



The Top Ten Artificial Intelligence-Deep Neural Networks for IoT Intrusion Detection System

V. Kanimozhi¹ · T. Prem Jacob¹

Accepted: 8 February 2023 / Published online: 18 February 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Despite the fact that there are numerous methods for detecting IoT intrusions, this research explorations conducted the implementation of the Top 10 Artificial Intelligence—Deep neural networks may be advantageous for unsupervised as well as supervised learning concerning IoT network traffic data. It shows a thorough comparison study to detect IoT intrusions on intelligent embedded devices which are essential to detect such intrusions using the most recent dataset IoT-23. Although several solutions are being developed to secure IoT networks, development might still be required. The use of different deep learning techniques may enhance IoT security. To enhance the security execution of IoT network traffic, the top 10 deep-learning approaches were investigated using the realistic IoT-23 dataset. For recognizing five different IoT attack classes—DoS (Denial of Service), Mirai, Scan, Normal records, and MITM-ARP (Man in the Middle attack)—we developed a variety of neural network models. In deep-learning neural network models, a "softmax" function of multiclass classification may be used to identify these assaults. NumPy, Pandas, Scikit-learn, Scipy, TensorFlow 2.2, Seaborn, and Matplotlib were only some of the programs used in the Anaconda3 environment for this study. Healthcare, banking, finance, scientific research, and corporate organizations, as well as ideas like the Internet of Things, are just some of the many fields that have embraced the utilization of AI-deep learning models. We discovered that the best deep-learning algorithms are capable of minimizing function loss, improving accuracy, as well as reducing execution time for developing that particular model. By using cutting-edge technology like deep learning neural networks as well as artificial intelligence, it makes a significant contribution to the identification of IoT anomalies. As a result, it will be effective to reduce attacks on IoT organizations. CNN (Convolutional neural networks), GANs (generative adversarial networks), and multilayer perceptron provide the best accuracy scores of 0.996317, 0.995829, and 0.996157 among the top 10 neural networks, respectively, with the smallest loss function and the shortest execution times. This paper helped to fully understand the peculiarities of IoT anomaly detection. To help you better understand various neural network models and IoT anomaly detection, this study analysis shows the Top 10 AI-deep learning model implementations.

✉ V. Kanimozhi
kanimv@yahoo.co.in

Extended author information available on the last page of the article

Keywords IoT-23 · Top 10 AI-deep learning models · Anomaly detection · Unsupervised and supervised learning · Multiclass classification

1 Introduction

IoT devices are often exposed to security risks, which makes them subject to transparency problems. The Internet of Things (IoT) has captured the interest of many professionals in latest days and are searching for strategies to fight against IoT assaults such as Mirai and Botnet by applying cutting-edge technology. When developing IoT projects, some of the most notable IoT threats are taken into account. The imprecise software used in surveillance camera hacking allows anybody with access to the IP address of the camera to log in as a programmer. The IoT equipment is used to identify and have an impact on patients' cardiac capacity limits, such as pacemakers and defibrillators. Let's have a look at Mirai, an amazing Botnet that manipulates IoT devices to unleash a massive amount of distributed DDoS attacks [1]. A Mirai variant called "Thingbot" keeps producing and contaminating Internet Protocol (IP) cameras [2]. As a result, attacks that are more uncommon and devastating than Mirai are perceived as IoT system vulnerabilities. Additionally, to address several shortcomings and issues with IoT applications, we tried to identify frameworks in IoT systems throughout the investigation [3]. Making profitable and effective IoT security solutions served as a search center throughout this task, as previously said. Therefore, the creation of a robust method to defend the system from malicious assaults on IoT traffic data networks, particularly in the banking finance, and medical sectors, is urgently needed. Despite the presence of several techniques, the goal of the examined article is to improve IoT device safety by using the Top 10 AI Deep Learning models and by containing the useful IoT-23 network traffic dataset.

