropagation and SoftMax regression. Back-propagations are used to apply full network training with major parameters being batch size (the number of input samples in each of the group before performing updating the weight) and number of iterations (epochs) per tuning (Mazur, 2015). With a small iteration and a large iteration, backpropagation produces better finetuning. When using SoftMax regression on the other hand, the model determines the likelihood that an input belongs to each class before allocating it to the class with the highest likelihood. SoftMax regression is also known as the multinomial logistic regression used for handling analysis involving multiple classes (Ping, et al., 2018).

The Equation below displays the conditional probability of Y=l (class l) when given the output where **X'** =stacked Restricted Boltzmann Machines, **C** = The coefficient matrix, while **d** = intercept. The final prediction lastly is the class which has the highest probability.

$$P(Y = l|X', C, d) = \frac{e^{{c_l}^T x' + d_l}}{\sum_k e^{{c_k}^T x' + d_k}}$$

$$Prediction = max_{l \epsilon L}\{P(Y = l|X', C, d)\}$$

Fig.3. Conditional probability equation describing DBN model training (Ping, et al., 2018)

3.2 Data Collection Techniques

For this study, the dataset used is CSE-CIC-IDS2018 dataset compiled and made public for use by two Cybersecurity Research and development Units namely, CSE & CIC (Communication Security Establishment &Canadian Institute of Cybersecurity) (UNB, 2023).

The dataset is a combination of several intrusion attack types presented in Table 2. These data were gathered for 5 days from 420 machines from 5 different departments. Each attack set has over 80 features needed for effective network analysis.

The dataset was extracted using the CICFlowMeter-V3 (UNB, 2023) and CSV file format was downloaded using AWS command line interface (AWS, 2023) and windows command shell.

The data was grouped into ten subsets for the different days whereby the attack scenarios were performed.

The table below presents the sub datasets, attack types for each subgroup, sample size and date.

Table 2. Dataset Attack types and samples.

Sub dataset	Attack type	Sample	Date
Sub Data 1	Benign	1,048,574	16-02-2018
	DoS Hulk		
	DoS-Slow HTTP Test		
Sub Data 2	Benign	1,044,751	14-02-2018
	Brute Force-FTP		
	Brute Force-SSH		
Sub Data 3	Benign	1,040,548	15-02-2018
	DoS-GoldenEye		
	DoS-Slowloris		
Sub Data 4	Benign	7,889,295	20-02-2018
	DDoS-LOIC-HTTP		
Sub Data 5	Benign	1,048,575	20-02-2018
	DDoS-HOIC		
	DDoS-LOIC-UDP		
Sub Data 6	Benign	1,042,965	22-02-2018
	Brute Force -Web		
	Brute Force -XSS		
	SQL Injection		
Sub Data 7	Benign	1,042,867	23-02-2018
	Brute Force -Web		
	Brute Force -XSS		
	SQL Injection		
Sub Data 8	Benign Infiltration	606,902	28-02-2018
Sub Data 9	Benign Infiltration	328,181	01-03-2018
Sub Data 10	Benign	1,044,525	02-03-2018
	Bot		

3.3 Evaluation Metrics

For any study or research, the results generated are based on the metrics chosen. These metrics are used to determine the best outcome of the models.
For this research paper, the DBN model is evaluated with the following metrics listed (A-D).

A. Accuracy. This metric counts the proportion of cases that were successfully categorized out of all instances and the correctness of the data (MAOHUA, et al., 2022). It is determined as the proportion of occurrences that were correctly categorized to all the instances. That is summation of True Negative (TN) and True Positive (TP) divided by the sum of False Positive (FP), False Negative (FN), TP and TN. Below is the formular.

$$Accuracy : A = \frac{tp + tn}{tp + tn + fp + fn}$$ (Hercules, 2018)

B. Precision. This metric counts the proportion of true positive events among all positive occurrences. It is computed as the ratio of true positive cases to the total of both true positive and false positive instances (Alice, 2015). Below is the formular.

Precision = TP/ (TP + FP)

C. Recall. The Recall metrics is calculated as the ratio of the number of true positive to the sum of true positive and false negative. The formular is given below.

$$Recall = TP/ (TP + FN)$$ (Alice, 2015)

D. F1 score. This is a measure of the balance between precision and recall. In other words, it is the harmonic mean of precision and recall. The formular is given below.

$$F1\ score = 2 * (Precision * Recall) / (Precision + Recall)$$ (Md. Alamgir Hossain, 2023)

4 Design Specification

This section provides the description of the architecture used for the proposed detection method. The core stages/steps involved in the development and analysis of this study include data collection, Feature Engineering, Data pre-processing, performance evaluation, generating result of the DBN algorithm. Finally, comparison with other classical algorithms.
The various stages involved in the implementation of the brute force detection system from start to finish are explained in concise order for quality comprehension.

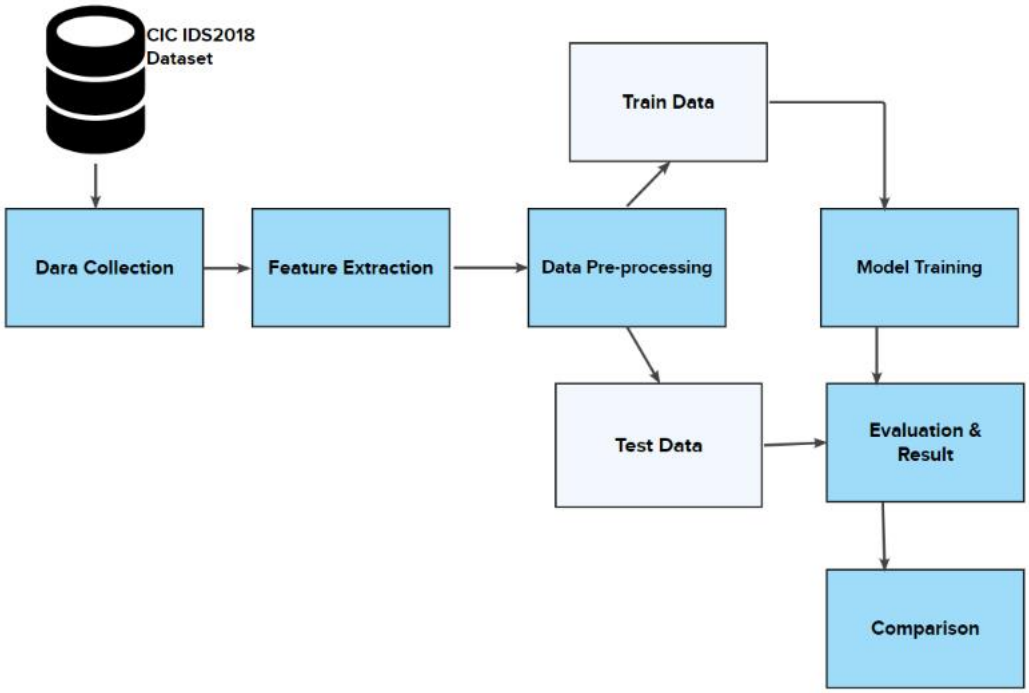

Fig.4. Design Specification Flow of the Model.

