Joined Bi-Model RNN with Spatial Attention and GAN based IoT bot net Attacks Detection

Senthil S (jaysen1984research@gmail.com)
A.R. College of Engineering and Techn

N Muthukumaran
FX Engineering College

Research Article

Keywords: JBiRSA, IoT botnet attack detection, GAN

Posted Date: June 10th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1534435/v1

Title Page

Tile: Joined Bi-Model RNN with Spatial Attention and GAN based IoT bot net Attacks Detection

Author Information:

Senthil.S, AR College of Engineering and Technology, Tamilnadu, India

N.Muthukumaran, Professor, FX Engineering College, Tamilnadu, India

Corresponding Author:

Senthil.S, AR College of Engineering and Technology, Tamilnadu, India

Email: jaysen1984research@gmail.com

Joined Bi-Model RNN with Spatial Attention and GAN based IoT bot net Attacks Detection

Senthil.S, AR College of Engineering and Technology, Tamilnadu, India
N.Muthukumaran, Professor, FX Engineering College, Tamilnadu, India

Abstract

Currently, in many purposes machine and deep learning methods were utilized to identify bot net threats in IoT networks. On the other hand, highly imbalanced network traffic information in the training set frequently humiliates the classification. A novel method is proposed called Joined Bi-Model RNN with Spatial Attention (JBiRSA). In this work, feature generation and classification is the two major processes. For feature generation process, Generative Adversarial Network (GAN) is applied. GAN consist of two neural networks, namely Generator and Discriminator. In GAN, loss is also calculated to enhance accuracy In this model GAN with Traffic Encoder loass is proposed along with an effective generator especially suited for network traffic. Two data sets are used, namely N-BaIoT and IoT-23. N-BaIoT data set is formed by adding bot net attacks such as Bashlite and Mirai. IoT-23 data set is formed with 20 malware confines from various IoT devices and 3 precincts for benign anomalies. JBiRSA with GAN has proven to be efficient and has the potential to differentiate between benign and malicious traffic data in IoT attacks. JBiRSA with GAN provides an overall accuracy of 98.75%.

Keywords: JBiRSA, IoT botnet attack detection, GAN

1 Introduction

The most cited top ten technological enablers, are namely Internet of Things (IoT), Cloud Computing, Robotics, Block chain, Big Data, Fog and Mobile Computing, Human-Computer Interaction, Artificial Intelligence and Internet [13]. The fourth industrial revolution 4.0 is comprised with Industrial IoT (IIoT) and with Internet of Things (IoT) [1]. This revolution has shown that IoT smart devices will widen into additional 20 times than the previous Information Technology (IT) position in 2023 [15]. Nowadays, computers with Internet have turn out to be a ubiquitous component of day to day lives. On the other hand, Internet with computer systems has launched many new purposes with varieties of security threats [5]. The general usage of IoT in different areas has been increased, namely government, physical security, vehicles, utilities and health care. Nowadays, IoT devices connected to the network has been increased a lot. Hence, IoT devices are susceptible to a variety of bot net attacks, namely Operating System (OS) finger printing, Distributed DoS (DDoS), network disruption, key logging, flooding, data exfiltration, Denial of Service (DoS) and service scanning [4]. Hence, there is a great challenge to develop and provide the most favorable security models for every device as shown in the Figure 1. Elegant devices are easily hacked recently, namely fitness trackers, well-designed refrigerators, smart bicycles, elegant watches, smart Mobiles, fashionable fire alarms, stylish door locks and medical sensors.

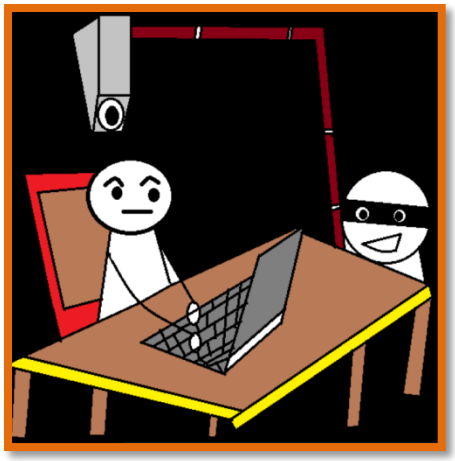

Figure 1 Representation of IoT attacks

An efficient deployment of IoT services expects a proficient Intrusion Detection System (IDS). While transmitting data in IoT devices, IDS guarantees safe, consistent and sound detecting anomalous activities [27]. A comparative study of IDSs is also presented [29] to evaluate anomaly depend IDSs on deep learning methods [29]. In recent times, machine learning also has become a dominant tool to devise bot net detection attacks. Using the current improvements in hardware capacites of computer systems, deep learning also has happened to be more well-known among the current techniques. Deep Neural Networks (DNNs) have also been effectively utilized for resolving many problems with far above the ground accuracies [6]. The major problem in bot net attack detection is unbalanced data sets. To overcome this problem, Synthetic Minority Oversampling Technique (SMOTE) and Generative Adversarial Network (GAN) has been used. These two methods are an efficient approach for the creation of classifiers from imbalanced data sets. GAN is demonstrated to be extremely valuable in many research works [5, 8]. GAN generates realistic samples by combining different various competitively learning types of AI models. On comparing with SMOTE, GAN provides efficient outcomes in balancing data sets. In recent times, GANs have appeared as an efficient unsupervised anomaly detection which is advance in computer vision and time series purposes [27].

However, GAN has contacts to all of the information points. It can also reason communication above your head and enlarge the susceptibility of the IoT to other attacks on central units. Here, a novel method is proposed called Joined Bi-Model RNN with Spatial Attention (JBiRSA). GAN consist of two steps, namely generator and discriminator. First, generator learns to generate possible data. This generated instance provides negative training samples for discriminator. Second, discriminator learns to classify generator's forged data from actual data. It also castigates generator for generating improbable outcomes. Generator chose in random figures and results an image. This generated result is provided as input to discriminator next to a flow of images full from definite and ground truth data set. Both actual and fraud images are considered as input to discriminator. The remainder of this paper is structured as it follows. Related works on the IoT security threats is explained in both machine learning and deep learning techniques are depicted in the Section 2. The Section 3 explains JBiRSA with GAN. The Section 4 describes experimental outcomes and analysis. The Section draws conclusions and recommends probable future research guidelines.

2 Related Works

Due to provision of IoT devices, novel security issues can be difficult [55]. Depending on deep learning and machine learning algorithms, a lot of research works on IoT studies is going on

nowadays. Antonio et al [56] have provided a model to handle security and privacy concerns. This proposed model depends on ARM compliant security framework. It is mainly projected to support the devise and the expansion of secure and privacy-aware IoT-enabled services. Ray et al [31] have provided an efficient survey on IOT oriented architectures. This survey is very much helpful to develop the thoughtful of related tool, knowledge, and method to make easy developer's necessities.

Research Works on IoT devices by applying Machine learning techniques

Jiyeon Kim et al [1] have applied machine learning methods for IoT attack detection. Machine learning techniques are applied in every IoT devices on bot net attacks, namely Bashlite and Mirai. Hence, bot net attacks are inserted in the N-BaIoT data set. IoT devices applied are web cam, security camera, door bell and baby Monitor. Tarun Ganesh Palla and Shahab Tayeb [3] have detected Mirai by applying machine learning algorithm called Artificial Neural Network (ANN). Over fitting problem is avoided by choosing accurate number of hidden layers and neurons. Rizwan et al [6] have proposed an innovative technique called Bot shot. This technique depends on Generative Adversarial Networks (GANs) for creating bot net detectors aware of adversarial evasions. Alauthman et al [8] have proposed a complicated traffic reduction mechanism which is incorporated with a reinforcement learning method. Bot malware and Bot net are the tools used for facilitating malicious cyber actions such as DDoS attacks. Amin et al [11] have proposed an energy consumption models for various procedures to classify ransom ware from non malicious purposes. This technique is a machine learning depended method to identify ransom ware assaults by examining power consumption of Android devices. While considering in terms of recall rate, accuracy rate, precision rate and F-measure, this method provides excellent outcomes on comparing with K-Nearest Neighbors, Support Vector Machine, Neural Networks and Random Forest.

Haddad et al [16] have proposed Advanced RISC Machine (ARM) based IoT applications. LSTM classifier is applied on malware information, which is not utilized in replica training. The proposed model provides 97% average accuracy outcomes. Hodo et al [17] have explained the characteristics features of host based Intrusion Detection System (IDS) and network-based IDS. DDoS/DoS attack detection model are performed by applying ANN with 99.4% accuracy. Bastos et al [20] have recommended a new system to classify Mirai and Bashlite in C&C servers by joining four heuristic techniques. Xiaofeng et al [21] have prepared to collect data from sensors, humans and smart cities. Anomaly detection is carried out with a model depend machine learning techniques. Here, Long Short Term Memory Neural Network (LSTM-NN) and MLP models are also applied for effective outcomes. Furqan Alam et al [22] have identified attacks by using four classification methods, namely Latent Dirichlet Allocation (LDA), Support Vector Machine (SVM), K-nearest neighbor (KNN) and Naïve Bayes (NB). This LDA technique utilized in this method has a faster processing speed.

Research Works on IoT devices by applying Deep learning techniques

In numerous research works, deep learning techniques have proven its cability to recognize anomalies correctly [24]. Segun et al [4] have utilized Deep Recurrent Neural Network (DRNN), Long Short-Term Memory Auto encoder (LAE), and Synthetic Minority Oversampling Technique (SMOTE), to create a memory competent technique. This memory efficient method is called LS-DRNN. This technique yields best outcomes for minority classes using SMOTE method. Abdulghani et al [5] have utilized deep learning to recognize zero-day bot net assaults in real time. This method is trained and assessed on a CTU-13 data set. This method also has multiple neural network designs and hidden layers. Deep learning with Artificial Neural Network (ANN) model

identifies bot nets exactly and proficiently. Nicolas et al [9] have applied Deep Reinforcement Learning (DRL) does not necessitate classifiers to be skilled. Here, the malicious outcome is called a poisoning attack which can be originated by an adversarial evasion attack. Furkan et al [10] have proposed a deep learning based technique for detection of routing attacks for IoT. Here, the Cooja IoT simulator has been applied for creating high-fidelity attack information within IoT networks. This simulator is extremely scalable with high accuracy and precision. Imtiaz Ullah et al [24] have proposed and increased a novel anomaly based intrusion detection model for IoT networks. For multiclass classification model, CNN is used. Here, Transfer learning is employed to implement multiclass pre-trained model. Sriram et al [25] have proposed a network traffic flow depend deep learning bot net recognition method. This technique entirely depends on the reliability and the integrity of the data set provided. Yin et al [26] have shown an integrated deep learning model for anomaly detection in IoT networks. LSTM auto encoder and CNN are used to recognize the anomalies present in the data set. To attain enhanced learning calculations, window-based two steps of preprocessing in data is performed.