2 Major Objective of the Proposal System

Artificial Neural Network has become quite common in the AI field, and organizations that deal with complicated challenges often use these techniques [4]. Currently, the operation of data reserving in calculating networks is exponentially high adding a fast data storehouse using deep learning (DL) stacked layers [5]. It imparts artificial intelligence with capabilities and a structure similar to the human nervous system. This technology operates on a massive volume of informative data records by using AI-Deep Learning models to carry out logical as well as statistical calculations [6]. Deep Learning always focuses on an idea of a system by using DNNs to detect the target accurately [7]. Ten distinct neural network models are implemented in the suggested system using the IoT-23 dataset [8]. Following is the list,

1. GAN Networks
2. Autoencoders
3. SOM Networks
4. Radial Basis Networks
5. Deep Belief Networks
6. RBM Model
7. RNN Model

8. LSTM Model
9. MLP Model
10. CNN Model

Artificial neurons that are nodes in a deep neural network are arranged like that of the human cerebrum, which is filled with biological neurons [9]. These artificial neurons are layered vertically into a three-layer deep neural network model [10].

Three layers are present: the output layer comes after the input layer and the invisible layer.

All the features are provided as the data Input. All of the incoming data is computed by the nodes using arbitrary weights and then summed with the bias. Different examinations have proved the fact that deep learning techniques uphold the evaluation of huge data sets productively [11]. To determine which node should be activated, we use activation functions, also known as non-linear functions.

The AI neural network identifies columns, groups comparable information, and then detects the important trends within machines for self-learning, during the learning neural network training phase. The computations for putting together the deep learning models are used in this at different stages. As a result, a Deep Learning model with a high computing capacity can handle large or complex data sets [12]. The most widely used AI models for IoT intrusion detection need to be thoroughly examined.

3 Dataset

On the website <https://www.stratosphereips.org/datasets-iot23>, the AIC research office supported by Avast Software is available as an 8.8 GB IoT-23 dataset that contains 3 class label subsets and 20 malware subsets ['Label', 'Cat', 'Sub Cat'] [13]. The Czech Stratosphere Laboratory created the dataset in January 2020 [14]. It aims to provide limitless data records [15] of organized IoT attacks derived from the actual occurrences to extend anomaly detection and provide a framework for safeguarding the model computations.

4 Dataset Pre-processing

As a first step, we identify columns in the dataset that have very big data values or null or infinite values. By correcting all of these noisy defects, the dataset is pre-processed and cleaned. Then it is evaluated using 83 characteristics and the 3 labels "Label," "Cat," and "Sub-Cat." We used the class label "Cat" with five different categories—Mirai, Scan, DoS, Normal Records, and MITM-ARP—for this multiclass categorization. If there is just one class, a binary function may be utilized. For example, a 'softmax' is an activation unit that can be utilized for more than two classes. We enter all 83 features with the label feature "Cat" as feature extraction and feature removal are not required to feed input dimension features into neural models. The total feature dimensions are 0–83 since the Pandas data frame technically begins with zero-indexing Fig. 1.

MITM_ARP attack numbers—35,377.

Normal record number—40,073.

NScan attack number—59,391.

DoS attack number—75,265.

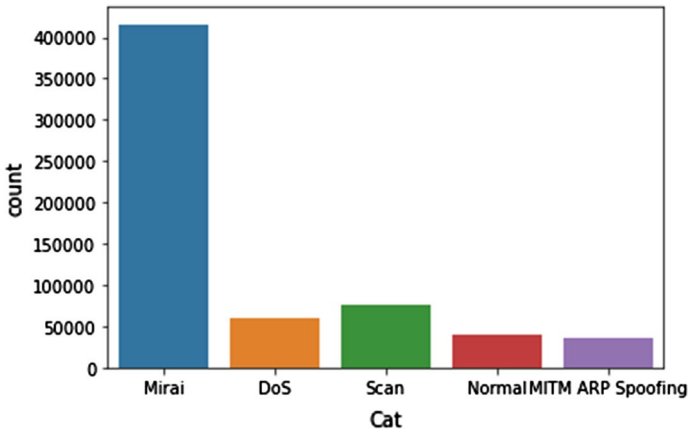


Fig. 1 'Category' label of IoT-23 dataset

Mirai attack number—415,677.

