Attention based Convolutional Neural Network for Network Intrusion Detection System

Danial Chakma, Rahat Samit and Md. Shohrab Hossain
Department CSE, Bangladesh University of Engineering and Technology,
Dhaka, Bangladesh
Email: danial08cse@gmail.com, rahathsamit@gmail.com, mshohrabhossain@cse.buet.ac.bd

***Abstract*—Network Intrusion Detection System (NIDS) helps detecting cyber attacks. However, these cyber attacks do not follow a particular or familiar trend all the time. Therefore, there are substantial challenges to develop an efficient NIDS which prevents high false alarm rates and low detection accuracy against unfamiliar attacks. In this paper, we propose a deep learning-based method to implement an effective NIDS. We incorporated and implemented attention mechanism into our convolutional neural network (CNN) based model, referred as A-CNN, on benchmark CICID dataset, containing benign and the most up-to-date common attacks, which resembles the true real-world data (PCAPs). Our Experiment against the CICID shows that our model generate best performance with learning rate 0.01 using Adagrad optimizer and achieves overall 99.96% and 99.82% accuracy on training and validation set respectively against this dataset and shows A-CNNs can be applied as a learning method for Intrusion Detection Systems(IDSs).**

***Index Terms*—Cyber security, Network security, NIDS, deep learning, CNN.**

I. Introduction

For the last decades, information and communication technology (ICT) have been making great progress. With this continuous development of ICT, it is efficiently making peoples' lives more convenient. However, over the years, cyber crimes have got more advanced and are being highly funded. These days, there are entire teams of highly trained hackers, and they have made it a very profitable business [1]. As a result, cyber crime has become an important social issue.

The principle of defense-in-depth is that layered security mechanisms increase security of the system as a whole. If an attack causes one security mechanism to fail, other mechanisms may still provide the necessary security to protect the system. For example, it is not a good idea to totally rely on a firewall to provide security for an internal-use-only application, as firewalls can usually be circumvented by a determined attacker requiring a physical attack or a social engineering attack of some sort. Other security mechanisms should be added to complement the firewall, thereby protecting against different other attack vectors. Although both relate to network security, an IDS differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An IDS describes a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system. This is traditionally achieved by examining network communications, identifying heuristics and patterns which often known as signatures of common computer attacks and taking action to alert an intrusion.

An IDS is a detection system put in place to monitor computer networks. By analysing patterns of captured data from a network, IDS help to detect threats [2]. These threats include, for example, Denial of service (DoS) which denies or prevents legitimate users from usage of resource on a network by introducing unwanted traffic [3]. Malware is another example, where attackers use malicious software to disrupt systems [4].

IDSs are capable of comparing signatures for similar packets to link and drop harmful detected packets which have a signature matching the records in the NIDS. We can classify the design of the NIDS according to the system interactivity property, there are two types: on-line and off-line NIDS, often referred to as inline and tap mode, respectively. On-line NIDS deals with the network in real time. It analyses the ethernet packets and applies some rules, to decide if it is an attack or not. Off-line NIDS deals with stored data and passes it through some processes to decide if it is an attack or not.

IDS can be also combined with other technologies to increase detection and prediction rates. Artificial Neural Network based IDS are capable of analyzing huge volumes of data, in a smart way, due to the self-organizing structure that allows IDS to more efficiently recognize intrusion patterns. Neural networks assist IDS in predicting attacks by learning from mistakes. Signature-based IDS refers to the detection of attacks by looking for specific patterns, such as byte sequences in network traffic or known malicious instruction sequences used by malware. Anomaly-based intrusion detection systems were primarily introduced to detect unknown attacks, in part due to the rapid development of malware. The basic approach is to use machine learning to create a model of trustworthy activity and then compare new behavior against this model. Since these models can be trained according to the applications and hardware configurations, machine learning based method has a better generalized property in comparison to traditional signature-based IDS. Although this approach enables the detection of previously unknown attacks, it may suffer from false positives: previously unknown legitimate activity may also be classified as malicious. Most of the existing IDSs suffer from the time-consuming during detection process that degrades the performance of IDSs. Efficient feature selection algorithm makes the classification process used in detection

more reliable.

Most of the existing research works have employed different techniques and approaches including SVM [**?**], MLP [**?**], Random Forest [], Regression, CNN [], RNN [], etc. which is discussed in Section II. Attention mechanism was introduced in early 2016 and was developed and thoroughly investigated in NLP tasks. However, to the best of our knowledge, it is not yet investigated in network security domain, specifically for intrusion detection system. Our work *differs* from others approaches in this regard and we have empirically studied the effectiveness of attention mechanism by incorporating it into the Convolutional Neural Network.

The *contributions* of this work are threefold: (i) Investigating the effectiveness of attention mechanism in detecting Network Intrusion. (ii) minimization of false alarm generation by empirical search of hyper-parameters and feature engineering, and (iii) Automated generation of synthetic data from existing dataset (CICID) using Generative Adversarial Network(GANs) and test the performance of our fine tuned model against these synthetic data.

To the best of our knowledge this is the *first application of attention mechanism* in network intrusion detection research papers which automatically deserves further investigation whether attention mechanism is actually capable to capture feature dependency among long distant flows.

The rest of the paper is organized as follows. Section II describes related works. Section III describes our model architecture and approach. Section IV describes the dataset and feature extraction process. Section V presents experimental result and performance analysis. Finally, Section VI provides concluding remarks and future works.

II. Background and Related Works

A lot of machine learning solutions to network intrusions resulted in achieving the high false positive rate with high computational cost [5]. This is due to the fact that most of the solutions have limited the learning patterns of attack locally with small-scale, low-level features patterns of normal and attack connections records. Most notably, machine learning has given birth to a new area called deep learning that can be defined as a complex model of machine learning algorithms. These can learn the abstract representation of complex hierarchical features globally with sequence information of TCP/IP packets.

In this section, we discuss previous attempts that applied machine learning and deep learning techniques to solve the existing problems at intrusion detection.

A. Machine learning techniques

Most of the machine learning based work used Bayesian network, genetic algorithm and support-vector machine as explained below.

1) Using Bayesian networks: Onik et al. [6] conducted an experiment using Bayesian networks on NSL-KDD dataset containing 25,192 records with 41 features. Apart from the normal class label it contained 4 more class of attacks known as Dos, U2R, R2L and probe attacks. The filter approach of feature selection was used to reduce the dataset features from 41 to 16 important features. Bayesian model was built and proved to predict attacks with superior overall performance accuracy rate of 97.27% keeping the false positive rate at a lower rate of 0.008. The model as compared to Naïve Bayes, K-means clustering, decision stamp and RBF network recorded 84.86%, 80.75%, 83.31% and 91.03% respectively in terms of accuracy.

Another experiment by Bode et al. [7] analysed the network traffic in a cyber situation with Bayesian network classifier on the KDD Cup'99 dataset with 490,021 records. The dataset was made up of 4 types of attacks (DoS, U2R, R2L and probe). In this experiment, they adopted the risk matrix to analyse the risk zone of the attacks. The risk analysis adopted showed DoS was most frequent attack in occurrences. The results showed Bayesian network classifier is a suitable model resulting in same performance level classifying the DoS attacks as association rule mining. Bayesian network classifier outperformed Genetic Algorithm in classifying probe and U2R attacks and classified Dos equally.