Segun et al [2] have used Synthetic Minority Oversampling Technique (SMOTE) for generating minority samples to accomplish class balance effectively. Significantly imbalanced network traffic information is overcome by DL-based bot net attack detection technique. Chuanlong et al [14] have proposed a deep learning framework based on Generative Adversarial Networks (GAN). This method is applied to enhance the performance of the original bot net detection model. It also improves the detection performance by decreasing the false positive rate. Meidan et al [18] have proposed anomaly detection based on Deep Auto Encoder (DAE) for N-BaIoT data set. The Mirai and the Bashlite bot net environments were put up and utilized. Shorman et al [19] have proposed an IoT bot net detection model in four steps, namely data migration, data cleaning, data rescaling and optimizing. N-BaIoT data set is used. Machine learning and deep learning methods are applied in this research. Grey Wolf Optimization with One-Class Support Vector Machine (GWO-OCSVM) is planned with Isolation Forest (IF) and Local Outlier Factor (LOF), to provide faster detection time. Then, DAE is applied to acquire higher accuracy outcomes. Christopher et al [23] have explained the recognition of bot net action contained by customer IoT devices and networks. A detection model is extended depending on a Bidirectional Long Short Term Memory based Recurrent Neural Network (BLSTM-RNN). Word Embedding is applied for text classification and modification of assault packets into tokenized integer format. Shaikh et al [30] have used GAN based models to identify threats to IoT devices from within and outside the network of interest.

2.1 Observations

A lot of learning techniques are utilized for bot net detection. But, efficient outcomes are attained by combining both machine learning and deep learning methods. SMOTE is used to synthesis minority samples, which is used to generate many samples. GAN is used for data synthesis minority samples, which is more efficient compared to SMOTE method.

3 The proposed method

3.1 Generative Adversarial Network (GAN)

GAN model consists of input vector, generator and discriminator. GAN is capable of finding out generative representation of any information all the way through adversarial techniques with admirable presentation. It has been generally useful to different areas, namely medical imaging analysis, medical informatics and bioinformatics [28]. The basic purpose of GAN is to train a generator and a discriminator in an adversarial way. Generator generates novel information occurrences. Discriminator evaluates this novel information occurrence for authenticity.

Discriminator also decides whether each occurrence of information that it reviews belongs to the real training data set or not.

The main advantage of using GAN network is to produce new data occurrences that are similar to training information. Usually GANs attain higher level of practicality by pairing a generator. This generator studies to create the target output, with a discriminator. This discriminator studies to differentiate true information from the output of the generator. The generator seeks to hoodwink the discriminator, and the discriminator tries to keep from being duped. The generator network in a GAN architecture gains knowledge to generate fraud data by integrating response from the discriminator. It discovers how to make the discriminator categorize its outcomes as actual. Figure 2 represents the proposed structural design of the proposed method.

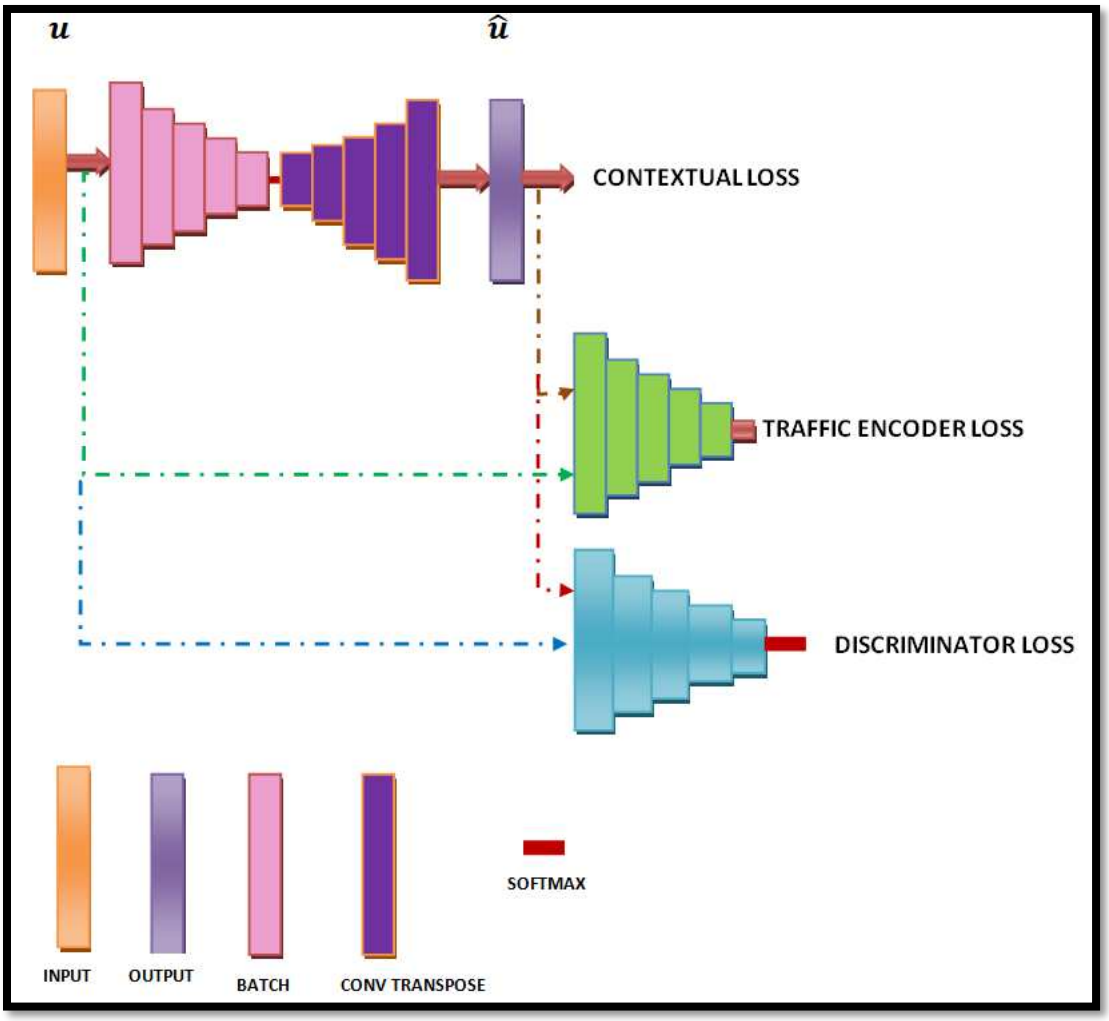

Figure 2 Structural design of the proposed method

The input is passed through the encoder and then to the decoder section. Here, traffic data is provided as the input to the encoder section. Then, the output of encoder section is sent to the decoder section. At this section, the output is send to the softmax layer for classification. The

proposed Generator and Discriminator networks help to identify the network traffic.

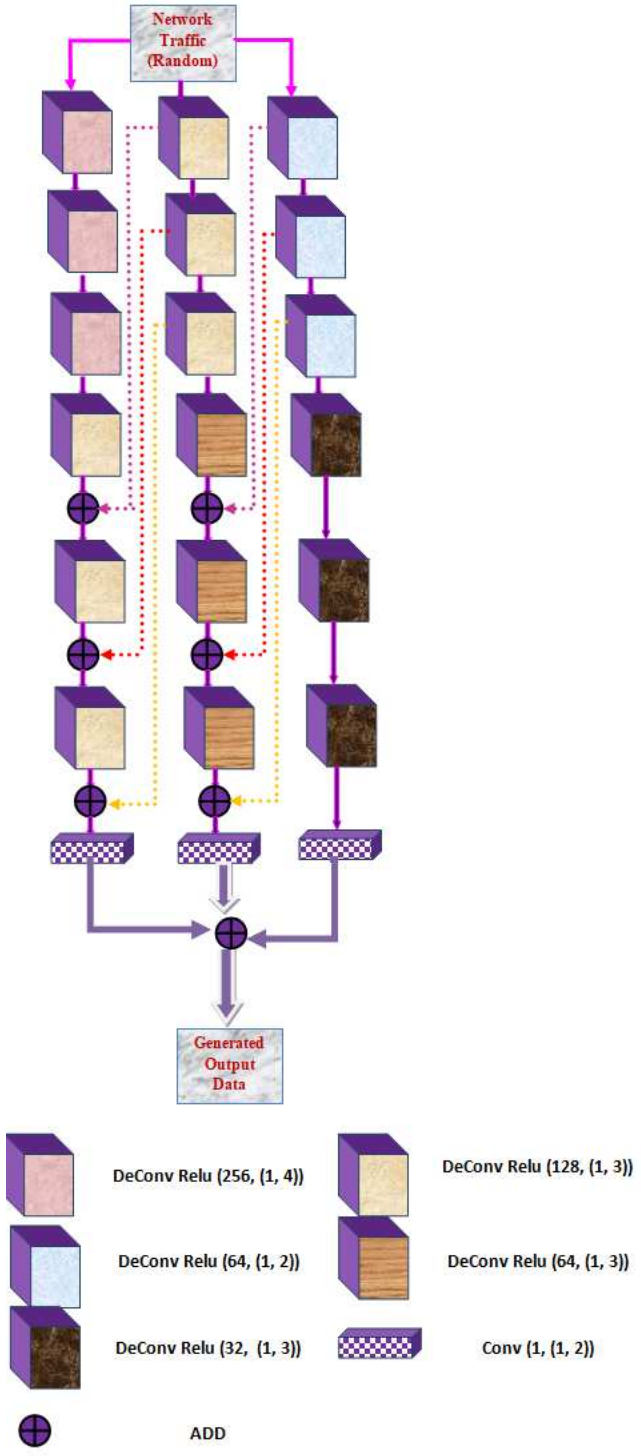

Figure 3 Generator Network

3.1.a Generator Network

Generator network generates probable data, whereas the generated occurrences turn into the negative training samples for discriminator. The proposed GAN architecture is applied for training generative models in deep convolutional neural networks for creating traffics. Generator component creates new outputs from actual data sets. Here, two general types of layers are available in generator model, namely deep convolutional layer with ReLU and convolutional layer. Convolution layer always applied for performing convolution operation.

Rectified Linear Unit (ReLU) is used as the activation function in deep learning methods [32]. It is also the most frequently used significant section in CNNs. This function is incredibly fast to calculate. This also is the default activation function when developing multilayer Perceptron and CNN. ReLU function obtains 0 if it accepts any negative input, but for any positive value x it obtains that value support. So it is represented as shown in Eq. (1) and Eq. (2).

$$F(X) = MAX(0, X) \quad (1)$$

$$F(X) = \begin{cases} o, if\ X \leq 0 \\ X, if\ X > 0 \end{cases} \quad (2)$$

ReLU function is non-linear in the region of 0, but the incline is forever 0 when there are negative inputs. The incline is forever 1 when there are positive inputs. ReLU function is constant, however it is not differentiable since its derivative is 0 for any negative input. The output of ReLU does not have a highest value; hence this assists with Gradient Descent.

The proposed architecture is depicted in the Figure 3. Network Traffic is provided as input to three major deep convolutional neural networks namely, deep convolutional with ReLU function of size (64, (1, 2)), deep convolutional with ReLU function of size (128, (1, 3)) and deep convolutional with ReLU function of size (256, (1, 4)). Then the output of the first deep convolutional with ReLU function of size (64, (1, 2)) is forwarded again to another two deep convolutional with ReLU function of size (64, (1, 2)). The output obtained from the two deep convolutional with ReLU function of size (64, (1, 2)) is forwarded to the other two deep convolutional with ReLU function of size (34, (1, 3)). Subsequently, the output of the two deep convolutional with ReLU function of size (34, (1, 3)) is provided as the input to convolution layer with size ((2, (1, 2)).