- Mirai: "Mirai" may be malware that targets Internet of Things (IoT) devices including TVs, printers, switches, and camcorders as it continues to spread. When determining usernames and passwords, the virus looks at pre-programmed or default options.
- Denial-of-Service: Release of attacks on IoT devices often involves a DoS attack. Additionally, if a DoS attack occurs, IoT devices are unable to reach the organization for further correspondence.
- Scan: This Scan virus filters the open ports used by IoT administrators, making it easy to attack certain IoT devices.
- Normal: if it is said to be a typical program and there is no virus.
- MITM_ARP: Inside the Middle assault, MITM may take care of business. Due to these modifications, the physical addresses are being attacked, the ARP Messages allow the invader to pose as a genuine user. As soon as the hacker connection is made, they have an approach to all communications and can even alter, add, and delete messages via ARP spoofing [16].

5 Top 10 AI—Neural Networks Models

5.1 Autoencoder

In an unsupervised learning method, autoencoders enable the reproduction of their input neurons as their output result. It takes the designs of neural networks and involves the functionalities of decoders and encoders. It makes sense of the idea of diminishing the intricacy by adopting an autoencoder structure [17]. It has two parts of an autoencoder: (i) encoder (ii) decoder. The encoders explore the feed data and then compress it, whereas the compression algorithms examine the crushed data and reconstitute the data fed signal. Through the hidden units, the encoder is capable to learn about the data which is compressed [18]. It uses the encoder segment to build compressed packed data after preparing it and discarding the decoder part (Table 1).

Table 1 Autoencoders model output result with execution time and 5 epochs

```

Model: "model_1"
-----
Layer (type)                Output Shape          Param #
-----
input_1 (InputLayer)        [(None, 83)]          0
-----
dense (Dense)               (None, 32)            2688
-----
dense_1 (Dense)             (None, 5)             33
-----
Total params: 2,721
Trainable params: 2,721
Non-trainable params: 0
Train on 400500 samples, validate on 100126 samples
Epoch 1/5
400500/400500 [=====] - 29s 73us/sample - loss: 0.0790 - acc: 0.9760 -
val_loss: 0.0565 - val_acc: 0.9841
Epoch 2/5
400500/400500 [=====] - 26s 65us/sample - loss: 0.0521 - acc: 0.9850 -
val_loss: 0.0450 - val_acc: 0.9855
Epoch 3/5
400500/400500 [=====] - 26s 66us/sample - loss: 0.0449 - acc: 0.9865 -
val_loss: 0.0416 - val_acc: 0.9868
Epoch 4/5
400500/400500 [=====] - 21s 51us/sample - loss: 0.0398 - acc: 0.9878
- val_loss: 0.0375 - val_acc: 0.9891
Epoch 5/5
400500/400500 [=====] - 21s 53us/sample - loss: 0.0353 - acc: 0.9895 -
val_loss: 0.0324 - val_acc: 0.9929
    
```

Loss: 0.033698

Accuracy: 0.992697

The steps followed in Autoencoder Algorithm

1. The sequential layer is used in the development of the autoencoder model’s encoder and decoder components.
2. The first model produced is called Encoder, which uses a transformation on the encoded signal to encrypt input data with 83 feature dimensions.
3. The second network that was developed is a decoder, which generates a compressed signal from an input signal that has been encoded.
4. The training and testing stages of the dataset are divided in proportion 80:20.
5. 'Softmax' as an activation unit used to distinguish the original signal and the five artificially created class states, the loss "categorical_cross_entropy" is determined.
6. The autoencoder neural network model is created in this manner, and until it converges, it attempts to determine the identity function of the input.

5.2 GAN Adversarial AI-Neural Network Model

It’s an unsupervised model made up of two different kinds of neural networks: discriminator and generator neural networks [19]. GANs combine two different network models, one designed to receive input records and the latter to distinguish between false and real records. (It is now known as the "adversarial" nature of GANs) [20] (Fig. 2).