2) Using Genetic Algorithm: Xia et al. [8] developed a hybrid method based information theory and Genetic Algorithm (GA). Information theory was used to filter out the most important features out of 41 features in KDD'99 dataset with 494021 records. Linear rule was used initially to classify normal and abnormal before GA to obtain the appropriate classification. The training was filtered to keep key features, such as rendce, logged-in, protocol-ppe and flag. The number of features in the new training data is decreased from 41 to 4. In the detection of Dos, U2R, R2L and probe attacks, the information theory and GA hybrid method recorded 99.33%, 63.64%, 5.86% and 93.95% as detection rates respectively. The detection rates were compared to Ctree(classification tree) and C5. Ctree recorded 98.91%, 88.13%, 7.41% and 50.35% whilst C5 recorded 97.1%, 13.2%, 8.4% and 83.3%, respectively for the attacks.

3) Using Support-vector machine: Senthilnayaki et al. [9] used Genetic Algorithm pre-processed KDD Cup 99 dataset in a pre-processing module for data reduction since it was complex to process the dataset with all 41 features. GA was used to select 10 features out of 41 features present in the KDD Cup 99 dataset and applied SVM for classification. The experiment was carried out with 100,000 records from the dataset out of which 95% was used as training data and remaining 10% as test data. The classification process continued till a 10 fold cross validation was done for results verification. The SVM classified four different attacks (DoS, probe, U2R, R2L attacks). The performance of SVM classifier was compared with four other classifiers are: for 41 features Gmm 30.5%, Naive 45.17%, MLP 55.24%, Linear 65.47% SVM 82.45%.

B. Deep learning techniques

1) Using RNN: A study by Jihyun et al. [10] applied recurrent neural network with hessian-free optimization. They

extracted features from DARPA dataset. Then they used 32 features as an input vector of the intrusion detection model. The hyper parameters used for model training and experiments were as follows:

- hidden unit = 45
- activation function = hyperbolic tangent
- batch size = 1000
- step size = 50
- epochs = up to 1,000

The authors reported that detection rate of their model was around 95.37 % and false alarm rate was only 2.1 %.

2) Using CNN and its variants: Vinayakumar et al. [11] model network traffic as time-series, particularly TCP/IP packets in a predefined time range with supervised learning methods such as multi-layer perceptron (MLP), CNN, CNN-recurrent neural network (CNN-RNN), CNN-long short-term memory (CNN-LSTM) and CNN-gated recurrent unit (GRU), using millions of known good and bad network connection. They used a CNN model with multiple layers and hybrid CNN model with RNNs,LSTMs, and GRUs for network anomaly detection with the KDDCup 99 dataset. There were three layers consisting of an input layer, a hidden layer, and an output layer, and the hidden layer had one or more CNN layer(s) followed by traditional Feed Forward Networks (FFNs) or three other deep learning models as mentioned above. The proposed model was implemented in TensorFlow, and hyper parameters used for model training and experiments were as follows:

- inputs = 41 * 1
- filters = 16, 32, and 64
- filter length = 3 and 5
- learning rate = the range of 0.01 through 0.5
- epochs = up to 1,000

In the experiments for binary classification with the KDD-Cup 99 dataset, both 1-layer and 2-layer CNN models showed 99% of the highest F-measure accuracy.

Li et al. [12] proposed an image conversion method of NSL-KDD data. They used Convolutional neural networks(CNN) model that learnt the features of graphic NSL-KDD transformation via the proposed graphic conversion technique. The first step in this model was feature extraction. A total of 41 extracted features were passed to the second step for data preprocessing. Three categorical features, protocol type, flags, and service, were converted into binary vectors through one-hot encoding, and numerical features were normalized by standard scaler. By doing so, the dataset was converted into binary vectors with 464 dimensions, and then these vectors were converted into 8*8 gray-scale pixel images. In the last step, two popular CNN models, ResNet [13] and GoogLeNet/Inception [14], were employed to identify the category of image conversion. The number of epochs and batch size, as well as the information of the optimizer and loss function to train both models was as follows:

- epochs = 100 for both models
- batch size = 256 for ResNet 50 and 62 for GoogLeNet
- optimizer = gradient descent
- loss function = cross entropy

The authors reported that both of ResNet and GoogLeNet show around 90% of accuracy using the metric of F-measure, against the NSL-KDD datasets.

III. Proposed Architecture and Methodology

We adopted attention based Convolutional neural network to detect network intrusion. Our model utilizes the sparsity of flow based intrusion detection dataset. In general, flow based intrusion detection system uses network traffic flow for feature extraction from raw packet. Data set constructed from those features are sparse in nature. In those cases, application of attention in neural network is typically useful for sparse dataset not only for faster training but also for obtaining good generalization of neural network for future unknown data. With this in mind, our proposed architecture consist of convolutional layer for feature extraction followed by one attention layer and then couple of feed forward layer for classification of network intrusions. Our proposed model divided into four basic parts: (a) Feature Extraction layer (b) Intermediate Normalization Layer (c) Attention Layer (d) Feed forward Classification layer.

a) Feature Extraction Layer: Feature extraction is one of the important aspects in machine learning based detection or prediction system. We have seen Convolutional neural network performs well in extracting image features in Computer visions and many other fields. We have used two convolution one dimensional layer, in both of layer rectified liner unit (RELU) is used. First convolutional layer consist of 78 filters(also called number of kernel) each of size 32(also called kernel size or sliding window size) and each kernel is slided or strided by 2 along the input dimension. With the defined kernel size and input size, the layer will output a matrix of shape or size 24 by 78, each column of which holds weight of single filter. In our case, each learn-able filter contains 24 trainable weight. In general, each filter is sufficient to train one feature of input data tuple. As activation function, we used rectified linear unit (ReLUs) in this layer. Similarly, at the next layer we used similar settings except 16 kernel size instead of 32. The reason for using strides of 2 in sliding the window across input dimension is, sparse data set which means only a few dimension contains value and all other are zeros in a data tuple or example. So, strides of 1 or more will not affect much in generalizing network towards ground truth.

b) Intermediate Normalization Layer: After feature extraction layer, a Maxpooling layer followed by one Batch Normalization and ReLUs layer is introduced which acted as intermediate bridge between Feature extraction layer and Attention layer. Batch Normalization is crucial for deep neural network not only for network stabilization but also to solve over-fitting and under-fitting problems during training deep network. At first, we were undecided whether to employ Maxpooling before Batch Normalization+Activation or not, but after some study we decided to go with Batch Norm+Activation+Maxpool.