The output of deep convolutional with ReLU function of size (128, (1, 3)) is provided as input to the other two deep convolutional with ReLU function of size (128, (1, 3)) as shown in the Figure 3. The output of the other two deep convolutional with ReLU function of size (128, (1, 3)) is given as input to deep convolutional with ReLU function of size (64, (1, 3)). Here the output of deep convolutional with ReLU function of size (64, (1, 3)) is added with the first deep convolutional with ReLU function of size (64, (1, 2)) and provided as input to deep convolutional with ReLU function of size (64, (1, 3)). The output from deep convolutional with ReLU function of size (64, (1, 3)) is added with the output of deep convolutional with ReLU function of size (64, (1, 2)) and provided as input to deep convolutional with ReLU function of size (64, (1, 3)). Subsequently, the output obtained from this function is provided as the input to convolution layer with size ((2, (1, 2)).

The output obtained from deep convolutional with ReLU function of size (256, (1, 4)) is promoted to the other two deep convolutional with ReLU function of size (256, (1, 4)). The output obtained from deep convolutional with ReLU function of size (256, (1, 4)) is given as input to deep convolutional with ReLU function of size (128, (1, 3)). The output from deep convolutional with ReLU function of size (128, (1, 3)) is concatenated with the output of first deep convolutional with ReLU function of size (128, (1, 3)) and given as input to other deep convolutional with ReLU function of size (128, (1, 3)). This before add step is repeated again with another deep

convolutional with ReLU function of size (128, (1, 3)) as shown in Figure 2. Again the same adding process is repeated for the other deep convolutional with ReLU function of size (128, (1, 3)) and provided as input to convolution layer with size ((2, (1, 2)).

At last in generator network as shown in the Figure 3, an average is manipulated for the output of the entire three obtained convolution layer with size ((2, (1, 2)). The output data is generated from adder estimated.

3.1.b Discriminator Network

Discriminator distinguishes generator's forged data from actual data. And also discriminator castigates generator for creating incredible outcomes. Discriminator in a GAN network is basically a classifier. It attempts to differentiate actual data from the data generated by generator. It might employ any network architecture suitable to the type of data it's classifying. Discriminator training data contains data from two sources, namely Real data occurrences as positive samples during training and fraud data occurrences. This fraud data occurrence is formed by generator, whereas it has negative samples during training.

Discriminator is amplified for evaluating the yield of actual sample and generated sample for validity. Real samples will attain high values on the scale originally. At the same time, generated samples will attain lower values on the score. Ultimately, discriminator will have dilemma distinguishing between generated and actual samples. Discriminator depends on constructing a model and potentially an initial loss function.

An objective of dilated convolution is to cover up additional information from the output accumulated with all convolution operation [33]. Dilated Convolutions are a kind of convolution that blows up the kernel by introducing holes among the kernel elements. This method assists increase the area of the input image covered devoid of pooling [34]. Its benefit is more information without increasing the number of kernel parameters. Dilation rate point towards how much the kernel is broadened. Frequently spaces are inserted between kernel components [35].

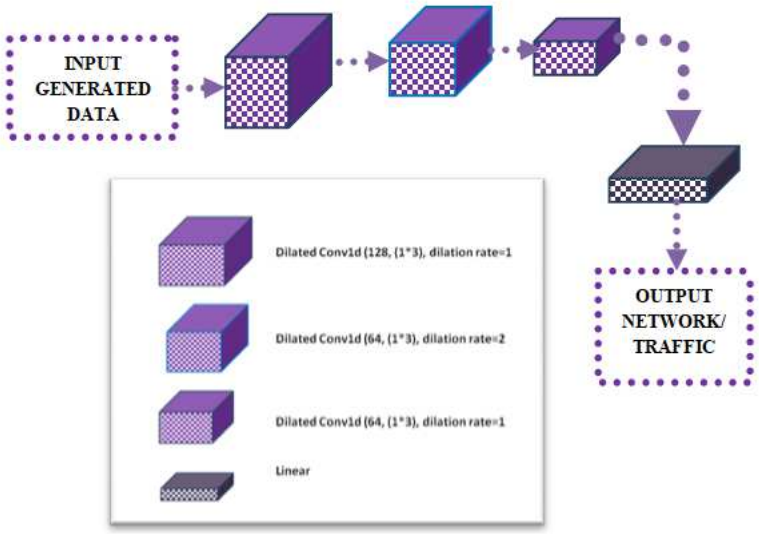

Figure 4 Discriminator Network

Figure 4 represents the proposed discriminator network. The input is forwarded to dilated Convolution with single dimension of size (128, (1*3) with dilation rate value as 1. In dilated one dimension convolution, the output O_1 is estimated as shown in Eq. (3).

$$O_1\,[i] = \sum_{s=1}^{S} f[i + dr\,.S]\delta\,[i] \quad (3)$$

Where f represents the location of i in dilated convolution with a filter δ of size S is defined. Here dr represents the dilated ratio, its value is dr = 1. Dilated convolutions correspond to standard convolutions. An instinctive and direct method to recognize dilated convolutions is that zeros are included between every two adjacent weights in the standard convolutional filters.

The output obtained from dilated Convolution with single dimension of size (128, (1*3)) with dilation rate value as 1 is send to from dilated Convolution with single dimension of size (64, (1*3)) with dilation rate value as 2 as shown in the Figure 4. Dilated convolutions with dilation rates larger than one will generate artifacts known as gridding artifacts. Therefore, adjacent units in the output are calculated from entirely detach sets of units in the input and thus have entirely diverse real receptive fields [34].

The output attained from dilated Convolution with single dimension of size (64, (1*3)) with dilation rate value as 2 is forwarded to dilated Convolution with single dimension of size (64, (1*3)) with dilation rate value as 1. The output obtained from dilated Convolution with single dimension of size (64, (1*3)) with dilation rate value as 1 is send to a linear layer as shown in the Figure 3. Finally, the output is obtained from linear layer.

3.1.c Traffic Coder loss

GANs try to duplicate a probability distribution. Here, loss functions replicate the distance between the allotment of the data generated by GAN and the allotment of the actual data. GANs are very dominant, but not faultless. They are tough to train and the outcomes at a halt frequently undergo from artifacts. Separable convolutions first carry out depth wise spatial convolution. This action is performed on each input channel individually. This process is pursued by a point wise convolution which mixes the resulting output channels. It also manages how many output channels are generated per input channel in the depth wise process. Naturally, separable convolutions can be unstated as a method to factorize a convolution kernel into two smaller kernels. It is also an extreme version of an Inception block [51].

Global Average Pooling (GAV) is a pooling process considered to substitute fully connected layers in classical CNNs. The scheme is to produce one feature map for every consequent class of the classification step. One benefit of GAV in the fully connected layers is that the convolution structure is forced communication between feature maps and classes. Thus the features maps can be effortlessly understand as classes confidence maps. A new benefit is that there is no parameter to optimize. Hence, over fitting is avoided at this layer. Moreover, it sums out the spatial information. This method is also faster to spatial translations of the input [37].

Global Max Pooling (GMP) is defined as an ordinary max pooling layer with pool sizes same to the size of the input. GMP has critical significance in providing input to segment-by-segment processing. The input is considered completely at the instant of processing. Performance is improved when convolutional layer is combined with global max pooling layer [38].

Dense layer is the only definite and basic network layer. A Dense layer provides all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. This layer is the common and frequently utilized layer that includes a deeply connected neural network layer. It is connected to each and all one node in the subsequent layer [42].

Reshape layer is applied to vary the dimensions of the given input, without varying its information. Similar to flatten layer, only the dimensions are altered; no data is copied in the process. Output dimensions are also précised [52].

When functioning with images, the most excellent approach is Convolutional Neural Network (CNN) architecture [39]. Images are forwarded through convolutional layers, whereas

numerous filters extract significant features [40]. After forwarding some convolutional layers in sequence, the output is associated to a fully-connected dense network [41].

Figure 5 represents the framework used in Traffic coder loss. When the input is fed into Contextual loss framework, it is divided and forwarded to two Separable convolutions. They are namely, Separable Con2D (256 features, 3 filter size) and Separable Con2D (256 features, 5 filter size). The output of these two Separable convolutions is individually forwarded to GAV layer. From this layer, the output is forwarded to dense layer with 256 features. The output obtained from dense layer is sent to reshape layer with size (1, 1) and 256 features. Now, the output obtained from both the reshape layers are forwarded to Average layer.

Then, again the output obtained from average layer is divided and forwarded to two convolution networks, namely Conv2D (256 features, 3 filter size). The output obtained from these two convolution layers is forwarded to GMP layer individually. From this layer, the output is forwarded to dense layer with 128 features. The output obtained from dense layer is sent to reshape layer with size (1, 1) and 128 features. Now, the output obtained from both the reshape layers are forwarded to Maximum layer as shown in the Figure 5.

Subsequently, all over again the output obtained from maximum layer is divided and forwarded to two convolution networks, namely Conv2D (64 features, 1 filter size) and Separable Con2D (64 features, 1 filter size). The output obtained from these two convolution layers is promoted to GAV layer independently. For a second time, the output obtained from GAV layer is forwarded to dense layer with 256 features. The output obtained from dense layer is sent to reshape layer with size (1, 1) and 256 features. Now, the output obtained from both the reshape layers are forwarded to average layer as depicted in the Figure 5.

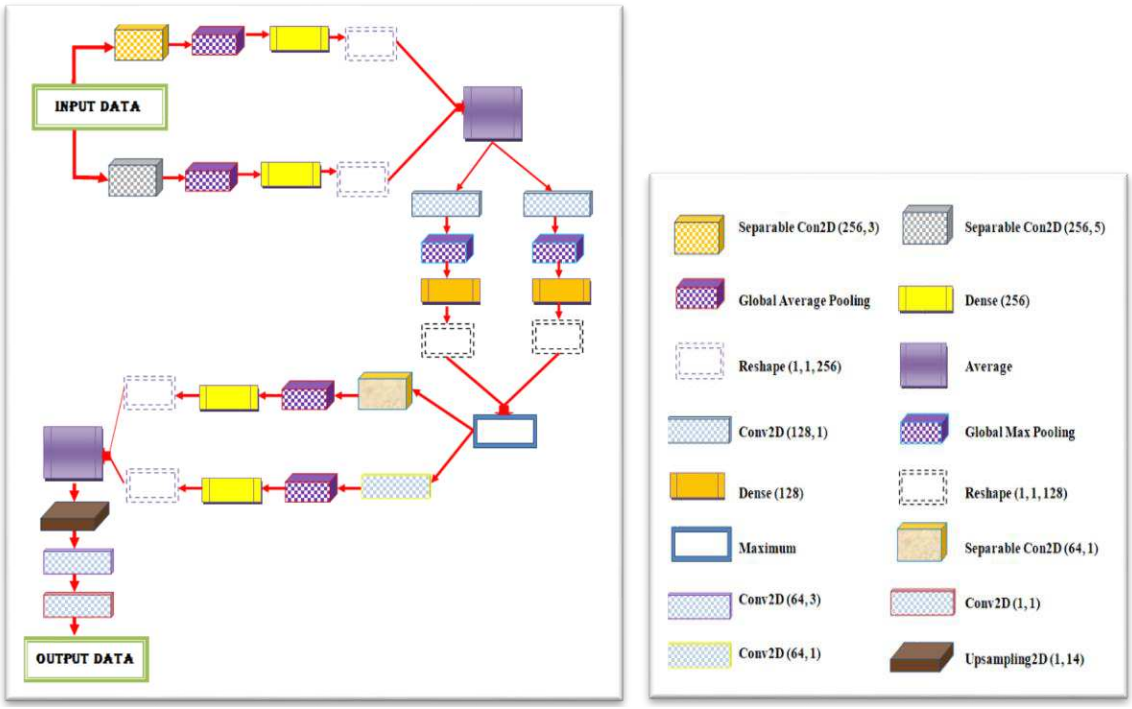

Figure 5 Traffic coder Loss

Finally, the output obtained from the average layer is forwarded to upsampling layer with (14 features, 1 filter size). Hence, the output received from upsampling layer is sent to convolution

layers. First, the output is applied to Con2D (64 features, 1 filter size). Then it is sent to Con2D (1 feature, 1 filter size). At last the output is received.