4.1 Data Collection

The first stage of the workflow involves searching and selection of the actual dataset to use for the analysis. For this, CSE-CIC-IDS2018 dataset was selected because the dataset has complete network features required for this analysis.

4.2 Feature Extraction

Feature extraction involves filtering the dataset and selecting only features represented in the forms that can be understood by the machine (Max & Kjell, 2019). Feature extraction before model training is essential to avoid ambiguity and resource overload during processing.

4.3 Data Pre-processing

This step involves cleaning the dataset, filtering, removing, formatting, and normalizing data to ensure it is suitable and has only required columns needed in the right format for analysis (Roy, 2022). Duplicates which exist were removed and checks for missing values were also carried out.
Additionally, encoding categorical variables was performed. In the concluding part of this stage, the data is split to Test data and Train data.

Test Data: The test data is a subset of the pre-processed data which is used to evaluate how the trained model performed. It is a mixture of both attack instances.

Train Data: The train data is the main dataset used to train the model. It consists of labelled examples of normal behaviour and attacks. The data is divided into input features such as IP addresses, login attempts and corresponding labels such as indicating whether an attack occurred or not.

4.4 Model Training

In this stage, train data is utilized to train the machine learning model. The model could be based on various algorithms. The model will learn to recognize patterns and make predictions based on the input features and labels provided in the train data.

4.5 Evaluation and Result

After the model is trained, the models need to be evaluated to assess the performance. The evaluation is typically done using the test data that was set aside earlier. The trained model is applied to the test data, and its predictions are compared to the known labels. Evaluation metrics can be used to measure how the model performed.
Once the evaluation is completed, results will be generated based on the outcome of metrics obtained from evaluation.

4.6 Comparison with Other Algorithms

In this stage, you can compare the results obtained from different models or approaches. For instance, the performance of the proposed method can be compared against different feature engineering techniques or machine learning algorithms.
For this study, below are the three classical algorithms which were trained and compared to the DBN model.

A. Random Forest

Random Forest is an ensemble learning technique which uses several decision trees to produce a single result. It is renowned for its resistance to overfitting and capacity for handling large-scale, multidimensional data (Pande S., 2021). Random Forest is a traditional Machine Learning classifier used often for handling classification and regression issues in high-dimensional data.

B. Decision Tree

Decision tree model has a tree-like structure in which each internal tree node from the root represents a question based on a specific feature or sequence of data splitting until a class label which is represented on a leaf node is obtained (Bahzad & Adnan, 2021). In general, they are well-known as machine learning or data mining technology that collects sets of attribute values (input) and produces a Boolean decision as output (Yang, 2019).

Decision trees are widely utilized in many other domains such as image processing, system anomaly detections and network interruptions.

C. Logistic Regression

This model predicts the likelihood of a binary result based on one or more predictor factors. Binary logistic regression is applied when the dependent variable has only two different values in which case, the predictor variables may be any type of variable such as either quantitative or qualitative (Hilbe, 2015). The model can also be used for multiclass classification with techniques like SoftMax regression.

5 Implementation

This section provides the actual work done starting from the selection of tools to the different processes carried out at each stage of the design specification workflow.

5.1 Research Tools

5.1.1 Lab

My personal computer (HP ProBook) with the following details was used for the analysis.

- OS: Windows 11 Pro.
- Version of PC: 21H2
- System: 64-bit OS, x64-based processor
- Processor of PC: 11th Gen Intel(R), Core (TM) i5-1135G7
- Storage: 512 GB
- RAM size: 16GB.

5.1.2 Installed Applications

- *AWS CLI*. This is an AWS command line tool used for both downloading, configuration and execution of OS shell commands (AWS, 2023). It was installed and used with the windows command shell to execute the commands for downloading the subject dataset.
- *CICFlowmeter V3* was used to extract the raw files format and convert them to CSV file format for further processing (UNB, 2023).
- *Anaconda and Jupyter Notebook.* Anaconda is a popular platform for python which includes JupyterNotebook, JupyterLab, RStudio, PyCharm etc. (Anaconda, 2023). JupyterNotebook is used for creating codes for web developments, visualizations, arithmetic programs, and machine learning (Jupyter, 2023). It creates faster and more scientific computing and allows for sharing codes with the output.
- *Python.* This is used widely for numeric and scientific calculations, web development, building applications etc. (Python, 2023) The choice of Python language for the development of the study is because of its versatility, flexibility, availability of libraries and machine learning/data science frameworks (Svitla, 2022).

5.1.3 Python Libraries

- *Pandas*. It is one of the foundation libraries in Python used in data manipulation, analysis, and visualization (Pandas, 2023). Pandas provides easy-to-use data structures, high performance calculations and data analysis framework.
- *NumPy*. This is also one of Python's Library used for data manipulations, data cleaning, transformation and provides efficient tool for working with structured data. It handles generic multi-dimensional data and provides tools for arrays such as statistical operations (NumPy, 2023).
- *Sklearn*. This Library package offers a vast selection of supervised machine learning techniques and unsupervised machine learning techniques. It provides a simple and efficient tool for data mining, regression, classification, and pre-processing (Igor, 2023).
- *Matplotlib*. Matplotlib is used for plotting representations during analysis (Matplotlib, 2023).