c) Attention Layer: Following the feature extraction layer, we introduced a attention layer which is critical component in our network architecture. Attention mechanism in neural network was born to solve long distance dependency problem in machine translation (MT) from source language to target language. The intuition behind attention came from how human pay visual attention to different region of object such as image or words in one sentence. Human visual attention usually focuses on a certain part with "high resolution or attention" while perceiving surrounding part in "low resolution or attention" and then adjust the focal point or do inference on the object accordingly. In a words, we do not need all aspect to recognize object only important one will suffice. Motivated from this, the attention mechanism was introduced in [15]–[18], via applying some weight on the input data. We used additive attention mechanism, our attention layer consist of 32 dimensional attention vector, local attention width of 16, and softmax activation function. Attention vector computation formula is given below:

$$h_t = \tanh(x_t^T W_t + x_t^T W_x + b_h) \tag{1}$$

$$e_t = h_t^T W_a + b_a \tag{2}$$

$$a_t = \frac{\exp(e_t)}{\sum_{i=1}^{N} \exp(e_t(i)) + \epsilon} \tag{3}$$

Where a_t attention vector, W_a, W_t, and W_x are trainable weight matrices, b_h, b_a are trainable biases and N is input layer number in attention layer, $\epsilon = \exp(-7)$ is added to avoid divide by zero run-time errors. And finally, attention-focused value (output) v_t to the next layer is computed as the weighted sum of the input using following equation:

$$v_t = \sum_{t'=1}^{n} a_{t,t'} x_{t'} \tag{4}$$

Where $a_{t,t'}$ is the attention score indexed at t and t' computed using eq. 3 and $x_{t'}$ is the input value indexed at t'.

d) Classification Layer: Classification Layer constitute of three consecutive dense layer one after another with exception before last dense layer in which a drop-out+flatten layer introduced just before final dense layer. Drop-out proven to be useful for faster training and reduction of over-fitting during training. In each dense layer, as activation function we employed LeakyReLUs except the final layer where softmax is employed to produce a probability distribution over the target class. First two dense layer employed 128 and 64 neurons or units respectively whereas last layer witnessed 6 neurons or units as we had 6 different target classes in data set, five malicious and one benign class. Activation function used:

$$f(x) = \begin{cases} ax, & \text{if } x < 0. \\ x, & \text{if } x >= 0. \end{cases} \tag{5}$$

where, a is a scaler parameter. This activation function is named differently depending on values of a. For 0.01 and 0 values the function is called Leaky ReLU and ReLU respectively.

Softmax function is defined as:

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \quad \text{for} \quad i = 1, 2, \ldots, J \tag{6}$$

where, $\vec{x}$ is a input vector to the softmax functon and J is number of input layer.

IV. Dataset and Features

We have used the CICID [19] dataset that includes various attacks [15]. Among the attacks, most notable are Brute-force attack, DoS attack, Web attack, Infiltration attack, Botnet attack, DDoS attack and Port-Scanning. There are many tools for conducting Brute-force attacks on password cracking such as Hydra, Medusa, Ncrack, Metasploit modules and Nmap NSE scripts. Also, there are some tools such as hashcat and hashpump for password hash cracking. Meanwhile, Patator is one of the most comprehensive multi-threaded tools which is written in Python and seems to be more reliable and flexible because it can save every response in a separate log file for later review and support more than 30 different methods such as FTP, SSH, Telnet, and SMTP. In this scenario the attacker is a Kali Linux and the victim is an Ubuntu 16.04 system as the web server and executes the FTP-Patator and the SSH-Patator. Among the available tools of DoS attack such as LOIC, HOIC,Hulk, GoldenEye, Slowloris, and Slowhttptest, we used the four last ones. Slowloris and Slowhttptest let a single machine keeping connections open with minimal bandwidth that consumes the web server resources and take it down very fast. In this scenario the attacker is a Kali Linux and the victim is an Ubuntu 16.04 system with Apache web server. For another kind of DoS attack, Heartleech is one of the most famous tools to exploit Heartbleed which can scan a system to find the vulnerabilities. In this scenario we compiled and installed OpenSSL version 1.0.1f which is a vulnerable version of OpenSSL on Ubuntu 12.04 and then by using Heartleech we retrieved the memory dumps of web server's process on the server. In order to implement Web attack scenario, this dataset used Damn Vulnerable Web App (DVWA), which is a vulnerable PHP/MySQL web application. In order to automate the attacks in XSS and Brute-force section we developed an automated code with Selenium framework. The attacker is a Kali Linux and the victim is an Ubuntu 16.04 system as a web server. For infiltration attack scenario, this dataset used the Metasploit as the most common security issues and vulnerability verifier. After victim download the infected file in first level, from Dropbox for windows machine or from infected USB falsh memory for macintosh machine, the attacker execute the Nmap and portscan for the second level on the entire Victim-Network. The attacker is a Kali Linux and the victims are Windows, Ubuntu and Macintosh systems in the Victim-Network. For Botnet attack, there are different tools such as Grum, Windigo, Storm and Ares. In this dataset, we used Ares which is a Python based Botnet, that can provide remote shell, file upload/download, capturing screenshots and key logging. The attacker is a Kali Linux and

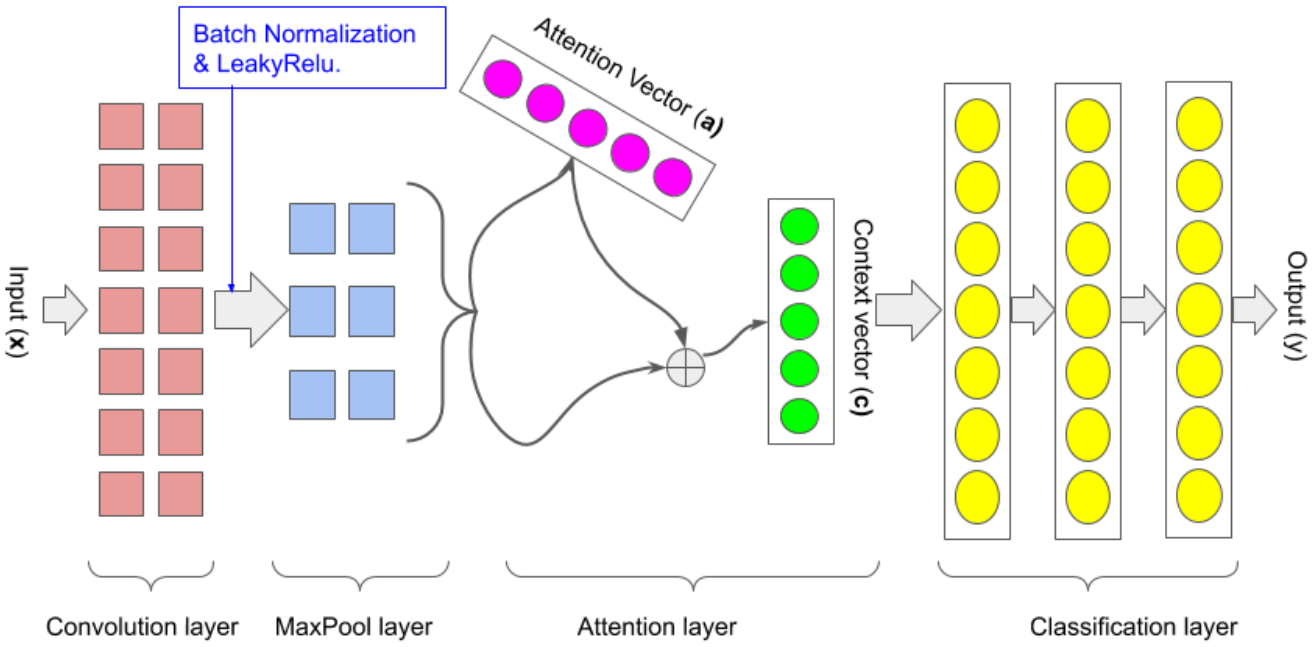

Fig. 1. Our proposed neural network. Each Layer is Keras implemented layer except "Attention" Layer which we implemented in Keras.

the victims are five different Windows OS, namely Vista, 7, 8.1, 10 (32-bit) and 10 (64-bit). DDoS attack and PortScan, among the available DDoS attack tools such as High Orbit Ion Canon (HOIC), Low Orbit Ion Canon (LOIC), DDoSIM, this dataset used the LOIC which is sending the UDP, TCP, or HTTP requests to the victim. For extracting the network traffic features, this dataset used the CICFlowMeter [15], which is a flow based feature extractor and can extract 80 features from a pcap file. The flow label in this application includes SourceIP, SourcePort, DestinationIP, DestinationPort and Protocol. We are inlcuding two charts from the dataset analyzing existing attak data.