3.1.d Contextual loss

Contextual loss is a balancing technique for sustaining natural internal statistics [36]. This loss trains a feed-forward CNN to maintain natural interior information. Contextual loss function represents the difference value attained from the generated traffic and a actual traffic as represented in Eq. (3) and Eq. (4).

$$con_L = \|u - \hat{u}\|_1 \quad (3)$$

$$con_L = E_{u\sim \mathrm{pu}} \|\mathrm{u} - G(u)\|_1 \quad (4)$$

Here, u represents a actual traffic and $\hat{u}$ represents the generated traffic. Figure 6 represents contextual loss function.

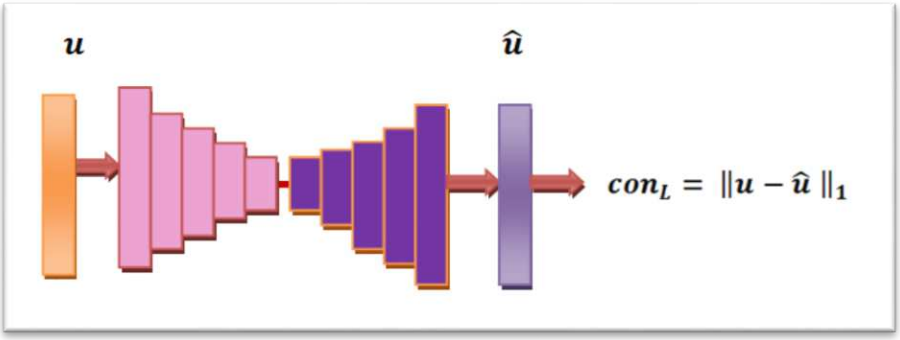

Figure 6 Contextual Loss

3.1.e Discriminator loss

Discriminator loss maximizes the average of the log probability for actual traffics and the log of the inverted probabilities of fraud traffics. Discriminator loss function represents the difference value between the generated traffic and a actual traffic obtained from the generator. The output of the generator is passed to estimate actual or fraud using function as shown in the eq. (3).

$$dis_L = \|f(u) - f(\hat{u})\|_2 \quad (5)$$

Figure 7 represents discriminator loss function. In this work, two inputs are provided as input to the discriminator. First, input that is provided to the generator is considered as the first input of discriminator. Second, the output that is obtained from the generator is considered as the second input of discriminator. Both these values, i.e.. $u, \hat{u}$ are provided as input to the discriminator for calculating the loss function.

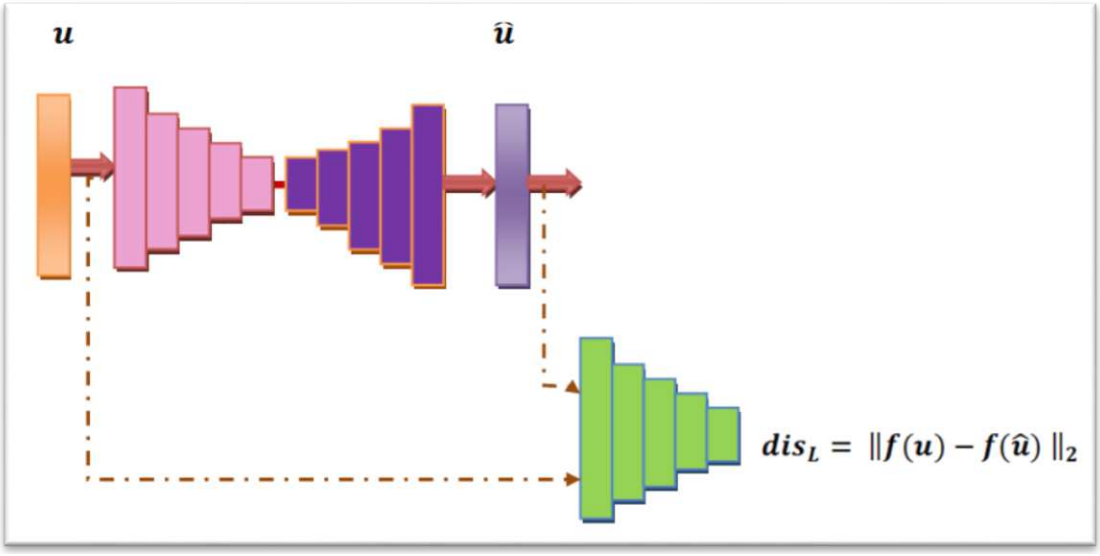

Figure 7 Discriminator Loss

To authenticate the proposed GAN network hypothesis, an objective function is estimated by accumulating three loss functions together. Here, each loss functions optimize individual sub networks. An objective function of generator is calculated by combining Contextual loss (con_L), Discriminator Loss (dis_L) and Traffic Encoder loss (te_L). Overall, the loss estimated using GAN network is obtained using Eq. (6).

$$FINAL_{LOSS} = con_L con_W + dis_L dis_W + te_L te_W \quad (6)$$

Where con_W, dis_W, te_W are the weighting constraints altering the impact of individual losses to the overall objective function. The summation of all weighting values is obtained as 1.

3.2 Classification

Bidirectional Gated Recurrent Unit (BiGRU) is a series dealing out with model that contains of two GRUs. The first GRU is enchanting the input in a forward direction and the second GRU in a backwards direction. This method provides a better classification assignment which objective is to recognize a topic from specified time series chronological information [46]. BiGRU is otherwise called as Bidirectional Recurrent Neural Network (BRNN) which has only the input and the forget gates.

Gated Recurrent Units (GRUs) are a gating method in Recurrent Neural Networks (RNNs). This technique is initiated in 2014 by Kyunghyun Cho et al [43]. GRU is similar to Long Short Term Memory (LSTM) with a forget gate. GRU has only smaller quantity of parameters than LSTM because it doesn't have an output gate [44]. GRUs have been exposed to show evidence of better automation on assured smaller and less normal data sets [45]. GRU combines the forget gate and the input gate into a particular revise gate. It also unites the hidden state and the cell state at the same time [47].

Bidirectional Long Short Term Memory (Bidirectional LSTM) is the procedure of creating any neural network to have the progression in sequence in both directions either backwards or forward [49]. Bidirectional LSTMs are an expansion of conventional LSTMs that can provide progress form automation on series classification tribulations. Bidirectional recurrent neural networks (RNN) are actually accumulate together two independent RNNs. This model allocates the networks to have equally toward the back and onward data about the series at all time foot step.

Attention layers are deep learning layers that suggest the scheme of attention. Attention layers are basically a weighted mean diminution. Spatial attention layer is a module for spatial

attention in CNN. It produces a spatial consideration map by applying the inner spatial connection of facial appearances. This technique points out an informative part in features. In this process, first regular pooling and max pooling actions are applied. An efficient feature descriptor is obtained in this process along the channel axis. A spatial attention map is generated by applying a convolution layer [48]. Flattening is integrating all visible layers into the background layer to decrease the file size. This layer also crumples the spatial dimensions of the input into the channel dimension.

A dense layer is a layer that is extremely associated with its earlier layer. It explains the neurons of the layer are connected to every neuron of its earlier layer. Dense layer is common and repeatedly applied layer that consists of deeply connected Neural Network (NN) layer. This is also utilized as hidden layer that is connected to each and every one node in the subsequent layer. To enhance the automation of classification model in CNN, this layer is used [50].

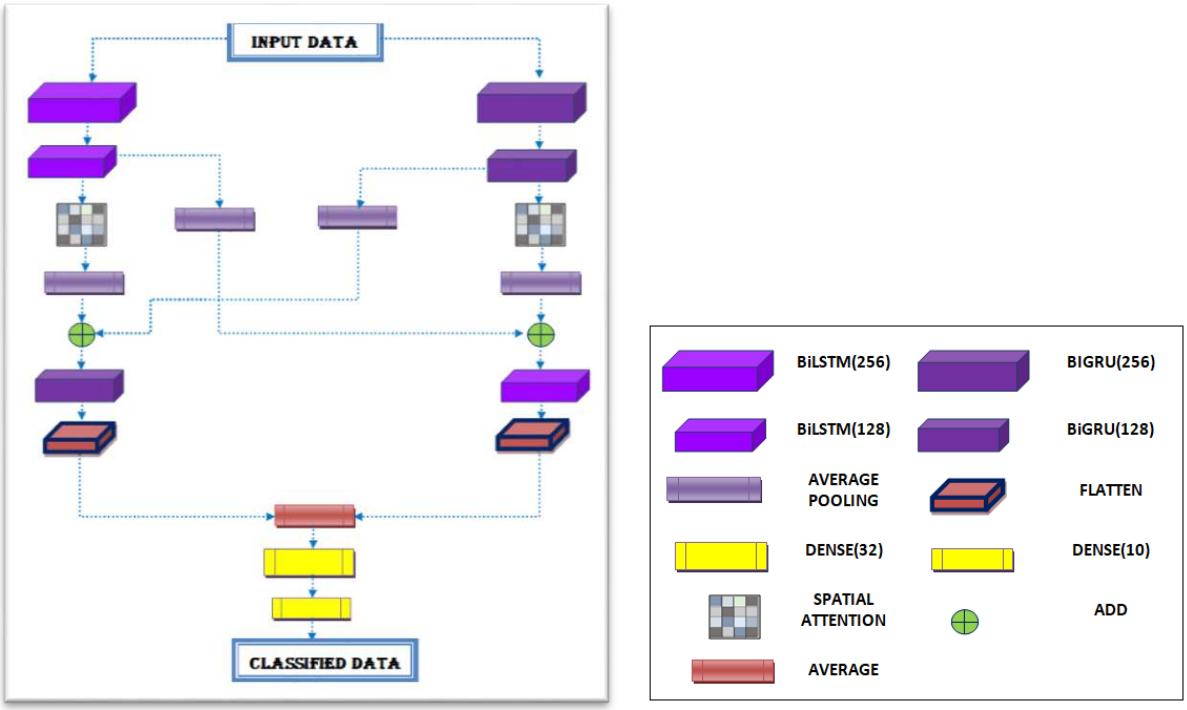

Figure 8 Classification framework

Figure 8 represents classification framework. The output obtained from GAN network is provided as input to classification framework. The input is fed into both the layers, namely BiLSTM of size 256 and BiGRU of size 256. The output obtained from BiLSTM of size 256 is given to BiLSTM of size 128 and the output obtained from BiGRU of size 256 is given to BiGRU of size 128. Now, the outputs from BiLSTM of size 128 and BiGRU of size 128 are provided to two layers, namely special attention and average pooling layer with size 2. Then, the outputs from both special attention layers are sent to average pooling layers with size 2 as shown in the Figure 8.