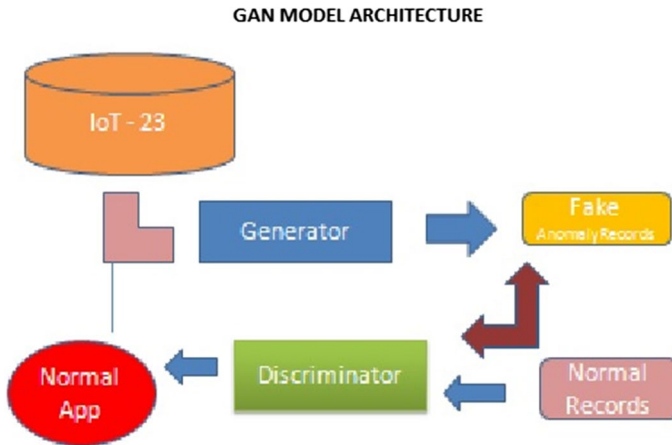


Fig. 2 GAN Model Architecture

Lowering the model’s loss function will enable it to interpret and learn the likelihood function of the output. During the training phase, a technique for managing the GANs’ parameters is employed. Follows a comprehensive illustration of the GAN model (Fig. 3 and Tables 2 and 3).

Steps followed in GAN Model Algorithm

1. It passes through two learning stages. The discriminator model is trained in the first stage of training. The IoT-23 false data is produced by the generator $G(z)$ utilizing a few previous distributions. The discriminator uses supervised learning to distinguish between false malware data from the discriminator and regular records from the generator.
2. Both the generator and discriminator models in the model use a hidden layer with 64 nodes.

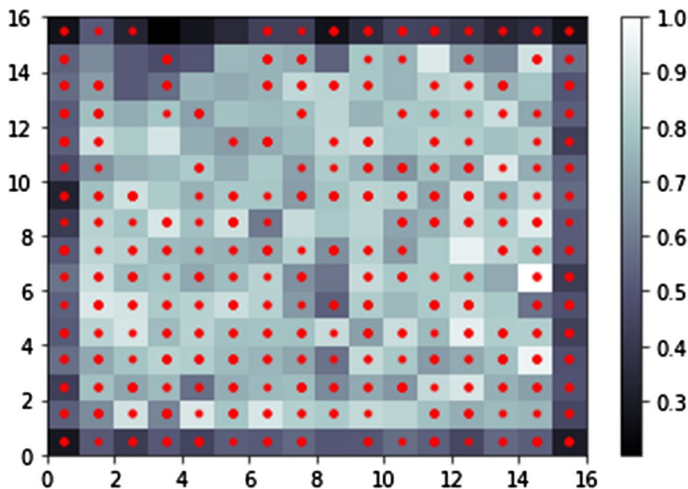


Fig. 3 SOM with Mirai winning unit in red color

Table 2 GANs model output value with execution time and 5 epochs

Model: "model_2"		
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 83)]	0
sequential (Sequential)	(None, 83)	10771
sequential_1 (Sequential)	(None, 5)	5701

Total params: 16,472
 Trainable params: 10,771
 Non-trainable params: 5,701

Train on 400500 samples, validate on 100126 samples

Epoch 1/5
 400500/400500 [=====] - 38s 94us/sample - loss:0.0624 - acc: 0.9817 - val_loss: 0.0446 - val_acc: 0.9858

Epoch 2/5
 400500/400500 [=====] - 36s 91us/sample - loss:0.0398 - acc: 0.9883 - val_loss: 0.0301 - val_acc: 0.9897

Epoch 3/5
 400500/400500 [=====] - 37s 92us/sample - loss:0.0290 - acc: 0.9933 - val_loss: 0.0213 - val_acc: 0.9957

Epoch 4/5
 400500/400500 [=====] - 29s 73us/sample - loss:0.0246 - acc: 0.9947 - val_loss: 0.0188 - val_acc: 0.9958

Epoch 5/5
 400500/400500 [=====] - 25s 63us/sample - loss:0.0223 - acc: 0.9950 - val_loss: 0.0166 - val_acc: 0.9962

Loss: 0.017877

Accuracy: 0.995829

3. When dealing with malware data, the discriminator value is near 0, while when dealing with a record of normal data, it is near 1.
4. The generator model is trained in the second stage of training. The discriminator is intended to be duped into thinking the data produced by the generator is accurate as its main goal.
5. GANs successively repeat the two stages until convergence is obtained.
6. A GAN model is created by integrating the discriminator and the generator.