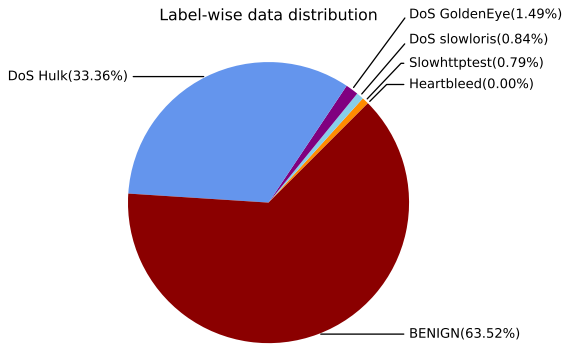

Fig. 2. Label-wise percentage of instances in CICID dataset and their frequency are Benign(440031), DoS Hulk(231073), DoS GoldenEye(10293), DoS slowloris(5796), DoS Slowhttptest(5499), and Heartbleed(11).

We are inlcuding another data plot from the dataset analyzing existing attack data.

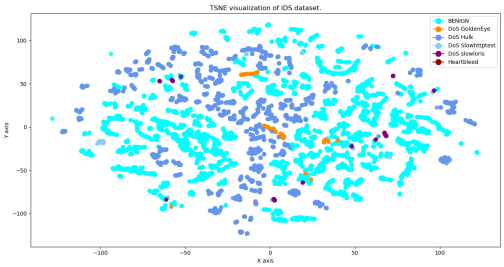

Fig. 3. Dataset attack data in 2d plot

V. Experimental Result and Analysis

We have performed experiment to test our model's rigidity with different settings. For example, following subsection discuss about our model performance and sensitivity to different learning rate. In this experiment, we used CICID Dataset [19] after some data pre-processing. Pre-processing involved missing attribute value handling by replacing with zeros and finally transformed dataset within range of [0, 1] for faster processing and/or training our neural network with min-max normalization.

$$x_{std} = \frac{x - min(x)}{max(x) - min(x)} \tag{7}$$

$$x_{new} = x_{std} * (max - min) + min \tag{8}$$

where min and max are the new feature range. In this experiment, we sampled 10k data tuples randomly from CICID

dataset and split into 63 percent training and 37 percent validation set, trained model with batch size 124 and 100 epoch. For optimization algorithm, we employed stochastic gradient descent algorithm using Adagrad [20] optimizer. We have seen our model generate best performance with learning rate 0.01 and achieve overall 98.72 and 98.64 accuracy on training and validation set respectively.

A. Performance Metrics

To measure model performance, we used some evaluation metrics whose definitions and formulas are given below.

Definition. *True positive(tp) is the number of actual positive instances that are predicted as positive, False negative(fn) is the number of actual positive instances that are predicted as negative, False positive(fp) is the number of actual negative instances that are predicted as positive, True negative(tn) is the number of actual negative instances that are predicted as negative, N is the total number of instances, and C is the number of different classes.*

Having defined above definition, we can now define accuracy(AC), precision(PR), recall(RC), f-measure, micro and macro average measures.

Definition. *Accuracy(AC) is the fraction of correctly classified instances.*

$$AC = \frac{\sum_c^C tp_c}{N}, \textit{in generall} \tag{9}$$

$$AC = \frac{tp + tn}{N}, \textit{for binary case} \tag{10}$$

Definition. *Error(ER) is the fraction of miss-classified instances.*

$$ER = \frac{\sum_c^C fp_c}{N} = 1 - AC \tag{11}$$

Definition. *Precision (PR) or Positive Predictive value(PPV) is the ratio of correctly classified instances(tp), in front of all the positive predicted instances(tp+fp).*

$$PR = \frac{tp}{tp + fp} \tag{12}$$

Definition. *Recall (RC) OR Sensitivity(OR True positive rate) is the ratio of correctly classified instances (tp), in front of all true positive instances(tp+fn).*

$$RC = \frac{tp}{tp + fn} \tag{13}$$

Definition. *F-measure(F1) which takes account both precision(PR) and recall(RC), is the single measure of combination of two and defined as the harmonic mean of two measures precision and recall.*

$$F1 = \frac{2}{\frac{1}{PR} + \frac{1}{RC}} \tag{14}$$

The macro-averaged results can be computed as indicated by: $L = \{\lambda_j : j = 1 \dots q\}$ is the set of all labels. Consider a binary evaluation measure B(tp, tn, fp, fn) that is calculated based on the number of true positives (tp), true negatives (tn),false positives (fp) and false negatives (fn). Let $tp_\lambda, fp_\lambda, tn_\lambda$ and fn_λ be the number of true positives, false positives, true negatives and false negatives after binary evaluation for a label λ.

$$B_{macro} = \frac{1}{q} \sum_{\lambda=1}^{q} B(tp_\lambda, tn_\lambda, fp_\lambda, fn_\lambda) \tag{15}$$

A micro-averaged can be computed as follow:

$$B_{micro} = B\Big(\sum_{\lambda=1}^{q} tp_\lambda, \sum_{\lambda=1}^{q} tn_\lambda, \sum_{\lambda=1}^{q} fp_\lambda, \sum_{\lambda=1}^{q} fn_\lambda \Big) \tag{16}$$

For loss function, we have used categorical cross-entropy to compute loss at the end of each epoch because our target label were one-hot encoded. The equation for computation of loss is given below:

$$\hat{y}_{norm} = \frac{\hat{y}}{\sum_{x \in \hat{y}} x} \tag{17}$$

$$\mathcal{L}(\hat{y}_{norm}, y_{target}) = -\sum_x y_{target}(x) * \log(\hat{y}_{norm}(x)) \tag{18}$$

where, $\hat{y}$ is the output vector of last layer, $\hat{y}_{norm}$ is the normalized vector of $\hat{y}$, y_{target} is the true class label vector, x is the index of both equal dimensional vector y_{target} and $\hat{y}_{norm}$, and finally $\mathcal{L}$ is the scalar loss value which is minimized by the optimizer.

B. Model Accuracy Evaluation

In fig. 4, we observe that over epoch, training accuracy increases for the noted learning rate [0.7, 0.05, 0.03, 0.02, 0.01, 0.001, 0.0001]. However, all the depicted learning rate(lr) settings, model do not perform equally. For instance, model achieve accuracy slightly above 0.6 with lr(=0.7) while achieve accuracy slightly below 0.9 with lr(=0.0001), for the others learning rate model achieves more or less same training accuracy at the end of 100 epoch. One important thing to notice here is, model performs comparatively better in accuracy when learning rate set to 0.01 and degrades performance if set either below or above this learning rate which confirms our earlier claim in this section. Validation curves in fig. 5 follows more or less similar trends. Notice that, within 10 epoch model learn around 98% about the distribution of training and validation dataset. In fig. 6, we notice that validation accuracy curve follows training accuracy curve and on average validation accuracy is always below the training accuracy, though we witness some overlapping and spike for both curves at the initial epoch, namely between [0, 21], due to random initialization of weights and biases of the model in all layers. Overall, model achieves quite significant level of accuracy with limited model capacity(few layers and parameters).