5.2 Dataset

5.2.1 Data Collection.

For the purposes of this study, the sub dataset "Sub Data 2" saved in the site as "**Wednesday-14-02-2018_TrafficForML_CICFlowMeter"** in Table 2. with three attack types Benign, Brute Force-FTP and Bruteforce SSH was loaded into the JupyterNotebook using the Pandas python library. Below is the count of the attack types.

```
In [10]: # number of attack labels
         df['Label'].value_counts()

Out[10]: Benign             667626
         FTP-BruteForce     193360
         SSH-Bruteforce     187589
         Name: Label, dtype: int64
```

Fig.5. Collected Dataset Labels and count.

A representation of the attack types and count was done using a bar chart created with the Python library matplotlib.

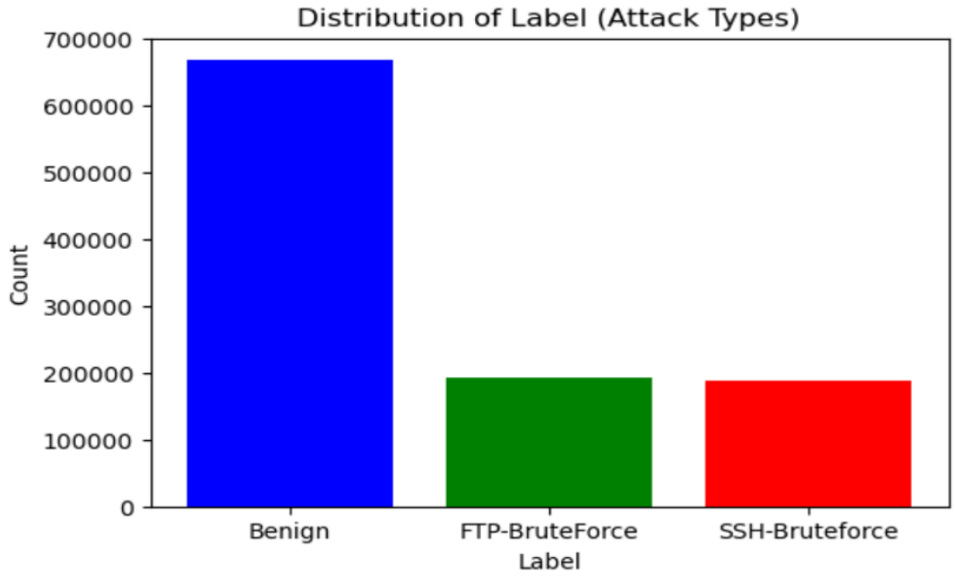

Fig.6. Bar plot of the Attack Labels and count.

5.2.2 Feature Extraction

The area of concentration on this study is on protocols FTP and SSH. Hence, during the feature extraction stage, only attack types-FTP-Bruteforce and SSH-Bruteforce were extracted. Also, out of the 80 features, only total of 69 of them were extracted specifically and used for the analysis. Below are the features extracted.

```
#check the current shape
df.shape
```

```
(380949, 69)
```

```
df.columns
```

```
Index(['Dst Port', 'Protocol', 'Flow Duration', 'Tot Fwd Pkts', 'Tot Bwd Pkts',
       'TotLen Fwd Pkts', 'TotLen Bwd Pkts', 'Fwd Pkt Len Max',
       'Fwd Pkt Len Min', 'Fwd Pkt Len Mean', 'Fwd Pkt Len Std',
       'Bwd Pkt Len Max', 'Bwd Pkt Len Min', 'Bwd Pkt Len Mean',
       'Bwd Pkt Len Std', 'Flow Byts/s', 'Flow Pkts/s', 'Flow IAT Mean',
       'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Tot',
       'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min',
       'Bwd IAT Tot', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max',
       'Bwd IAT Min', 'Fwd PSH Flags', 'Fwd Header Len', 'Bwd Header Len',
       'Fwd Pkts/s', 'Bwd Pkts/s', 'Pkt Len Min', 'Pkt Len Max',
       'Pkt Len Mean', 'Pkt Len Std', 'Pkt Len Var', 'FIN Flag Cnt',
       'SYN Flag Cnt', 'RST Flag Cnt', 'PSH Flag Cnt', 'ACK Flag Cnt',
       'URG Flag Cnt', 'ECE Flag Cnt', 'Down/Up Ratio', 'Pkt Size Avg',
       'Fwd Seg Size Avg', 'Bwd Seg Size Avg', 'Subflow Fwd Pkts',
       'Subflow Fwd Byts', 'Subflow Bwd Pkts', 'Subflow Bwd Byts',
       'Init Fwd Win Byts', 'Init Bwd Win Byts', 'Fwd Act Data Pkts',
       'Fwd Seg Size Min', 'Active Mean', 'Active Std', 'Active Max',
       'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min', 'Label'],
      dtype='object')
```

Fig.7. Dataset Features Extracted.

5.2.3 Data Pre-processing

After extracting the required features needed for building the model, the data was set up for cleaning. The positive infinite values, negative infinite values (INF, -INF) and not a number (NAN) were replaced with a generic number (999). This is to allow compatibility with the algorithms being developed (Christian, 2020). It enabled the data to remain in a consistent format allowing for easier manipulation.

The attack types were also converted into numerical values (0,1) using the python replace function. See below the code and output.

```
#changing the Label values of the dataset into numerical values
d1 = df.replace('FTP-BruteForce', 0)
d2 = d1.replace('SSH-Bruteforce', 1)
d_label = d2.Label.copy()
d_label.unique()
d_label.value_counts()
```

```
0    193360
1    187589
```

Fig.8. Label attack types converted to numerical.

Below is a view of the dataset with features after converting the Label.

```
d2.head()
```

	Dst Port	Protocol	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	Fwd Pkt Len Mean	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
94	21	6	19	1	1	0	0	0	0	0.0	...	40	0.0	0.0	0	0	0.0	0.0	0	0	0
95	21	6	3	1	1	0	0	0	0	0.0	...	40	0.0	0.0	0	0	0.0	0.0	0	0	0
96	21	6	3	1	1	0	0	0	0	0.0	...	40	0.0	0.0	0	0	0.0	0.0	0	0	0
97	21	6	2	1	1	0	0	0	0	0.0	...	40	0.0	0.0	0	0	0.0	0.0	0	0	0
98	21	6	2	1	1	0	0	0	0	0.0	...	40	0.0	0.0	0	0	0.0	0.0	0	0	0

5 rows × 69 columns

The features were also normalized to convert the minimum and maximum observable values into a similar range and avoid issue of having larger magnitude dominating the model training process using the python Sklearn functions Normalizer(), fit() and transform() (Scikit-learn, 2023).