5.3 Radial Basis Neural Network model

RBN may be an unsupervised, and only the two layers make up the whole model of RBN, and 'radial basis' is the utilized activation unit. The hidden unit of this model handles the calculation as well as feeds the input into the input layer [21]. Compared to the MLP, the RBFN model is more intuitive. A "model" is kept in each RBFN neuron. Each neuron calculates the Euclidean distance between its model and the input at the time when characterization is required. In general, the input data is referred to be class X if it resembles the class X model as compared to the class Y model [22]. A description of the Radial Basis Networks approach for effective DoS attack detection is also provided. If IoT malicious nodes are adequately segregated and found to be more resistant to DoS assaults, it delivers better outcomes [23] Table 3.

Table 3 Output Result of RBFN model with execution time and 5 epochs

Layer (type)	Output Shape	Param #
rbf_layer_1 (RBFLayer)	(None, 64)	5312
dense_2 (Dense)	(None,5)	325
Total params: 5,637		
Trainable params: 5,637		
Non-trainable params: 0		
Train on 400500 samples, validate on 100126 samples		
Epoch 1/5		
400500/400500 [=====] - 38s 95us/sample - loss: 0.2379 - acc: 0.9359 - val_loss: 0.1980 - val_acc: 0.9371		
Epoch 2/5		
400500/400500 [=====] - 35s 87us/sample - loss: 0.1944 - acc: 0.9360 - val_loss: 0.1821 - val_acc: 0.9371		
Epoch 3/5		
400500/400500 [=====] - 31s 77us/sample - loss: 0.1775 - acc: 0.9360 - val_loss: 0.1675 - val_acc: 0.9371		
Epoch 4/5		
400500/400500 [=====] - 31s 76us/sample - loss: 0.1678 - acc: 0.9360 - val_loss: 0.1646 - val_acc: 0.9371		
Epoch 5/5		
400500/400500 [=====] - 31s 76us/sample - loss: 0.1628 - acc: 0.9360 - val_loss: 0.1572 - val_acc: 0.9371		

Loss: 0.159412

Accuracy: 0.935050

The steps followed in RBFN Model Algorithm

1. The execution of the model is carried out much more quickly with radial basis networks than with other methods because they only make use of the "RBFLayer," a single hidden layer that uses 64 nodes.
2. The Radial Basis function, which takes into account a point's distance from the cluster's center, is the activation function in use here.
3. The basis model receives input data with 83 features and an 80:20 split between training and testing.
4. To distinguish 5 different classes 'Softmax', is employed to distinguish between the original input and the regenerated state, and the loss "categorical_crossentropy" is determined.
5. In this manner, the RBF model is created, and until convergence is attained, it attempts to identify the input identity function.

5.4 SOM (Self-Organizing Maps)

SOMs, Kohonen networks, or WTU (Winner takes all units) are additional names for this Unsupervised Learning model in which classes are clustered separately [24]. It differentiates from other clustering learning methods, such as PCA and k-means, in that it maintains the topological relationship between the connected neuron layers. Let's say we use a dataset with (x, y) size, hereby x indicates this training samples, also y indicates these column features used in each model. In the first stage, where C is the cluster number is initialized

as `size_weights`. By repeatedly going over the input file for each training model (weight vector with a small distance from the training model, such as the Euclidean distance), it updates the winning vector.

Steps followed in SOM Algorithm

1. We developed a large SOM (16, 16). For a dataset with more dimensions, SOMs are challenging. To implement SOM, weights are loaded with some random measurements and each neuron is given a particular position. Out of all the inputs utilized in this experiment, the neuron that is located closest to the data input is called the WTU.
2. It is calculated how far away the weights are from the inputs of the neurons. The winner is determined to be the neuron with the lowest "d" value [25].
3. Higher-dimensional SOM is hardly used in practice. SOMs are not practical for large datasets because of their high processing costs. Below is a map depicting our SOM categorization of MIRAI attacks, with a red marking indicating winning units. Our SOMs accurately represent the locations of neurons that have a common goal of achieving success for the same colors.

5.5 RBM Model (Restricted Boltzmann Machine)

RBM is an ANN network with two layers. The structure has two distinct layers, named as.

1. Hidden layer.
2. Visible layer.

No two neurons in the same layer may be linked to each other, although both hidden and visible layers are interconnected [26]. Additionally, every single one of such neurons in the particular model is binary. As a result, it has the name "Restricted Boltzmann machine" [27] (Table 4).