In Fig. 7, for learning rate [0.01, 0.02, 0.03] the model do quite well to learn or distinguish target classes from feature set or data distribution, but performance degrades or slowly learn for others learning rate. For those learning rate the model achieves overall training loss of around 0.00362 at the end of

epoch 100. Notice that for lr(=0.1), the model do not learn at all, loss curves almost flat for all the epoch. Likewise, almost all Loss curves in fig. 8 follows similar pattern for validation set also. To observe whether the model is over-fitting or under-fitting, we plotted training loss and validation loss together over epoch in fig. 9. We witnessed that our model do not over-fit or under-fit for the future dataset significantly, which confirms effectiveness of batch normalization to reduce under-fitting or over-fitting. Therefore, our model do in-fact generalize towards ground truth very efficiently for the future unknown attack flows.

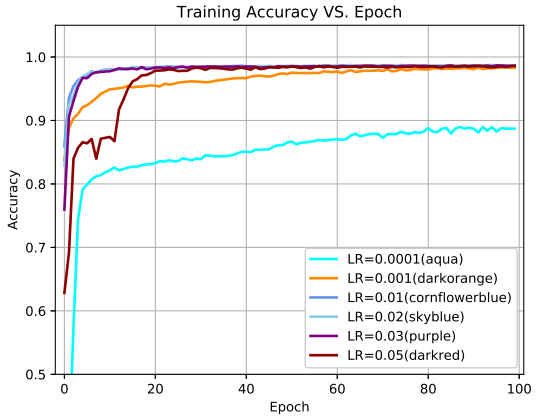

Fig. 4. Training accuracy with varying number of learning rate(LR) of optimizer. Y-axis correspond to Accuracy [0,1] while X-axis correspond to Epoch number. One Epoch means, training of the whole training dataset once. The higher the accuracy with lower number of epoch the better is the model's performance.

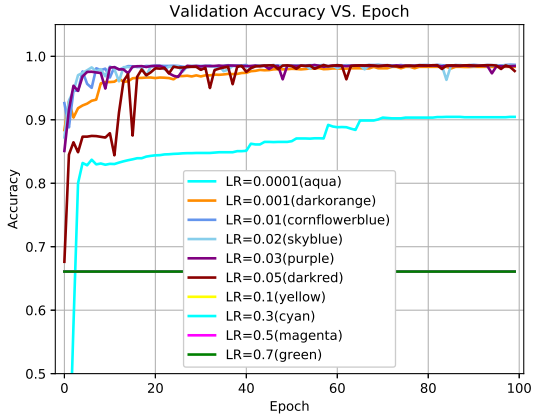

Fig. 5. Validation accuracy with varying number of learning rate(LR) of optimizer. X-axis correspond to accuracy of validation dataset measured at the end of every training epoch. Notes, validation dataset is independent of training dataset and is not used for training the model. The accuracy of validation dataset measures how well model will perform for the unseen data.

C. Performance Evaluation with CM and ROC Curves

To further investigate and performance evaluation, we plotted confusion matrix and receiver operating curves (ROC)

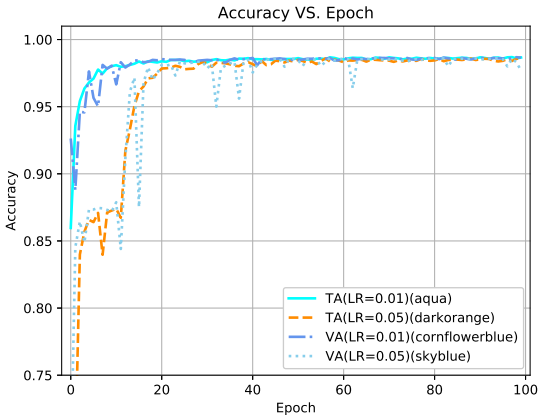

Fig. 6. Training and validation accuracy plotted together to see whether model is over-fitting or not. TA for training accuracy and VA for validation accuracy.

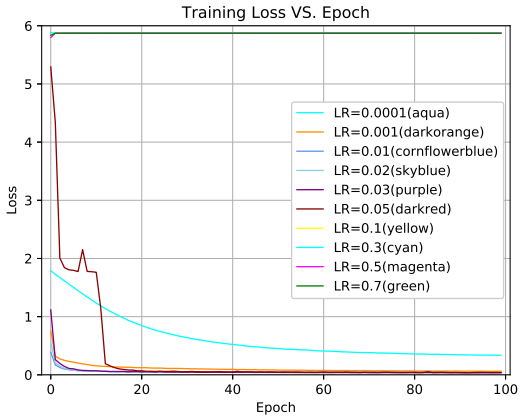

Fig. 7. Training Loss with varying learning rate(LR). X-axis correspond to Epoch number while Y-axis correspond to cross-entropy loss. The lower the cross-entropy loss with lower number of epoch the better for the model's performance. Cross-entropy measure the similarity between two distribution, higher similarity between two distribution correspond to lower cross-entropy quantity and it becomes zero when two distribution are identical.

with data evaluated at the end of final training epoch using validation dataset. ROC curves are very useful technique or tool for performance evaluation of model having trained with unbalanced dataset(unbalanced w.r.t. target or attack classes), for instance, detecting cancer which is scarce in nature. In our case, attack flows is very rare compared to normal or benign flows. So, ROC evaluation is perfect choice from data imbalance standpoint. From fig. in 10, we observe that our model achieved area of 0.99, 1.0, 0.99, 0.99, 1.0, and 0.98, which indicates detection rates for DOS Slowloris, DOS slowhttptest, DOS goldenEye, DOS Hulk attack flows, Heartbleed, and Benign flows respectively. Confusion matrix plotted in fig. 11 not only confirms this findings but also tells us that we are still mis-classifying 1% of DoS slowloris, DoS Slowhttptest, DoS Hulk, 3% DoS GoldenEye, and 2% Benign that indicates false alarm. This is, mis-identification between attack types,

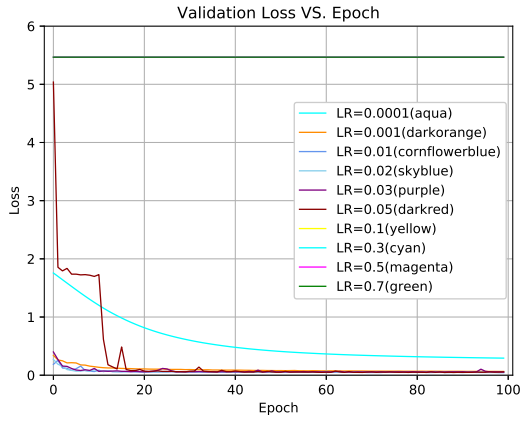

Fig. 8. Validation Loss similar to the training loss except the quantity is measured using validation dataset. It indicates how well a model will perform for the unseen future data.