Below is the code and the output features after the pre-processing.

```
# Normalize the feature values
scaler = Normalizer().fit(features)
features = scaler.transform(features)
```

```
features
```

```
array([[7.13777953e-04, 1.59461032e-04, 1.44274267e-04, ...,
        0.00000000e+00, 0.00000000e+00, 7.59338248e-06],
       [1.16287748e-04, 2.57057128e-05, 3.67224468e-06, ...,
        0.00000000e+00, 0.00000000e+00, 1.22408156e-06],
       [1.17511830e-04, 2.57057128e-05, 3.67224468e-06, ...,
        0.00000000e+00, 0.00000000e+00, 1.22408156e-06],
       ...,
       [6.84369803e-01, 3.92921825e-05, 1.07160498e-05, ...,
        0.00000000e+00, 0.00000000e+00, 1.78600830e-06],
       [1.54545849e-01, 8.46970216e-06, 4.03319150e-07, ...,
        0.00000000e+00, 0.00000000e+00, 4.03319150e-07],
       [7.38426198e-01, 4.23955373e-05, 1.34894891e-05, ...,
        0.00000000e+00, 0.00000000e+00, 1.92706988e-06]])
```

Fig.9. Normalized Dataset Features.

The last step of the pre-processing done was splitting the dataset to train data (80%) and test data (20%) using the imported Sklearn function train_test_split().

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)
```

Fig.10. Pre-processed dataset into Train and Test Data.

5.3 Model Training.

The DBN model, and the three other algorithms were trained using the train dataset and were evaluated using the test dataset.

Below is the list of the algorithms trained using the trained.

- Proposed Model: Deep Belief Network
- Other Algorithms: Random Forest, Logistic Regression and Decision Tree.

6 Evaluation

This section covers the experiments and results analysis from training the DBN model and the three models (Random Forest, Logistic Regression and Decision tree) using the dataset. The final performance result from the DBN model training was also compared to three algorithms.

6.1 Experiment 1: DBN Model Training

To train the DBN model, two parameters optimization was experimented during the finetuning to arrive at the final performance result. The experiments involve using components of RBM stack. Each RBM was taught to reconstruct its input during the pre-training, then the outputs were taken as the inputs to produce the output after finetuning.

A. First Parameter Experiment

n_components=400, learning_rate=0.001, n_iter=20, verbose=1

Learning Curve:

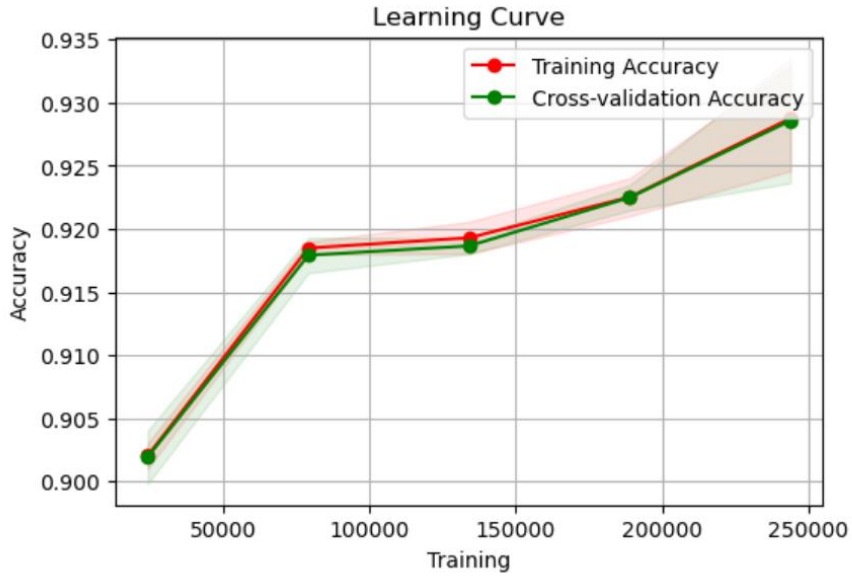

Fig.11. DBN Learning curve of the first experiment.

Confusion Matrix:

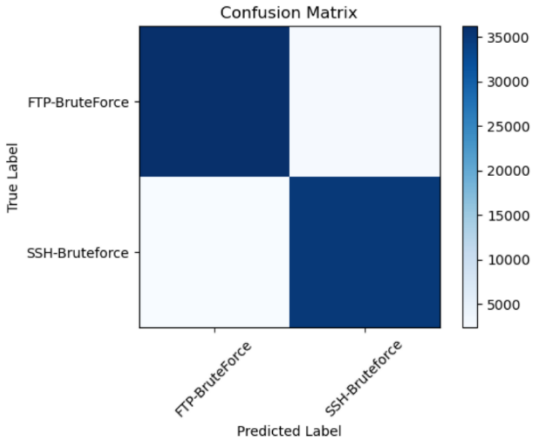

Fig.12. Confusion Matrix of DBN model.

Result from the experiment showed accuracy rate of 93.4%

Classification Report:

	precision	recall	f1-score	support
FTP-BruteForce	0.94	0.93	0.94	38866
SSH-Bruteforce	0.93	0.94	0.93	37324
accuracy			0.93	76190
macro avg	0.93	0.93	0.93	76190
weighted avg	0.93	0.93	0.93	76190

Fig.13. Classification report of DBN Model.

B. <u>Final Parameter Experiment/DBN Model Training</u>

n_components=400, learning_rate=0.0005, n_iter=50, verbose=1

The learning curve for the training is shown in Fig.13. below.

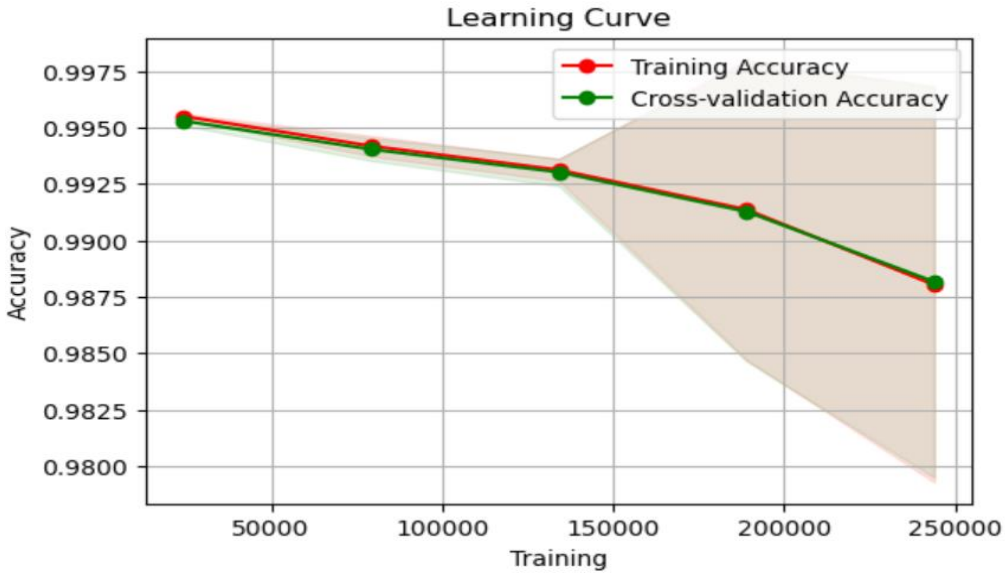

Fig.14. DBN Model Learning curve of the final parameter optimized.

Confusion Matrix:

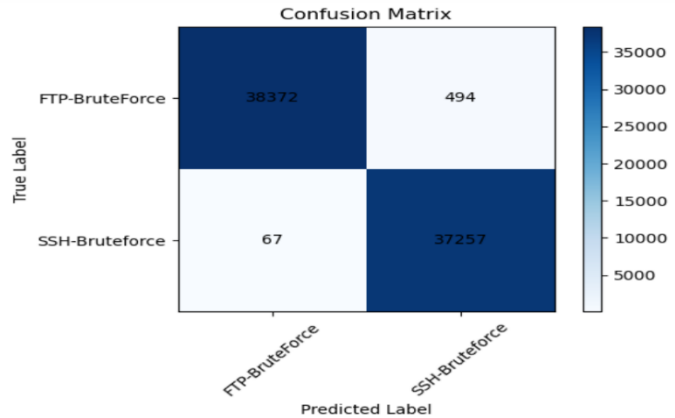

Fig.15. Confusion Matrix of the DBN Model.

Accuracy score:

```
accuracy = accuracy_score(y_test, y_pred)
print("Deep Belief Network-Accuracy is ", accuracy)
```

```
Deep Belief Network-Accuracy is  0.9926368289801811
```

Fig 16. Accuracy score of the DBN.

Classification Report:

	precision	recall	f1-score	support
FTP-BruteForce	1.00	0.99	0.99	38866
SSH-Bruteforce	0.99	1.00	0.99	37324
accuracy			0.99	76190
macro avg	0.99	0.99	0.99	76190
weighted avg	0.99	0.99	0.99	76190

Fig.17. Classification report of the DBN Model: Final optimization.

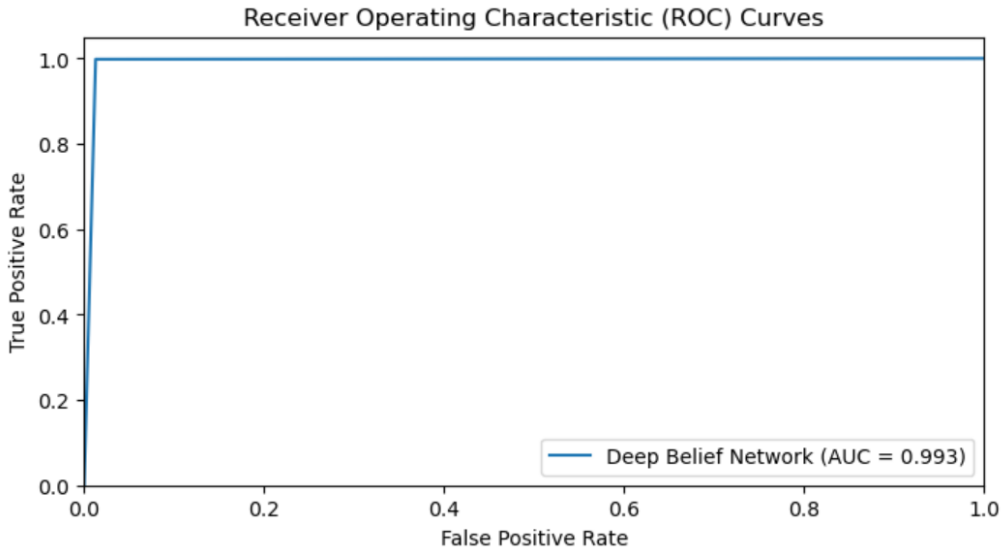

Fig.18. Learning Curve of DBN Model Training

6.2 Experiment 2: Other Algorithms

The same dataset was also used to train RandomForest, Logistic Regression and Decision tree. The accuracy results from the algorithms are presented below. Also, the classification report for each is displayed.

A. Random Forest

```
cm = metrics.confusion_matrix(expected, predicted)
print("Random Forest Set-Accuracy is ", RF_accuracy)
```

```
Random Forest Set-Accuracy is  0.9818742617141357
```

```
print (metrics.confusion_matrix(expected, predicted))
```

```
[[37488  1378]
 [    3 37321]]
```

Fig.19. Accuracy score and Confusion Matrix.

	precision	recall	f1-score	support
0	1.00	0.96	0.98	38866
1	0.96	1.00	0.98	37324
accuracy			0.98	76190
macro avg	0.98	0.98	0.98	76190
weighted avg	0.98	0.98	0.98	76190

Fig.20. Classification report.

B. Logistic Regression

Accuracy and Confusion matrix score.