At the same time, the outputs from before average pooling layers with size 2 are interchanged and forwarded to add layer. It is interchanged in a way whereas, the output from average pooling of BiLSTM of size 128 is added using add layer with average pooling of BiGRU of size 128 and provided to BiLSTM of size 128. The same procedure is followed in average pooling of BiGRU of size 128 is added using add layer with average pooling of BiLSTM of size 128 and provided to BiGRU of size 128 as shown in the Figure 8.

The outputs obtained from both BiLSTM of size 128 and BiGRU of size 128 are provided as input to flatten layers. Both the outputs received from flatten layers are send to single average layer. From the average layer, the output is forwarded to dense layer with size 32. The output of dense layer with size 32 is sent as input to dense layer with size 12. Finally, the output that is obtained from dense layer with size 10 is considered as final output. The output obtained will be of 10 classes.

4 Automation Analyses

4.1 Data set explanation

Table 1 IoT-23 Data set classes with samples

CLASS	SAMPLES WITHOUT REDUNDANCY	SAMPLES TAKEN FOR ANALYSIS
0	**4253672**	**1700000**
1	**1699608**	**1200000**
2	**756**	**756**
3	**7707**	**7707**
4	**12648**	**12648**
5	**20612**	**20612**
6	**24492**	**24492**
7	**2999999**	**1200000**
8	**4619869**	**1200000**
9	**12908506**	**1700000**
Total samples	**26547869**	**7066215**

IoT-23 Data set

IoT-23 data set was given by the Avast AIC laboratory [12] and it is represented in the Figure 6. This data set contains 3 captures for benign anomalies and 20 malware captures from variety of IoT devices. This data set has an extension .pcap files. This data set is a unique network confine files acquired from conn.log.labeled files. The .pcap files are created by the network capture program called Wire shark. Totally 325,307,990 captures are available in the data set. In that 294,449,255 are malicious captures. Therefore, every conn.log.labelled files contains of 23 columns.

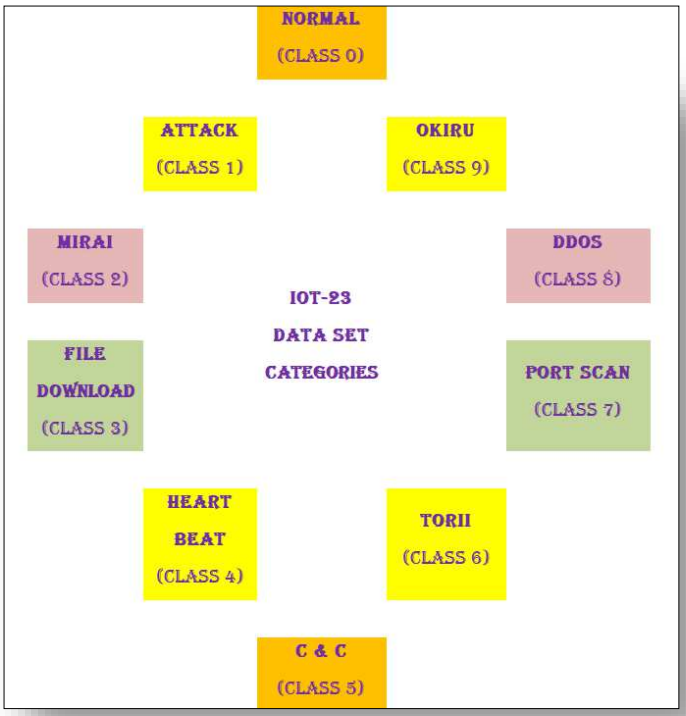

Figure 9 IoT-23 Data set

Figure 9 represents IoT-23 data set in detailed. In this data set, ten classes have been considered. The ten classes are namely Attack, Normal, File download, Mirai, Torii, C & C, Port Screen, Heartbeat, DDoS and Okiru. As represented in Table 1, in each class, samples without redundancy and samples taken for analysis are explained. Mirai which is of class 2 is the smallest class with samples; consist of 756 samples without redundancy and for analysis. Okiru which is of class 9 is the largest class with samples; consist of 12908506 samples without redundancy and 1700000 samples for analysis. This data set totally consists of 26547869 samples without redundancy and 7066215 samples for analysis.

4.2 Performance Analysis

4.2.1 F1-score

F1-score represents and select the image classification accuracy [39]. It is a computation technique applied to check accuracy [54]. It is calculated using recall (Re) and precision (Pr) result values. It is shown in Eq. (7).

$$\text{F1-score} = \frac{2*Pr*Re}{Pr+Re} \qquad (7)$$

Precision (Pr) [41] is called as the number of properly classified outcomes divided by the number of all obtained classified outcomes. Recall (Re) [41] is called as the number of properly classified outcomes divided by the number of images in the database [39]. It is shown in Eq. (8) and (9).

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad (8)$$

$$\text{Recall} = \frac{True\ Positive}{False\ Negative + True\ Positive} \qquad (9)$$

The term True Positive estimates the number of data that are properly classified as benign. The term False Negative estimates the number of data that are wrongly classified as benign data

as a bot net. The term False Positive estimates that a sample has wrongly predicted a real bot net as benign. The term True Negative estimates the number of data that are correctly classified as a bot net.

Accuracy is calculated as a measure of number of properly classified divided by the total number of classifications [53]. This is meant that the classified data set is resulting by the ratio of the number of properly classified tested images and the total number of tested images [54]. It is shown in Eq. (7).

$$\text{Accuracy} = \frac{Number\ of\ images\ classified\ correctly}{Total\ images\ in\ data\ set} \tag{7}$$

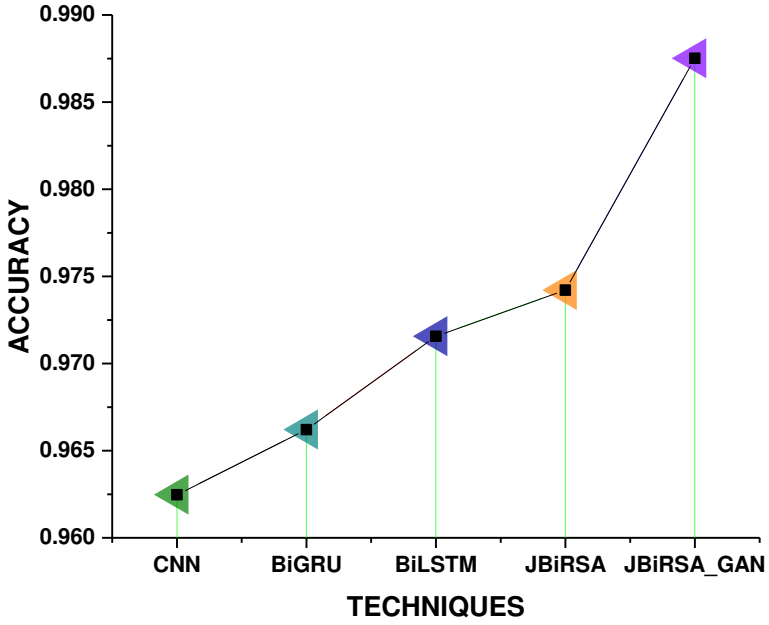

Figure 10 Comparison between the previous and JBiRSA with GAN in terms of accuracy

4.2.2 Comparison of Accuracy of the proposed model with the previous models

Figure 10 represents the comparison of accuracy between JBiRSA with GAN and the other existing methods. The proposed method with GAN architecture provides efficient outcomes on comparing with the previous methods. Every previous method provides outcomes nearly above 95%. Whereas CNN depicts an accuracy of 0.96, while Bidirectional GRU shows slight more accuracy than CNN. Even though CNN yields higher accuracy, the proposed methods JBiRSA and JBiRSA with GAN shows tremendous outcomes.

Compared to CNN and Bidirectional GRU, Bidirectional LSTM depicts 1% more accuracy. JBiRSA without GAN model also provides excellent outcomes compared to the other models. Finally JBiRSA with GAN provides an accuracy of 0.9875, which is 2% to 3% more compared to the previous ones. Hence this proposed framework is effective with the combination of Bidirectional GRU and Bidirectional LSTM techniques. Hence, GAN with the proposed model, yields better outcomes compared to the other models.

4.2.3 Comparison of Accuracy among the classes of the proposed model with the previous models

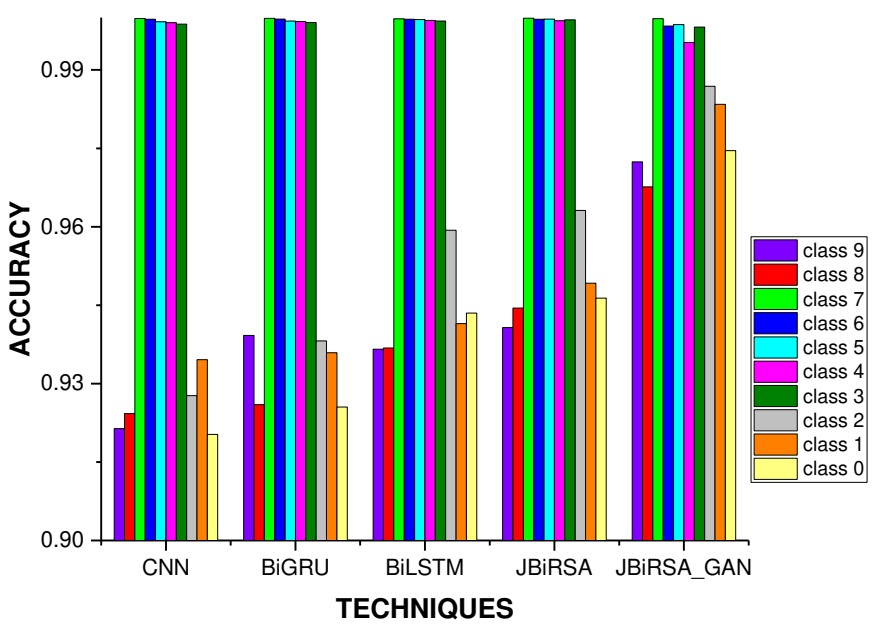

Figure 11 Comparison between the previous and JBiRSA with GAN for ten classes in terms of accuracy

Figure 11 depicts the comparison of ten classes with the previous models and JBiRSA with GAN. For class 0, CNN provides very less accuracy compared to the other methods which is 0.9213. Class 2 has very less samples compared to the other nine classes. For class 2, both the previous and the proposed method with GAN technique yield good outcomes. The proposed method with GAN architecture has proven with 99.97% highest accuracy for class 2 when compared with other nine classes. The next highest accuracy is depicted for class 4 with an accuracy of 99.86%. On comparing among ten classes and these techniques, CNN shows very less accuracy outcomes while comparing with the other classes and the exiting techniques. For class 5, all the techniques have predicted nearly same accuracy outcomes. For class 7, there is much accuracy variance among the techniques. The proposed with GAN architecture have shown nearly 98.68%, while without GAN it yields 96.31% and BiLSTM shows 95.93% accuracy outcomes. Whereas CNN yields 92.76% only and BiGRU shows 93.03% accuracy outcomes.