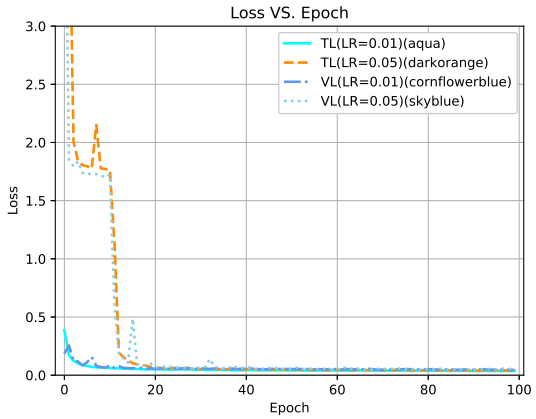

Fig. 9. Training and Validation Loss plotted together to see whether model is over-fitting or under-fitting. TL for training loss and VL for validation loss.

attack type as benign, and false alarm(identification of benign as attack type), partially attributed to the noise in the data, overlapping distribution, and data imbalance problems.

D. Performance Comparison with LSTM

To compare performance of our base model(A-CNN), with other RNN based model we have chosen to replace first and second convolution layer at a time with two consecutive 32 unit and 78 unit LSTM layer into our base model resulting two model A-LSTM-32 unit and A-LSTM-78 unit respectively and kept all other layer's parameters exactly the same. In fact, model capacity in terms of trainable parameters of A-LSTM-32 unit and A-LSTM-78 unit were 56,903 and 127,099 while for base model A-CNN it was 130,879 trainable parameters. In this experiment, for all three model, we selected best run from several run performed with the same settings: random 10k dataset and adaptive gradient(Adagrad) optimizer with learning rate 0.01. Even though parameters and/or model capacity of A-CNN and A-LSTM-78 unit were roughly the

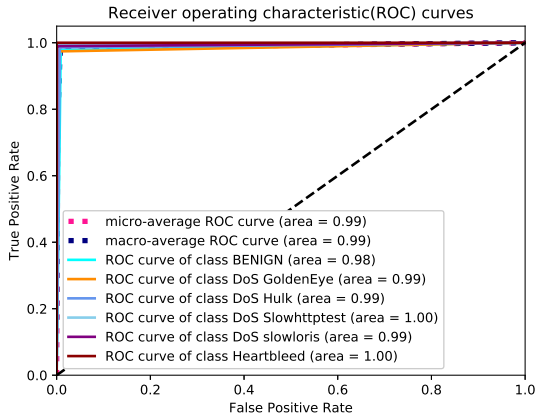

Fig. 10. Receiver Operating Characteristic Curves. The higher area in the unit square means the better for model's performance.

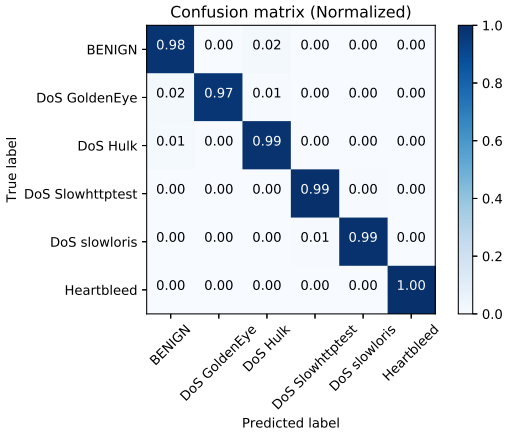

Fig. 11. Normalized Confusion Matrix. Predicted class labels are annotated horizontally and true class labels are annotated vertically. The elements in the matrix indicate percentage of instances of that corresponding horizontal and vertical class labels. The higher percentage in the diagonal direction means the higher accuracy of the model in predicting class labels.

same, there is a clear difference in the performance of the two model. Our A-CNN model's performance is evidently ahead of the two RNN model A-LSTM-32 unit and A-LSTM-78 unit in terms of performance matrices as shown in fig. 12, 13, 14, and 15. Notice that, the performance of the A-LSTM-78 unit is not significantly increased compared to A-LSTM-32 unit even though model capacity of the first is increased roughly twice than that of first.

E. Cross Validation with StratifiedKFold

We have performed cross-validation to verify how our model performs with whole dataset in consideration. Previous experiment involved two fixed independent set training and validation set for training and testing respectfully. Now, we are going to perform stratified 10-folds cross-validation which is similar to K-fold cross validation with one difference. K-fold do not consider class or group distribution while

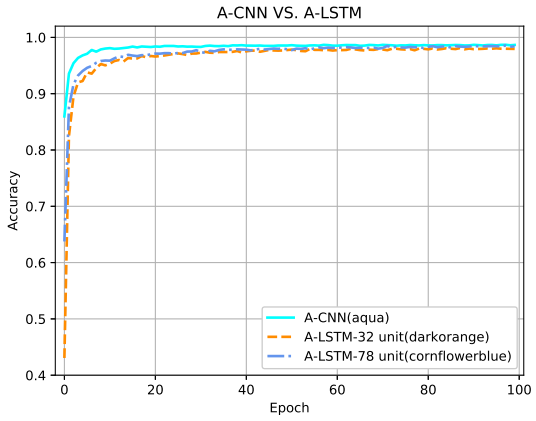

Fig. 12. Training Accuracy Comparison, (a) Our model A-CNN (deep blue), (b) A-LSTM-32 unit (light green), (c) A-LSTM-78 unit (light blue). For all, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

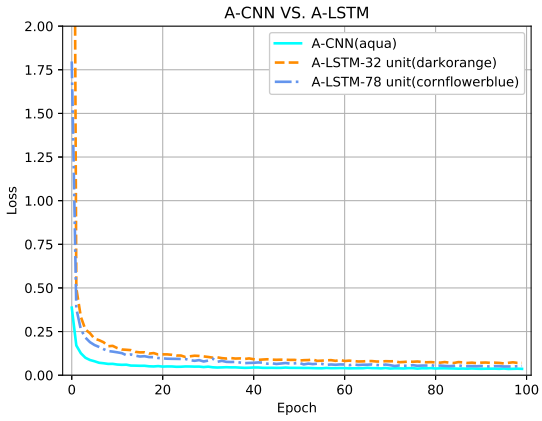

Fig. 14. Training Loss Comparison, (a) Our model A-CNN (deep blue), (b) A-LSTM-32 unit (light green), (c) A-LSTM-78 unit (light blue). For all, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

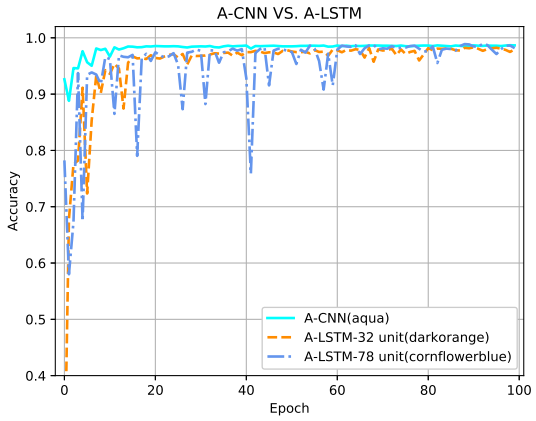

Fig. 13. Validation Accuracy Comparison, (a) Our model A-CNN (deep blue), (b) A-LSTM-32 unit (light green), (c) A-LSTM-78 unit (light blue). For all, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