```
cm = metrics.confusion_matrix(expected, predicted)
print("Logistic Regression Accuracy is ", LR_accuracy)
```

```
Logistic Regression Accuracy is  0.991757448484053
```

```
#confusion matrix
print(metrics.confusion_matrix(expected, predicted))
```

```
[[38313   553]
 [   75 37249]]
```

Fig.21. Accuracy Score & Confusion Matrix

Classification report.

	precision	recall	f1-score	support
0	1.00	0.99	0.99	38866
1	0.99	1.00	0.99	37324
accuracy			0.99	76190
macro avg	0.99	0.99	0.99	76190
weighted avg	0.99	0.99	0.99	76190

Fig.22. Classification report.

C. Decision Tree

Accuracy and confusion matrix score:

```
print("Decision Tree Accuracy  is ", DT_accuracy)
```

```
Decision Tree Accuracy  is  0.8579603622522641
```

```
#confusion matrix
print(metrics.confusion_matrix(expected, predicted))
```

```
[[31970  6896]
 [ 3926 33398]]
```

Fig.23. Accuracy score and Confusion Matrix.

Classification report:

	precision	recall	f1-score	support
0	0.89	0.82	0.86	38866
1	0.83	0.89	0.86	37324
accuracy			0.86	76190
macro avg	0.86	0.86	0.86	76190
weighted avg	0.86	0.86	0.86	76190

Fig.24. Classification Report.

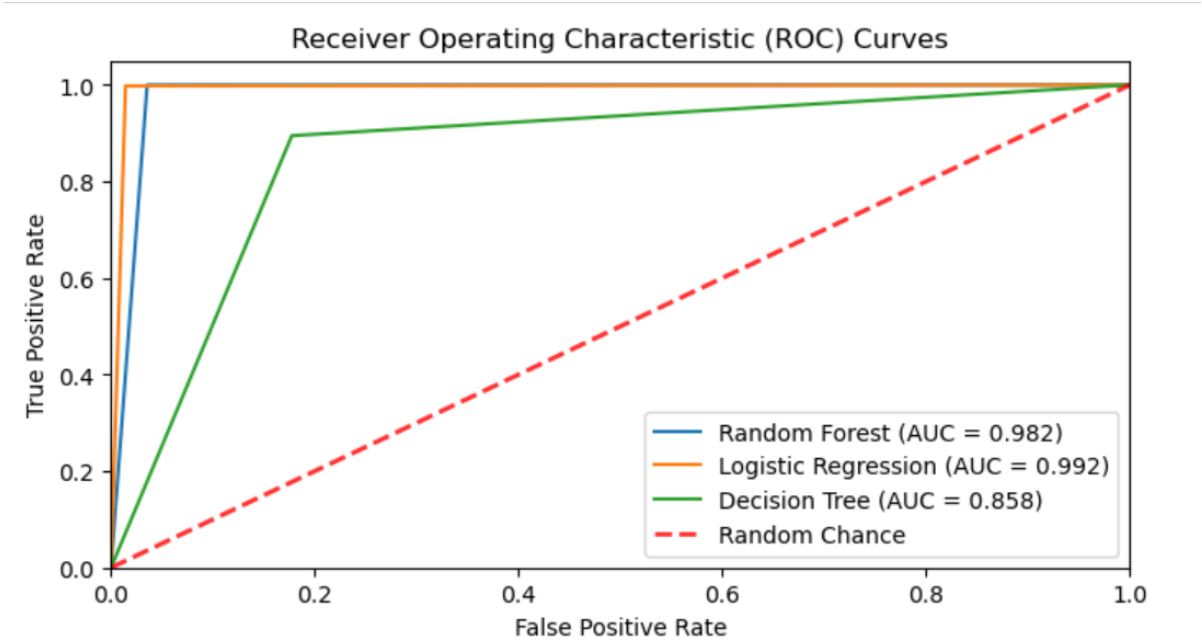

Fig.25. ROC Curve of the Algorithms.

6.3 Comparison

Having trained and evaluated the models to produce the output results, the summary of the performance scores can be presented in a comparison table for better illustration.

Table 3. AUC score and Accuracy score of the Models

Model	AUC (.3f)	Accuracy (Percentage)
DBN	**0.993**	**99.3%**
Decision Tree	0.858	85.8%
LogisticRegression	0.992	99.2%
RandomForest	0.982	98.2%

Table 4.

Attack type	Algorithm	Precision	Recall	f1-score
FTP	**Deep Belief Network**	1.00	0.99	0.99
	Decision Tree	0.89	0.82	0.86
	Logistic Regression	1.00	0.99	0.99
	Random Forest	1.00	0.96	0.98
SSH	**Deep Belief Network**	0.99	1.00	0.99
	Decision Tree	0.83	0.86	0.86
	Logistic Regression	0.99	1.00	0.99
	Random Forest	0.96	1.00	0.98

6.4 Discussion

The output results above show the learning curve, matrix and performance scores of the entire experiments carried out in the research. During the first experiments of the DBN model training, the accuracy of 93.4% was obtained. During the second experiment, the parameters n_components and learning rate were optimized which made the model learning perform better with an accuracy of 99.3%. This result exceeded that of the three classifiers trained. Random Forest 98.2%, Logistic regression 99.2% while Decision tree had the lowest accuracy of 85.8%. Also, when compared to the performance of the previous work done with the same dataset where CNN was applied on the detection of SSH-Bruteforce, accuracy was 94.3% (Wanjau, et al., 2021) This shows that DBN performs very well in detecting attacks.

Furthermore, the classification reports presented for each of the models show that the precision score, recall score and f1-score of the DBN model is higher than that of RandomForest, Logistic Regression and Decision tree. The ROC curve presented showed that the AUC score of DBN model is 0.993, Random Forest = 0.982, LogisticRegression = 0.992, while Decision Tree 0.858.

Overall, the design and performance of the proposed DBN model was good. To obtain a high-performance rate of the model, more layers of the RBM had to be trained with a very low learning rate of 0.0005 and higher iteration (50) thereby causing longer period.

Hence, there is need for more studies to be explored in developing a DBN training mechanism with shorter period.

7 Conclusion and Future Work

The purpose of this research paper is to demonstrate that Deep Belief Network can be successfully trained and applied to detect of brute force attacks. The model learns high-dimensional representations and it also performs classification tasks which in this case was needed while using the dataset CSE-CICIDS2018, a big dataset with multiple features and

attack types. The DBN model was pre-trained with a multilayered unsupervised RBM and fine-tuned.

The result showed that the model performed well and produced an accuracy score of 99.3% when the parameters were optimized. Furthermore, three classification algorithms were also trained using the same dataset. Decision tree scored the accuracy rate of 85.8%, Logistic Regression 99.2% and Random Forest 98.2% showing that the accuracy rate of the DBN model is better. Therefore, it answered the research question confirming that the deep belief network model can perform very well in detecting FTP-Bruteforce and SSH-Bruteforce attacks. Also, it performs better than traditional machine learning methods when compared. The limitation of this model however is that the training process which involves pre-training stacks of RBM, and fine-tuning took longer time when compared to the other methods.

In the future, I intend to explore more on pre-training and fine-tuning methods of training a Deep Belief Model to optimize the result in a shorter timeframe.