The steps followed in RBM Algorithm

1. RBM forward and backward training is possible.
2. Forward phase: From the visible layer, inputs containing 83 characteristics are combined with biases and weights before being transferred to the hidden layer. The likelihood of the sigmoid function is used to decide whether or not the node units may be fired.
3. Backward phase: The input is once again reconstructed by the visible units after the hidden units have passed them with identical weights but a different bias.

Table 4 RBM model output result with execution time and 10 iterations

[BernoulliRBM] Iteration 1, pseudo-likelihood = -4606.59, time = 16.42s
[BernoulliRBM] Iteration 2, pseudo-likelihood = -9065.59, time = 27.92s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -13744.98, time = 25.46s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -18323.45, time = 24.71s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -22794.24, time = 24.64s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -27335.72, time = 24.44s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -31734.70, time = 24.62s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -35955.17, time = 24.90s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -41456.63, time = 24.67s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -45685.65, time = 24.78s

Loss: 0.15941

Accuracy: 0.93505

4. These two passes are iteratively repeated until convergence is attained.
5. Once the RBM model is created, it is exciting to breed the amazing classification result for anomaly detection in IoT since it can train with a dataset divided into a ratio of 80:20.

5.6 DBN (Deep Belief Neural Network)

An unsupervised method known as a Deep Belief Neural Network is just many RBMs placed on top of one another [28]. The output of the first RBM becomes the contribution of the following RBMs. Deep learning algorithms are capable of learning each layer in a turn, making deep belief nets quick and yet fast. Finding the clusters in the data is quite helpful since DBN learns different levels and types of input and anticipates recreating the input [29] (Table 5).

The steps followed by the DBN algorithm

1. The IoT-23 input characteristics are used to train the first RBM. The second RBM, followed by the opposite sequential stacked RBMs, gets as input the primary RBM's output.
2. Yes, stacking the collection of RBMs prepared the model belief network. Two RBMs are layered in this model to develop this concept. The first RBM has 100 hidden units, while the second has 50 units.
3. It offers an excellent accuracy value with a low loss.

Table 5 DBN model output value with reconstruction error and 5 epochs

```
[START] Pre-training step:
Instructions for updating:
>> Epoch 1 finished      RBM Reconstruction error 110170.711261
>> Epoch 2 finished      RBM Reconstruction error 1657281.413061
>> Epoch 3 finished      RBM Reconstruction error 8021137.124861
>> Epoch 4 finished      RBM Reconstruction error 24884444.570548
>> Epoch 5 finished      RBM Reconstruction error 59707284.337118
>> Epoch 1 finished      RBM Reconstruction error 785692460318720.000000
>> Epoch 2 finished      RBM Reconstruction error 758497197162496.000000
>> Epoch 3 finished      RBM Reconstruction error 757690414399488.000000
>> Epoch 4 finished      RBM Reconstruction error 799343678324736.000000
>> Epoch 5 finished      RBM Reconstruction error 789654299213824.000000
[END] Pre-training step
Instructions for updating:
[START] Fine-tuning step:
>> Epoch 0 finished      ANN training loss 0.234188
>> Epoch 1 finished      ANN training loss 0.234211
>> Epoch 2 finished      ANN training loss 0.234156
>> Epoch 3 finished      ANN training loss 0.234134
>> Epoch 4 finished      ANN training loss 0.234140
>> Epoch 5 finished      ANN training loss 0.234134
>> Epoch 6 finished      ANN training loss 0.234133
>> Epoch 7 finished      ANN training loss 0.234133
>> Epoch 8 finished      ANN training loss 0.234135
>> Epoch 9 finished      ANN training loss 0.234137
[END] Fine-tuning step
Done.Accuracy: 0.936500 Loss: 0.06350
```

5.7 LSTM (Long Short-Term Memory)

In brief, the supervised category model’s main learning model is the LSTM, which is also well-known in natural language processing. Memory inadequacy is the main shortcoming in the RNN model [30]. LSTM cells, a unique kind of RNN, were developed to solve this problem. The LSTM is sufficiently intelligent to choose how long to hold onto old data, when to neglect and recall it, and how to link its prior memory with the current data being given to it. It does this by employing the forget gate. The gradient of the loss characteristic, on the other hand, diminishes as we flow again through time in RNN backpropagation through time (BPTT). Because of this impact, fashionable RNNs can’t be directly applied to clear up problems with long-time dependencies known as vanishing gradients. We invoke lengthy quick-time period memory (LSTM) networks which could successfully address the vanishing gradient trouble through the usage of unique cells which could maintain records. LSTM addresses the trouble of vanishing and exploding gradients through the usage of a state of a cell [31] (Table 6).