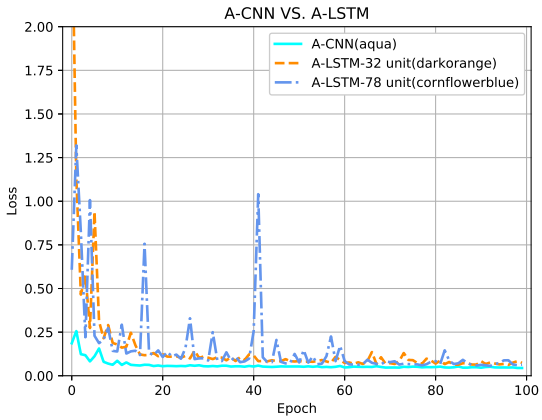

Fig. 15. Validation Loss Comparison, (a) Our model A-CNN (deep blue), (b) A-LSTM-32 unit (light green), (c) A-LSTM-78 unit (light blue). For all, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

producing k number of folds but StratifiedKFold maintains equal percentage of samples for each class in each of the k-fold. Each of 10 iteration, training set contains 9 folds and test set contains 1 fold, with 9 folds we train our model and peform evaluation with remaining 1 fold. In words, we are testing all the dataset each time testing a unique fold against all the other 9 folds. Finally, taking the average of all 10 round scores as the final score. Our model achieved accuracy 98.55%(+/-0.25%), Precision 98.57%(+/-0.26%), Recall 98.53%(+/-0.26%), and F-measure 98.55%(+/-0.26%).

F. Boosting Performance using Synthetic Dataset

As we mentioned in earlier section, imbalance dataset is a major obstacle in achieving state of the art performance in detecting network intrusion. For examples, DOS Slowhttptest and DOS slowloris significantly lower than DOS GoldenEye and BENIGN type and Heartbleed attack contains only 11 instances which is way much lower than other attack type. Henceforth, we decided to produce synthetic dataset using Conditional generative adversarial network(CGAN) [21] from a family of GAN networks. We implemented CGAN network using Keras Dense and BatchNorm layer with LeakyRelu, sigmoid and tanh activation function and trained both Generator and Discriminator network using real dataset(CICID) to optimize KL-divergence as objective function before generation of synthetic dataset. An overview of synthetic data generation network is shown in fig. 16. We have generated a total of 16386 instances which consist of 50% benign and 50% attack instances with equal proportion of each attack type. Using this generated dataset, we trained our model(A-CNN) and tested with 63 percent train set and 33 percent validation set respectively from randomly selected 10k dataset. We performed experiment several times, noticed that every time the performance of our model enhanced by about 1.5% using synthetic dataset. Performances of our model under

TABLE I
CROSS-VALIDATION EXPERIMENTAL DATA

Round	Accuracy	Precision	Recall	F-measure
1	98.31	98.31	98.21	98.26
2	98.80	98.90	98.80	98.85
3	98.20	98.20	98.20	98.20
4	99.00	99.00	99.00	99.00
5	98.40	98.40	98.40	98.40
6	98.80	98.80	98.80	98.80
7	98.50	98.50	98.50	98.50
8	98.60	98.70	98.50	98.60
9	98.60	98.60	98.60	98.60
10	98.30	98.30	98.30	98.30
Mean(μ)	**98.55**	**98.57**	**98.53**	**98.55**
SD(σ)	**0.25**	**0.26**	**0.26**	**0.26**

original and synthetic dataset are given in fig. 17, 18, 19, and 20 in terms of accuracy and cross-entropy loss of both training and validation dataset. One more important thing is that we have eliminated as many false alarm as possible as can be seen from confusion matrix in fig. 21.

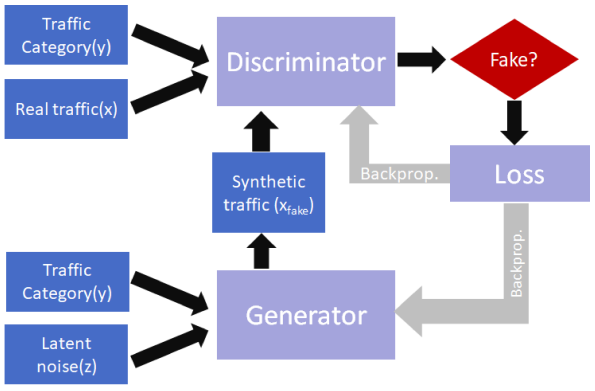

Fig. 16. Overview of synthetic data generation network, Conditional Generative Adversarial Network (CGAN).

G. Summary of Results

In summary, experimental results show that our model(A-CNN) achieves best performance using Adagrad with a learning rate of 0.01. In addition, Attention mechanism in CNN (A-CNN) performs better than A-LSTM and A-CNN achieves state of the art performance in terms of overall accuracy (98.55%), precision(98.57%), recall(98.53%), and f-measure(98.55%). Although overall performance of our model (A-CNN) is quite impressive, still mis-classification of Benign and Dos attack prevent us achieving nearly 100% accuracy. This might be due to the noise and unbalanced dataset. Hence, we tried to reduce noise and data imbalance problems by adopting CGAN network for synthetic data generation which has proven to be effective as shown by the performance boost

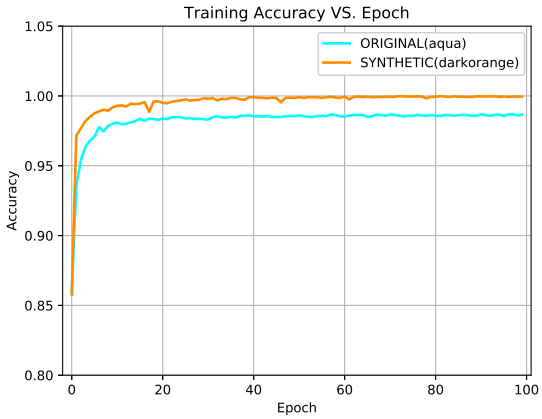

Fig. 17. A-CNN model's Training Accuracy Comparison, Using (a) Original Dataset(aqua), (b) Synthetic Dataset(darkorange). For two, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

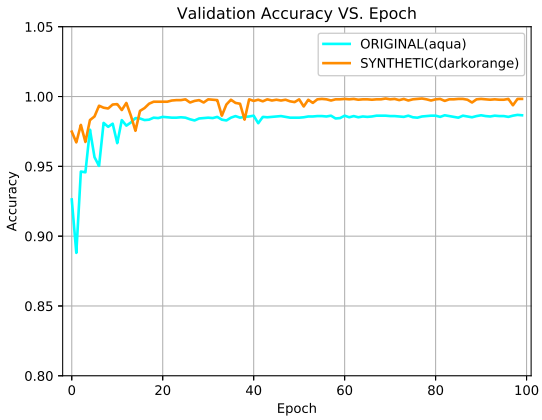

Fig. 18. A-CNN model's Validation Accuracy Comparison, Using (a) Original Dataset(aqua), (b) Synthetic Dataset(darkorange). For two, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

of A-CNN network by about 1.5% under synthetic balanced dataset, resulting in overall training and validation accuracy of 99.96% and 99.82% , respectively.