4.2.4 Comparison of precision of the proposed model with the previous models

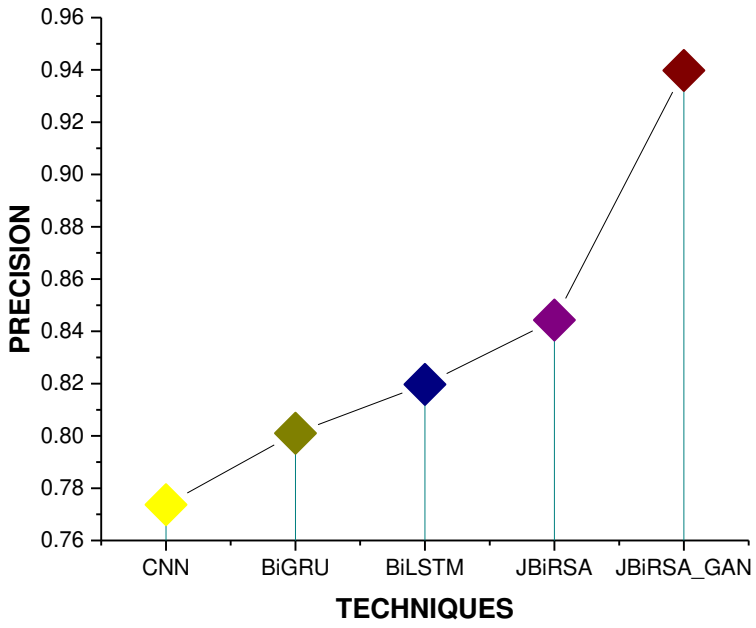

Figure 12 Comparison between the previous and JBiRSA with GAN in terms of precision

Figure 12 represents the comparison of Precision between JBiRSA, JBiRSA with GAN and the other existing methods. The proposed method JBiRSA with GAN network provides efficient outcomes compared to the other previous models. GAN is applied to generate new data occurrences that are similar to training information. Hence this proposed method shows efficient outcomes of 93.97% precision value, which is 10 % to 11% more compared to the other methods.

CNN provides only 77.37% precision which is very less compared to the other previous models. At the same time, precision values of both BiGRU and Bi LSTM are nearly similar. BiGRU gives in 80.10% precision value and Bi LSTM scores a value of 81.97%. precision value. The proposed method JBiRSA yields better outcomes than CNN, BiGRU and Bi LSTM. While comparing in precision values, JBiRSA with GAN provides efficient outcomes.

4.2.5 Comparison of Precision among the classes of the proposed model with the previous models

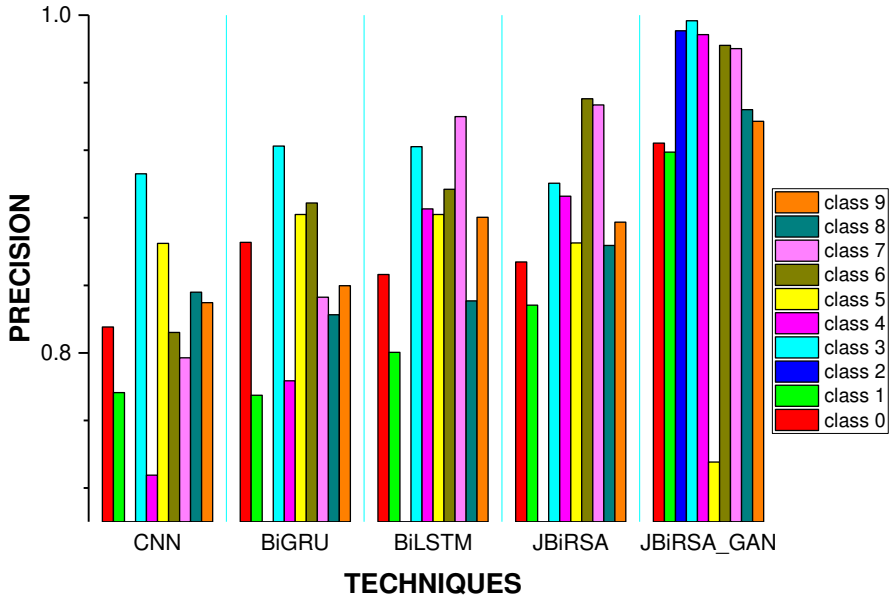

Figure 13 Comparison between the previous and JBiRSA with GAN for ten classes in terms of Precision

Figure 13 represents the comparison of precision values among the ten classes. JBiRSA with GAN provides efficient outcomes for all classes compared to the previous techniques. Class 3 shows highest precision outcomes when compared among all classes. Class 3 depicts 99.33% precision outcomes which is more effective compared to the other classes. For class 3, all the techniques have shown nearly 90% of outcomes. Class 6 shows improvement in precision outcomes when compared among all classes.

Class 6 provides 98.21% precision outcomes while comparing with the other classes. It also considers 24492 samples for analysis. Class 2 shows very less precision outcomes for all techniques. Specially, CNN illustrates only 37.18% precision result which is very less. BiGRU depicts 39.81% which is more compared to BiLSTM precision value of 31.32%. But JBiRSA with GAN depicts 99.07% which is almost 20% to 30% more compared to the other techniques.

4.2.6 Comparison of recall of the proposed model with the previous models

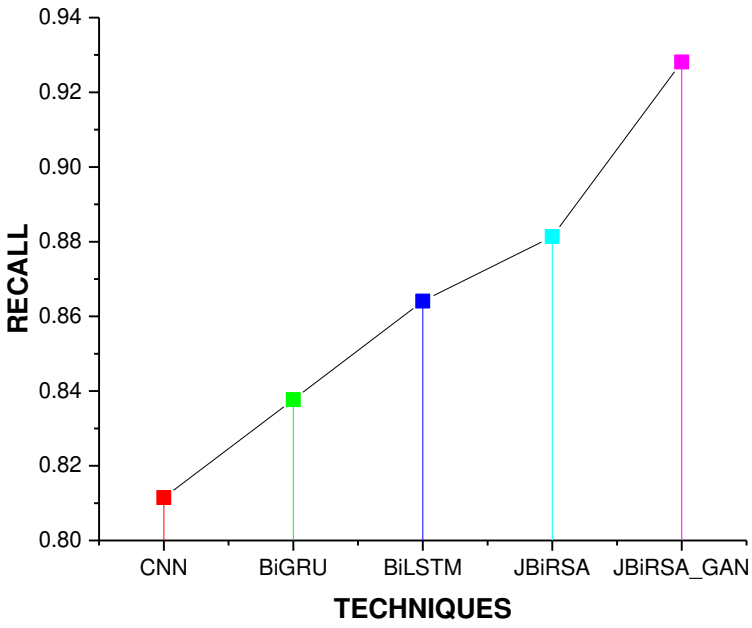

Figure 14 Comparison between the previous and JBiRSA with GAN in terms of Recall

Figure 14 represents the comparison of recall between JBiRSA with GAN and the other existing methods. BiGRU shows 83.77% recall value which is 2% more compared to CNN, which illustrates 81.14% of recall value. JBiRSA without GAN architecture has shown only 88.13% which is 4% more compared to the other methods. JBiRSA with GAN have illustrated 92.81%, which is 6% to 8% more compared to the other techniques.

4.2.7 Comparison of Recall among the classes of the proposed model with the previous models

Figure 15 represents the comparison of recall values among the ten classes. JBiRSA with GAN provides efficient outcomes for all classes compared to the previous techniques. CNN and BiGRU provide very poor outcomes for classes 7 and 8. CNN shows 77.03% recall value for class 7 and 76.48% recall value for class 8. BiGRU shows 79.54% recall value for class 7 and 79.36% recall value for class 8.

BiLSTM shows 81.24% recall value for class 7 and 82.3% recall value for class 8. JBiRSA without GAN architecture provides 82.93% recall value for class 7 and 83.23% recall value for class 8. JBiRSA with GAN shows efficient result of 93.77% recall value for class 7 and 95.4% recall value for class 8. For class 5, JBiRSA with GAN shows a 98.68% f1_score result which is very efficient compared to the other previous methods. On comparison, class 0 provides nearly same outcomes in all techniques.

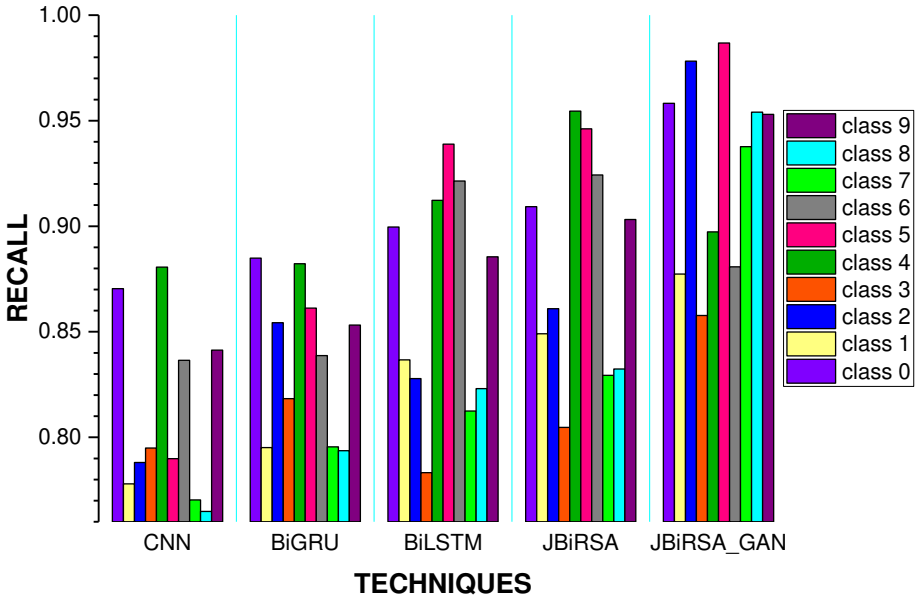

Figure 15 Comparison between the previous and JBiRSA with GAN for ten classes in terms of Recall

4.2.8 Comparison of F1_score of the proposed model with the previous models

Figure 16 represents the comparison of F1_score between JBiRSA with GAN and the other existing methods. F1_score is a benchmark metric, where JBiRSA with GAN provides 93.09% outcomes which provide 8% to 10% more improvement compared to the previous methods. But CNN shows a 78.35% F1_score result which is very less compared to the other methods. BiGRU and BiLSTM techniques illustrate nearly the same outcomes.