References

Ahmad, A., Harjula, E., Ylianttila, M. & Ahmad, I., 2020. Evaluation of Machine Learning Techniques for Security in SDN. *Globecom Workshops (GC Wkshps)*, 978-1-7281-7307-8/20(DOI: 10.1109/GCWkshps50303.2020.9367477), p. 6.

Ahmed, F. O., Wafa', E. & Emad, E. A., 2023. Deep Learning for Accurate Detection of Brute Force attacks on IoT. *ScienceDirect Procedia Computer Science 220*, 14(The 14th International Conference on Ambient Systems, Networks and Technologies (ANT)), p. 291–298.

Alatwi, H. A. & Morisset, C., 2022. Threat Modeling for Machine Learning-Based Network Intrusion Detection Systems. *IEEE International Conference on Big Data (Big Data)*, 78-1-6654-8045-1(DOI: 10.1109/BigData55660.2022.10020368), p. 10.

Alice, Z., 2015. Evaluation Metrics. In: S. Cutt, ed. *Evaluating Machine Learning Models.* 1 ed. 1005 Gravenstein Highway North, Sebastopol, CA: O'Reilly Media, Inc.,, pp. chp 7, pg 12-13.

Anaconda, 2023. *Anaconda.* [Online]
Available at: https://www.anaconda.com/
[Accessed 01 06 2023].

AWS, 2023. *AWS Command Line Interface.* [Online]
Available at: https://aws.amazon.com/cli/
[Accessed 01 04 2023].

Bahzad, T. J. & Adnan, M. A., 2021. Classification Based on Decision Tree Algorithm for Machine Learning. *Journal of Appliend Science and Technology Trends*, 02(1), pp. 20-28.

Bo, D. & Xue, W., 2016. Comparison Deep Learning Method to Traditional Methods Using for Network. *IEEE International Conference on Communication S oftw are and N etw ork s*, Volume 8th, p. 5.

Bronte, R., Shahriar, H., Haddad, H. M. & Claims, A. I. &., 2016. A Signature-Based Intrusion Detection System for Web Applications based on Genetic Algorithm. *Proceedings of the 9th International Conference on Security of Information and Networks*, SIN 16(2947626.2951964), pp. 32-39.

Christian, H., 2020. *Learning Scientific Programming with Python.* 2nd ed. UK: Cambridge University. ISBN: 9781108745918.

Deris, S. et al., 2019. Investigating Brute Force Attack Patterns in IoT Network. *Journal of Electrical and Computer Engineering,* 2019(4568368; Article ID 4568368), p. 13.

F. Pedregosa, G. V. A. G. V. M. B. T. O. G. B. P. P. R. W., 2011. Scikit-learn:Machine learning in python. *Journal of machine Learning Research,* Volume 12, p. 2825–2830.

Fernandez, G. C. & Xu, S., 2019. A Case Study on Using Deep Learning for Network Intrusion Detection. *arXiv Preprints,* 1(arXiv:1910.02203v1), p. 6.

Guido, M., 2018. Restricted Boltzmann Machines: Introduction and Review. 1(1), p. 41.

Günter, F., 2022. Realtime Risk Monitoring of SSH Brute Force Attacks. *22nd International Conference on Innovations for Community Services,* 22(DOI: 10.1007/978-3-031-06668-9_8), p. 23.

Gupta, B. B. & Srinivasagopalan, S., 2020. Handbook of Research on Intrusion Detection Systems. In: S. S. Brij Gupta, ed. *A volume in Advances in Information Security Privacy and Ethics (AISPE) Book series.* Pennysylvania USA: IGI Global, p. 68.

Hercules, D., 2018. Evaluation Metrics and Evaluation. In: D. Hercules, ed. *Evaluation Metrics and Evaluation.* Salmon Tower Building New York City: Springer International Publishing, pp. chp 6. pg 1-9.

Hilbe, J. M., 2015. *Practical Guide to Logistic Regression.* 1 ed. 6000 Broken Sound Parkway NW, Suite 300: Taylor & Francis Group.

Hossain, M. D., Ochiai, H. & Fall Doudou, Y. K., 2020. SSH and FTP brute-force Attacks Detection in Computer Networks: L STM and Machine L earning Approaches. *International Conference on Computer and Comunication System,* 5(978-1-7281-6136-5/20), p. 7.

Igor, R., 2023. *Sklearn – An Introduction Guide to Machine Learning.* [Online]
Available at: https://algotrading101.com/learn/sklearn-guide/
[Accessed 07 08 2023].

Jan, L., Karel, H. & Tomas, C., 2021. Detection of HTTPS Brute-Force Attacks with Packet-Level Feature Set. *IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC),* 78-1-6654-1490-6(DOI: 10.1109/CCWC51732.2021.9375998), p. 9.

Jeonghoon, P. et al., 2021. Network Log-Based SSH Brute-Force Attack Detection Model. *Computers, Materials & Continua; Tech Science Press,* 68(1), p. 15.

Jing, L. et al., 2017. A Review on Signature-Based Detection for Network Threats. *IEEE International Conference on Communication Software and Networks,* 9(9th), p. 5.

John, H., Taghi, M. K. & Leevy, J. L., 2021. Detecting SSH and FTP Brute Force Attacks in Big Data. *IEEE International Conference on Machine Learning and Applications (ICMLA),* 20(DOI: 10.1109/ICMLA52953.2021.00126), p. 6.

Jupyter, 2023. *Jupyter.* [Online]
Available at: https://jupyter.org/
[Accessed 01 06 2023].

Karel, H., Tomas, B., Tomas, C. & Hana, K., 2020. Refined Detection of SSH Brute-Force Attackers Using Machine Learning. *IFIP International Federation for Information Processing,* 2020(M. Holbl et al. (Eds.): SEC 2020, IFIP AICT 580, pp. 49–63, 2020.), p. 15.