VI. CONCLUSION AND FUTURE WORKS

To cope with new mutant and emerging threat, security researchers in recent decade sought numerous method for effective IDS in network security and are still striving to identify improved methods and approaches. This work is one such approach that convincingly illustrates the effectiveness of attention based convolutional method(A-CNN) in intrusion detection. We demonstrated that A-CNN can detect and classify almost all type of attacks(DOS Slowloris, DOS slowhttptest, DOS goldeneye, DOS hulk etc.) in CICID dataset with highest accuracy rate better than LSTM recurrent neural network. Based on our rigorous experimental analysis, we suggest to employ A-CNN based network intrusion detection software in real traffic analysis and attack detection system. Currently,

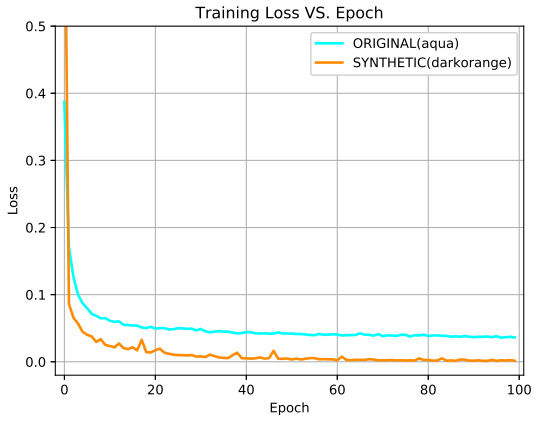

Fig. 19. A-CNN model's Training Loss Comparison, Using (a) Original Dataset(aqua), (b) Synthetic Dataset(darkorange). For two, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

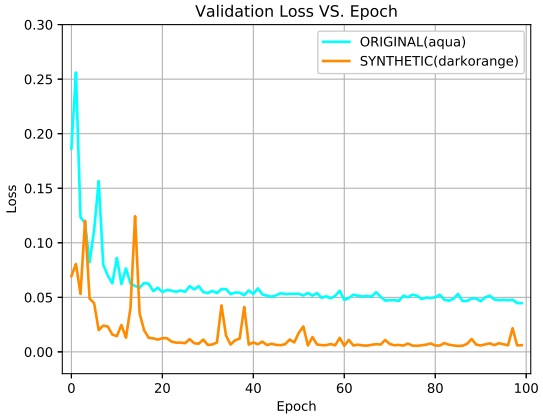

Fig. 20. A-CNN model's Validation Loss Comparison, Using (a) Original Dataset(aqua), (b) Synthetic Dataset(darkorange). For two, Adaptive gradient(Adagrad) optimizer with learning rate 0.01, were used.

we focused solely on CICID dataset and their effectiveness in detecting and classifying intrusion attack.

Although the dataset covers varieties of intrusion attacks, it lacks mixture of multiple simulated attack flows along with novel type of attacks which might yet be unknown. In future works, we plan to explore new type of mutant attack generation from the known attacks in CICID dataset and test those attacks against various known attack patterns and model types including our model.

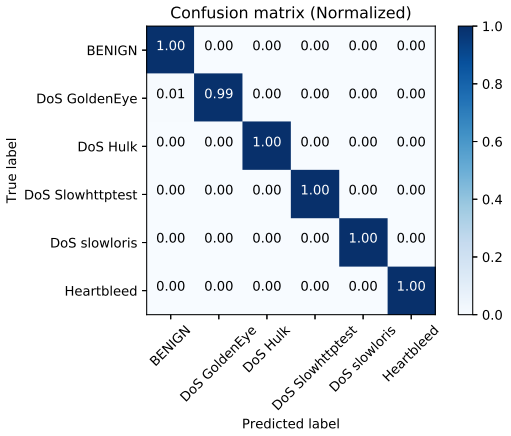

Fig. 21. A-CNN model's confusion matrix under synthetic validation dataset. The elements in the matrix indicate percentage of instances of that corresponding horizontal and vertical class labels. The higher percentage in the diagonal direction means the higher accuracy of the model in predicting class labels.

REFERENCES

[1] "Anatomy of a cyber attack the lifecycle of a security breach," 2017, accessed on 2019-04. [Online]. Available: http://www.oracle.com/us/technologies/linux/anatomy-of-cyber-attacks-wp-4124673.pdf

[2] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computer Survey*, vol. 46, no. 4, pp. 55:1–55:29, mar 2014.

[3] G. Bruneau, "The history and evolution of intrusion detection," https://www.sans.org/reading-room/whitepapers/detection/history-evolution-intrusion-detection-344, Oct 2001, accessed on 2018-06. [Online]. Available: https://www.sans.org/reading-room/whitepapers/detection/history-evolution-intrusion-detection-344

[4] S. K. Jonnalagadda, Machilipatnam, and R. K. Reddy, "A literature survey and comprehensive study of intrusion detection," *International Journal of Computer Applications*, vol. 81, no. 16, 2013.

[5] B. Subba, S. Biswas, and S. Karmakar, "A neural network based system for intrusion detection and attack classification," in *National Conference on Communication (NCC)*, March 2016, pp. 1–6.

[6] A. R. Onik, N. F. Haq, and W. Mustahin, "Cross-breed type bayesian network based intrusion detection system (cbnids)," in *2015 18th International Conference on Computer and Information Technology (ICCIT)*, Dec 2015, pp. 407–412.

[7] M. A. Bode, S. A. Oluwadare, B. K. Alese, and A. F. Thompson, "Risk analysis in cyber situation awareness using bayesian approach," in *2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*, June 2015, pp. 1–12.

[8] T. Xia, G. Qu, S. Hariri, and M. Yousif, "An efficient network intrusion detection method based on information theory and genetic algorithm," in *PCCC 2005. 24th IEEE International Performance, Computing, and Communications Conference, 2005.*, April 2005, pp. 11–17.

[9] B. Senthilnayaki, K. Venkatalakshmi, and A. Kannan, "Intrusion detection using optimal genetic feature selection and svm based classifier," in *2015 3rd International Conference on Signal Processing, Communication and Networking (ICSCN)*, March 2015, pp. 1–4.

[10] J. Kim, J. Kim, H. L. T. Thu, and H. Kim, "Long short term memory recurrent neural network classifier for intrusion detection," in *2016 International Conference on Platform Technology and Service (PlatCon)*, Feb 2016, pp. 1–5.

[11] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Sep. 2017, pp. 1222–1228.

[12] Z. Li, Z. Qin, K. Huang, X. Yang, and S. Ye, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*. Springer International Publishing, 2017, pp. 858–866.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Computer Vision and Pattern Recognition (CVPR)*, 2015. [Online]. Available: http://arxiv.org/abs/1409.4842

[15] I. Sharafaldin, A. Habibi Lashkari, and A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," 01 2018, pp. 108–116.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, 2015.

[17] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, 2015, pp. 2048–2057.

[18] M. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *CoRR*, vol. abs/1508.04025, 2015. [Online]. Available: http://arxiv.org/abs/1508.04025

[19] "Cicid database," accessed on 2019-04. [Online]. Available: https://iscxdownloads.cs.unb.ca/iscxdownloads/CIC-IDS-2017/#CI-IDS-2017

[20] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[21] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *CoRR*, vol. abs/1411.1784, 2014. [Online]. Available: http://arxiv.org/abs/1411.1784