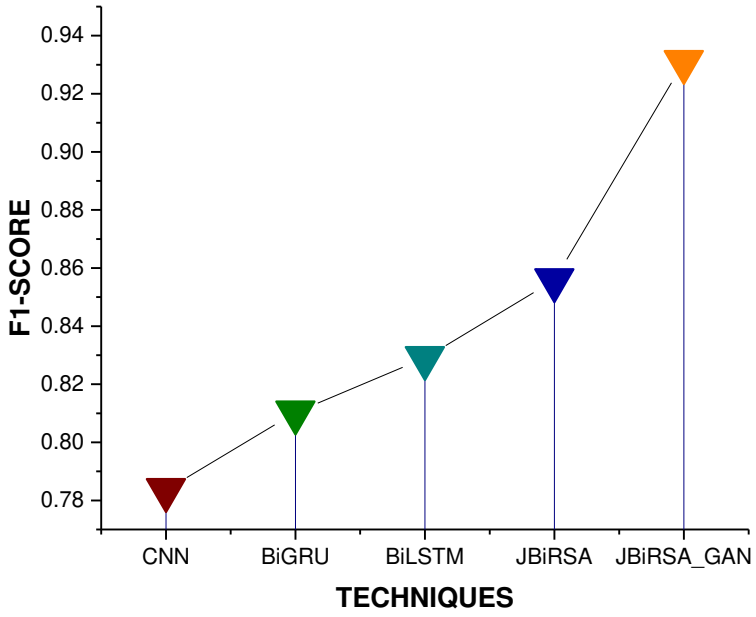

Figure 16 Comparison between the previous and JBiRSA with GAN in terms of F1-score

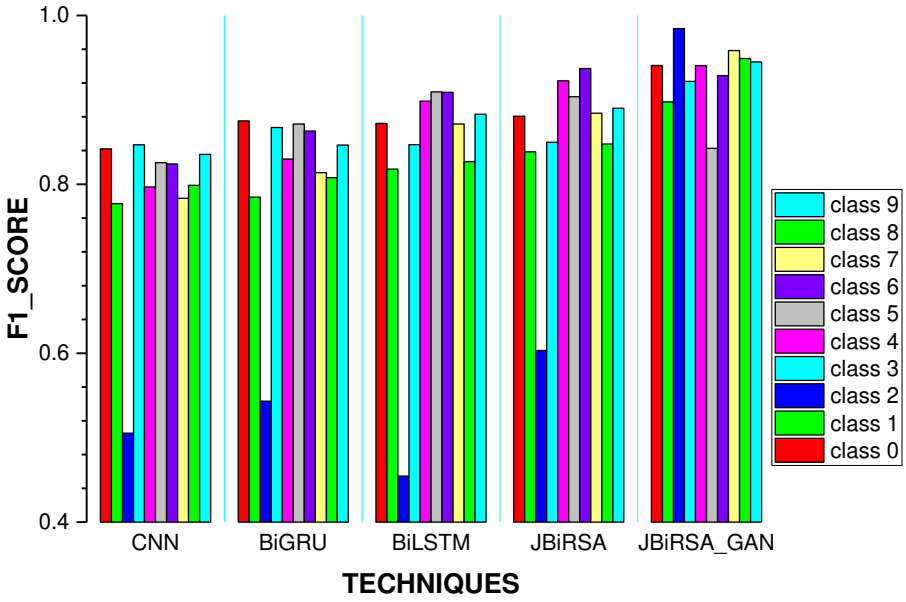

Figure 17 Comparison between the previous and JBiRSA with GAN for ten classes in terms of F1_score

4.2.9 Comparison of F1_Score among the classes of the proposed model with the previous models

Figure 17 represents the comparison of f1_score values among the ten classes. JBiRSA with GAN provides efficient outcomes for all classes compared to the previous techniques. Class

2 provides very less f1_Score outcomes compared to the other methods. On comparing, BiLSTM provides only 45.45% f1_Score outcomes on compared with other methods. CNN and BiGRU yields 50.53% and 54.31% f1_Score outcomes than BiLSTM. Hence JBiRSA with GAN yields 98.44%, which is 30% to 37% more compared to the other methods.

4.2.10 Comparison on training Losses of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models

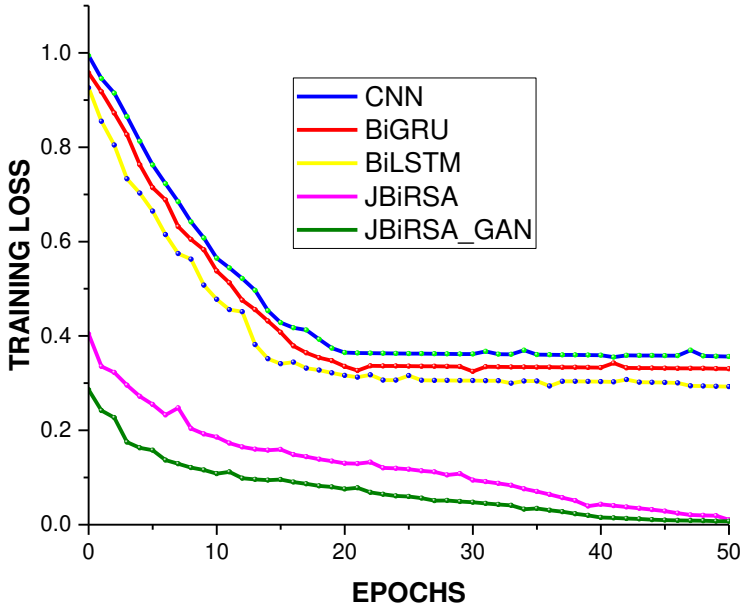

Figure 18 Training losses of CNN, BiGRU, Bilstm, JBiRSA and JBiRSA_GAN models

Figure 18 represents training losses of CNN, BiGRU, Bilstm, JBiRSA and JBiRSA_GAN models. In this work to estimate training loss, 0 to 50 epochs are considered for experimental analysis. As shown in the figure 18, when epochs for training process increases loss decreases naturally. For CNN, depending on epoch's training loss decreases from 99.4% to 35.64%. For BiGRU, training loss decreases from 95.75% to 33% and BiLSTM decreases from 92.6% to 29.27%. JBiRSA without GAN shows outcomes of training loss from 40.42% and JBiRSA with GAN depicts efficient outcomes of training loss from 28.6%.

4.2.11 Comparison on training accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models

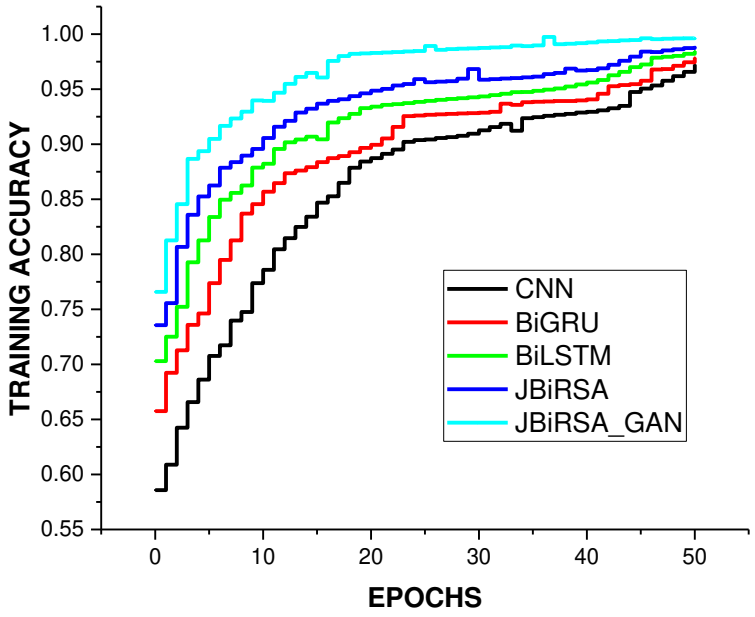

Figure 19 Training Accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models

Figure 19 represents training Accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models. In this work to estimate training accuracy, 0 to 50 epochs are considered for experimental analysis. As shown in the figure 16, when epochs for training process increases accuracy also increases naturally. In JBiRSA with GAN, training accuracy is improved tremendously with efficient outcomes. JBiRSA with GAN provides 99.65% training accuracy.

4.2.12 Overall training accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models

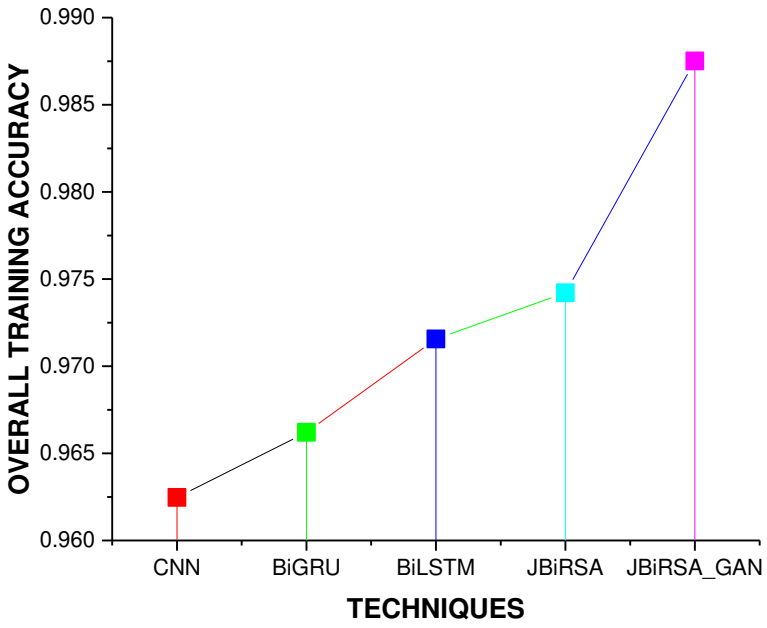

Figure 20 Overall training accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models

Figure 20 represents an overall training accuracy of CNN, BiGRU, BiLSTM, JBiRSA and JBiRSA_GAN models. CNN yields an overall training accuracy of 96.24%, whereas BiGRU provides an overall training accuracy of 96.62%. Compared to these two methods, BiLSTM illustrates 97.15% overall training accuracy with an improvement of 1.5%. But JBiRSA with GAN shows 98.75% which is 9% to 11% more than all other existing methods.

5 Conclusion

In this work, feature generation process is very much important. Hence a novel method called Joined Bi-Model RNN with Spatial Attention (JBiRSA) is proposed and combined with Generative Adversarial Network (GAN). GAN consist of two neural networks, namely Generator and Discriminator. In GAN, loss is also calculated to enhance accuracy. IoT-23 data set is utilized with 20 malware confines from various IoT devices and 3 precincts for benign anomalies. JBiRSA with GAN has proven to be efficient and has the potential to differentiate between benign and malicious traffic data in IoT attacks.

This proposed framework with GAN architecture illustrates 98.75% accuracy, which is 2% to 3% more compared to the previous techniques, namely CNN, BiGRU, BiLSTM. This proposed framework with GAN architecture illustrates 93.97% precision rate which is 8% to 10% more compared to the previous techniques, namely CNN, BiGRU, BiLSTM. JBiRSA with GAN yields 92.81% recall rate and 93% F1_score, which is 10% to 12% more compared to the other models.