Khraisat, A., Gondal, I., Vamplew, P. & Kamruzzaman, J., 2019. Survey of intrusion detection systems: techniques, datasets and challenges. *Open Access,* 1(1), p. 22.

Kim, J., Shin, Y. & Choi, E., 2019. An Intrusion Detection Model based on a Convolutional Neural Network. *Journal of Multimedia Information System ,* 6; No 4(ISSN 2383-7632 (Online)), pp. 165-172.

Liu, J. et al., 2021. Deep Anomaly Detection in Packet Payload. *arXiv Preprints,* 1(1912.02549), p. 27.

Luo, X., Yao, C. & Zincir-Heywood, A. N., 2021. Modelling and visualising SSH brute force attack behaviours through a hybrid learning framework. *International Journal of Information and Computer Security,* 16(1-2), pp. 170-191.

MAOHUA, G., ZEYNEP, Y. & AKITO, M., 2022. Improvement and Evaluation of Data Consistency Metric CIL for Software Engineering Data Sets. *IEEE Access,* 3188246 (Digital Object Identifier 10.1109/ACCESS.2022.3188246), p. 15.

Matplotlib, 2023. *Matplotlib: Visualization with Python.* [Online]
Available at: https://matplotlib.org/
[Accessed 07 08 2023].

Max, K. & Kjell, J., 2019. *Feature Engineering and Selection: A Practical Approach for Predictive Models.* 1 ed. U.S: Taylor & Francis Group, LLC. .

Mazur, M., 2015. *A Step by Step Backpropagation Example.* [Online]
Available at: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
[Accessed 11 06 2023].

Md. Alamgir Hossain, M. S. I., 2023. Ensuring network security with a robust intrusion detection system using ensemble-based machine learning. *Array,* 19(https://doi.org/10.1016/j.array.2023.100306.), p. 14.

Montoya, V. D. A., Montana, V. D. F., Portela, F. G. & Diaz, T. O. A., 2022. Intrusion Detection System (IDS) with anomaly-based detection and deep learning application. *IEEE Xplore Digital Library,* 1(1), p. 4.

Numpy, 2023. *Numpy.* [Online]
Available at: https://numpy.org/
[Accessed 30 06 2023].

Pandas, 2023. *Pandas.* [Online]
Available at: https://pandas.pydata.org/
[Accessed 30 06 2023].

Pande S., K. A. G. D. a. T. D., 2021. DDOS Detection Using Machine Learning Techinques. *Studies in Computational Intelligence,* 921(doi: org/10.1007/978-981-15-8469-5_5), pp. pp. 59-68.

Ping, D. et al., 2018. Softmax Regression by Using Unsupervised Ensemble Learning. *2018 9th International Symposium on Parallel Architectures, Algorithms and Programming,* 9(doi: 10.1109/PAAP.2018.00041), pp. 196-201.

Python, 2023. *Applications for Python.* [Online]
Available at: https://www.python.org/about/apps/
[Accessed 07 01 2023].

Quincozes, S. E., Albuquerque, C., Passos, D. & Mossé, D., 2021. A survey on intrusion detection and prevention systems in digital substations. *Computer Networks,* 184(107679), p. pg 7.

Roger, A. G., 2020. Brute-Force Attacks. In: *Hacking Multifactor Authentication.* s.l.:Wiley Data and Cybersecurity, pp. 295 - 306.

Roy, J., 2022. *Hands-On Data Preprocessing in Python.* 1 ed. Birmingham, United Kingdom: Packt Publishing.

Saumya, B. & Tamanna, M., 2021. Hybrid_Intrusion_Detection_System_using_an_Unsupervised_method_for_Anomaly-based_Detection. *EE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* , 978-1-6654-4893-2/21/(1), p. 6.

Scikit-learn, 2023. *SKLearn Preprocessing Normalize.* [Online]
Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html
[Accessed 11 06 2023].

Svitla, 2022. *Machine Learning with Python.* [Online]
Available at: https://svitla.com/blog/machine-learning-with-python-best-libraries-tutorials-use-case-examples
[Accessed 01 06 2023].

UNB, 2023. *CICFlowMeter (formerly ISCXFlowMeter).* [Online]
Available at: https://www.unb.ca/cic/research/applications.html#CICFlowMeter
[Accessed 01 04 2023].

UNB, 2023. *CSE-CIC-IDS2018 on AWS.* [Online]
Available at: https://www.unb.ca/cic/datasets/ids-2018.html
[Accessed 01 04 2023].

Wanjau, S. K., Wambugu, G. M. & Kamau, G. N., 2021. SSH-Brute Force Attack Detection Model based on Deep Learning. *International Journal of Computer Applications Technology and Research,* 10(01; ISSN:-2319–8656), pp. 42-50.

WEI, P. et al., 2019. An Optimization Method for Intrusion Detection Classification Model Based on Deep Belief Network. *IEEE,* 1(1), p. 13.

Wichmann, P., Marx, M., Federrath, H. & Fischer, M., 2021. Detection of Brute-Force Attacks in End-to-End Encrypted Network Traffic. *The 16th International Conference on Availability, Reliability and,* 16(https://doi.org/10.1145/3465481.3470113), p. 9.

Wu, Y. et al., 2020. Mining Threat Intelligence from Billion-scale SSH Brute-Force Attacks. *Workshop on Decentralized IoT Systems and Security (DISS) 2020,* ISBN 1-891562-64-9(1), p. 7.

Yang, F.-J., 2019. An Extended Idea about Decision Trees. *International Conference on Computational Science and Computational Intelligence (CSCI),* 1(1), p. 6.

Yin, C., Zhu, Y., Fei, J. & He, X., 2017. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Network. *IEEE Access,* 5(doi: 10.1109/ACCESS.2017.2762418), pp. 21954 - 21961.