Statements and Declarations:

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability

IoT-23 data set which was given by the Avast AIC laboratory [12] is used in this work

6 References

1. Jiyeon Kim, Minsun Shim, Seungah Hong, Yulim Shin and Eunjung Choi, 2020, “Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning”, Applied Sciences.
2. Segun I. Popoola, Bamidele Adebisi, Ruth Ande, Mohammad Hammoudeh, Kelvin Anoh and Aderemi A. Atayero, 2021, “SMOTE-DRNN: A Deep Learning Algorithm for Botnet Detection in the Internet-of-Things Networks”, Sensors, Vol. 21, No. 2985.
3. Tarun Ganesh Palla and Shahab Tayeb, “Intelligent Mirai Malware Detection for IoT Nodes”, Electronics, Vol. 10, No. 1241.
4. Segun I. Popoola, Bamidele Adebisi, Ruth Ande, Mohammad Hammoudeh and Aderemi A. Atayero, 2021, “Memory-Efficient Deep Learning for Botnet Attack Detection in IoT Networks”, electronics, Vol. 10, No. 1104.
5. Abdulghani Ali Ahmed, Waheb A. Jabbar, Ali Safaa Sadiq and Hiran Patel, 2020, “Deep learning-based classification model for botnet attack detection”, Springer, Journal of Ambient Intelligence and Humanized Computing.
6. Rizwan hamid randhawa, Nauman aslam, Mohammad alauthman, Husnain rafiq and Fank comeau, 2021, “Security Hardening of Botnet Detectors Using Generative Adversarial Networks”, IEEE access, Vol. 9, pp. 78276 - 78292.
7. Al-Garadi, Mohamed, Al-Ali, Du, Ali and Guizani, 2018, “A Survey of machine and deep learning methods for internet of things (IoT) security”, arXiv 2018, arXiv:1807.11023.
8. Alauthman, Aslam, Al-Kasassbeh, Khan, Al-Qerem and K.-K. R. Choo, 2020, “An eficient reinforcement learning-based botnet detection approach," Journal of Network Computing Applications, Vol. 150, Art. no. 102479.
9. Papernot, McDaniel, Goodfellow, Jha, Celik, and Swami, 2017, “Practical black-box attacks against machine learning," in Proc. ACMAsia Conference Computing Communication Security, pp. 506 - 519.
10. Yavuz, Devrim, and Ensar, 2018, “Deep learning for detection of routing attacks in the internet of things”, International Journal of Computing Intelligence Systems 2018, Vol. 12, pp. 39–58.
11. Azmoodeh, Dehghantanha, Conti, and Choo, 2018, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint”, Journal of Ambient Intelligent Humaniz Computing, Vol.9, pp. 1141–1152.
12. Agustin Parmisano, Sebastian Garcia and M. J. E, 2018, “IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic. — Stratosphere IPS.
13. Aceto, Persico and Pescape, 2019, “A survey on information and communication technologies for Industry 4.0: State-of-the-art, taxonomies, perspectives, and challenges”, IEEE Communication Survet Tutor. Vol. 21, pp. 3467–3501.
14. Chuanlong Yin, Yuefei Zhu, Shengli Liu, Jinlong Fei and Hetong Zhang, 2018, “An Enhancing Framework for Botnet Detection Using Generative Adversarial Networks”, IEEE, International conference on Artificial Intelligence and Big Data.
15. Gartner Top 10 Strategic Technology Trends for 2020. Available online: https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for2020.
16. Haddad Pajouh, Dehghantanha, Khayami and Choo, 2018, “A deep recurrent neural network based approach for internet of things malware threat hunting”, Future Generation Computing Systems, Vol. 85, pp. 88–96.
17. Hodo, Bellekens, Hamilton, Dubouilh, Iorkyase, Tachtatzis and Atkinson, 2016, “Threat analysis of IoT networks using artificial neural network intrusion detection system”, In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, pp. 1–6.
18. Meidan, Bohadana, Mathov, Mirsky, Shabtai, Breitenbacher and Elovici, 2018, “N-BaIoT—Network-based detection of IoT botnet attacks using deep autoencoders”, IEEE Pervasive Computing, Vol. 17, pp. 12–22.
19. Al Shorman, Faris and Aljarah, 2020, “Unsupervised intelligent system based on one class support vector machine and Grey Wolf optimization for IoT botnet detection”, Journal of Ambient Intelligent Humanization Computing, Vol.11, pp. 2809–2825.
20. Bastos, Marzano, Fonseca, Fazzion, Hoepers, Steding-Jessen, Chaves, Cunha, Guedes and Meira, ‘Identifying and Characterizing bashlite and mirai C&C servers’, In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Barcelona, Spain, pp. 1–6, 2019.
21. Xie, Wu, Liu and Li, ‘IoT data analytics using deep learning’, arXiv 2017, arXiv:1708.03854.
22. Alam, Mehmood, Katib, and Albeshri, ‘Analysis of eight data mining algorithms for smarter internet of things (IoT)’, Procedia Computer Science, Vol. 98, pp. 437–442, 2016.
23. Mcdermott, C.D., Majdani, F. and Petrovski, A.V, ‘Botnet detection in the internet of things using deep learning approaches’, In Proceedings of the International joint conference on neural networks 2018 (IJCNN 2018), 8-13 July 2018, Rio de Janeiro, Brazil. Piscataway: IEEE [online], article ID 8489489.
24. Imtiaz ullah and Qusay H. Mahmoud, “Design and Development of a Deep Learning-Based Model for Anomaly Detection in IoT Networks”, Vol. 9, pp. 103906 – 103926, 2021.

25. S. Sriram, R. Vinayakumar, M. Alazab, and S. KP, ``Network flow based IoT botnet attack detection using deep learning," in Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS), Jul. 2020, pp. 189194, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162668.
26. C. Yin, S. Zhang, J. Wang, and N. N. Xiong, ``Anomaly detection based on convolutional recurrent autoencoder for IoT time series," IEEE Trans. Syst., Man, Cybern. Syst., early access, Feb. 7, 2020, doi: 10.1109/tsmc.2020.2968516.
27. Aidin Ferdowsi; and Walid Saad, "Generative Adversarial Networks for Distributed Intrusion Detection in the Internet of Things", 2019 IEEE Global Communications Conference (GLOBECOM), 2020.
28. Lan Lan, Lei You, Zeyang Zhang, Zhiwei Fan, Weiling Zhao, Nianyin Zeng, Yidong Chen and Xiaobo Zhou, "Generative Adversarial Networks and Its Applications in Biomedical Informatics", Front Public Health. Vol. 8, No. 164, 2020.
29. Khalid Albulayhi, Abdallah A. Smadi, Frederick T. Sheldon and Robert K. Abercrombie, "IoT Intrusion Detection Taxonomy, Reference Architecture, and Analyses", Sensors (Basel). Vol.21, No.19, 2021
30. Farooq Shaikh, Nasir Ghani and Elias Bou-Harb, "IoT Threat Detection Leveraging Network Statistics and GAN", 2019.
31. P.P.Ray, "A survey on Internet of Things architectures", Elsevier, Journal of King Saud University - Computer and Information Sciences, Volume 30, Issue 3, July 2018, Pages 291-319.
32. Abien Fred M. Agarap, "Deep Learning using Rectified Linear Units (ReLU)", arXiv:1803.08375v2 2019.
33. Fisher Yu and Vladlen Koltun, "multi-scale context aggregation by dilated convolutions" Published as a conference paper at ICLR 2016.
34. Zhengyang Wang and Shuiwang Ji, "Smoothed Dilated Convolutions for Improved Dense Prediction", IEEE transactions on pattern analysis and machine intelligence, 2019.
35. Yuhong Li, , Xiaofan Zhang and Deming Chen, "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes", IEEE CVPR 2018.
36. Roey Mechrez, , Itamar Talmi and Lihi Zelnik-Manor, "The Contextual Loss for Image Transformation with Non-Aligned Data", ECCV 2018.
37. Min Lin, Qiang Chen, Shuicheng Yan, "Network In Network", Neural and Evolutionary Computing (cs.NE); Computer Vision and Pattern Recognition (cs.CV); Machine Learning (cs.LG), 2013.
38. Hannah Kim and Young-Seob Jeong, "Sentiment Classification Using Convolutional Neural Networks", Appl. Sci. 2019, Vol. 9, Issue. 2347.
39. Ani Brown Mary N & Dejey Dharma, "Coral reef image/video classification employing novel octa-angled pattern for triangular sub region and pulse coupled convolutional neural network (PCCNN)", Springer, Multimedia Tools and Applications, Vol. 77, pp. 31545–31579, 2018.
40. Dr. N. Ani Brown Mary, Mrs. N. Adline Rajasenah Merryton and Dr. D. Sheefa Ruby Grace, "BANANA leaves diseases classification using DPVP and PCCNN", International Virtual Conference on Recent Trends and Techniques in Mathematical and Computer Science, 2021.
41. Ani Brown Mary N , Robert Singh A. & Suganya Athisayamani, "Banana leaf diseased image classification using novel HEAP auto encoder (HAE) deep learning", Springer, Multimedia Tools and Applications, Vol. 79, pp. 30601–30613, 2020.
42. Helen Josephine V, A.P.Nirmala and Vijaya Lakshmi Alluri, "Impact of Hidden Dense Layers in Convolutional Neural Network to enhance Performance of Classification Model", ICETCE 2021, IOP Conf. Series: Materials Science and Engineering, 1131, 2021.
43. Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger; Bengio, Yoshua (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". arXiv:1406.1078.
44. "Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano – WildML". Wildml.com. 2015-10-27. Retrieved May 18, 2016.
45. Gruber, N.; Jockisch, A. (2020), "Are GRU cells more specific and LSTM cells more sensitive in motive classification of text?", Frontiers in Artificial Intelligence, 3: 40, doi:10.3389/frai.2020.00040.
46. Htet myet lynn, Sung bum pan and Pankoo kim, , "A Deep Bidirectional GRU Network Model for Biometric Electrocardiogram Classification based on Recurrent Neural Networks", IEEE access, 2019.
47. Xing Yin, Changhui Liu and Xiaodong Fang, "Sentiment analysis based on BiGRU information Enhancement", Journal of Physics: Conference Series, ISCME 2020.
48. Sanghyun Woo, Jongchan Park, Joon-Young Lee and In So Kweon, "CBAM: Convolutional Block Attention Module", arXiv.org > cs > arXiv:1807.06521v2, 2018.

49. Md. Mostafizer Rahman, Yutaka Watanobe and Keita Nakamura, “A Bidirectional LSTM Language Model for Code Evaluation and Repair”, Symmetry, Vol.13, Issue. 247, 2021.
50. Helen Josephine V, A.P.Nirmala and Vijaya Lakshmi Alluri, “Impact of Hidden Dense Layers in Convolutional Neural Network to enhance Performance of Classification Model”, IOP Conference Series: Materials Science and Engineering, Vol. 1131, ICETCE 2021.
51. François Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions”, Computer Vision and Pattern Recognition (cs.CV), 2017.
52. Swarnendu Ghosh, Nibaran Das and Mita Nasipuri, “Reshaping Inputs for Convolutional Neural Networks -Some common and uncommon methods”, Pattern Recognition, Vol. 93. Issue. 10, 2019.
53. K. Jayapriya and N. Ani Brown Mary, “Employing a novel 2-gram subgroup intra pattern (2GSIP) with stacked auto encoder for membrane protein classification”, Springer, Molecular Biology Reports, https://doi.org/10.1007/s11033-019-04680-3.
54. N. Ani Brown Mary and Dejey Dharma, “A novel framework for real-time diseased coral reef image classification”, Springer, Multimedia Tools and Applications, https://doi.org/10.1007/s11042-018-6673-2.
55. Ramasamy, Lakshmana & Kadry, Seifedine. (2021). Internet of things (IoT). 10.1088/978-0-7503-3663-5ch1.
56. Skarmeta, Antonio & Hernández-Ramos, José & Bernal Bernabe, Jorge. (2015). A required security and privacy framework for smart objects. 10.1109/Kaleidoscope.2015.7383648.