The steps followed in LSTM Algorithm

1. The procedures were followed in the same manner as with previous models, and a key feature of this model is that it only takes 3-D data, allowing us to rearrange the input as well as output data.

Table 6 LSTM model output result with execution time and 5 epochs

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 83, 32)	4352
dropout (Dropout)	(None, 83, 32)	0
lstm_1 (LSTM)	(None, 8)	1312
dense_3 (Dense)	(None, 5)	45

Total params: 5,709
 Trainable params: 5,709
 Non-trainable params: 0

Train on 400500 samples, validate on 100126 samples

Epoch 1/5
 400500/400500 [=====] - 646s 2ms/sample - loss: 0.1559 - acc: 0.9434 - val_loss: 0.0945 - val_acc: 0.9661

Epoch 2/5
 400500/400500 [=====] - 625s 2ms/sample - loss: 0.0983 - acc: 0.9657 - val_loss: 0.0651 - val_acc: 0.9803... - ETA: 7:14 - loss: 0.1132 - acc: 0.9607

Epoch 3/5
 400500/400500 [=====] - 671s 2ms/sample - loss: 0.0663 - acc: 0.9792 - val_loss: 0.0614 - val_acc: 0.9794

Epoch 4/5
 400500/400500 [=====] - 647s 2ms/sample - loss: 0.0702 - acc: 0.9784 - val_loss: 0.0476 - val_acc: 0.9855

Epoch 5/5
 400500/400500 [=====] - 688s 2ms/sample - loss: 0.0517 - acc: 0.9844 - val_loss: 0.0439 - val_acc: 0.9867

Loss: 0.05170

Accuracy: 0.98440

2. The LSTM layer generates an accurate representation of the output parameters, shape, dense layers, and LSTM layers.

5.8 RNN Model

It is a feed backward model where the output value is sent back to the existing node to remember the output from the last node. Consider it as a simple circular loop where the input node is fed from the output again. As a consequence, they can recall information from distant records and use it to forecast data [32]. The LSTM neural model addresses its major long-term memory shortage problem. To make the fitness care gadget installation offerings a smart way, RNN Networks are employed, which makes the expectation of detected data for positive sensed records without determining date and time [33] (Table 7).

The steps followed in RNN Algorithm

1. The initial input layer which is shifted to the hidden layer has weights and biases applied to the input units.
2. Each hidden layer has 128 neurons as well as a "ReLU" activation unit, which automates the unwanted features.
3. The "Adam Optimizer" is used for error reduction, and also fits the produced model into the testing and training datasets to compile and assess the accuracy score.

Table 7 RNN model output result with execution time and 5 epochs

Model: "model_5"		
Layer (type)	Output Shape	Param #
reshape_1 (Reshape)	(None, 83, 1)	0
simple_rnn (SimpleRNN)	(None, 32)	1088
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 5)	85
Total params: 1,701		
Trainable params: 1,701		
Non-trainable params: 0		
Train on 400500 samples, validate on 100126 samples		
Epoch 1/5		
400500/400500 [=====] - 238s 594us/sample - loss: 0.0742 - acc: 0.9765 - val_loss: 0.0545 - val_acc: 0.9835		
Epoch 2/5		
400500/400500 [=====] - 258s 645us/sample - loss: 0.0557 - acc: 0.9824 - val_loss: 0.0531 - val_acc: 0.9840		
Epoch 3/5		
400500/400500 [=====] - 262s 655us/sample - loss: 0.0533 - acc: 0.9834 - val_loss: 0.0497 - val_acc: 0.9837		
Epoch 4/5		
400500/400500 [=====] - 268s 669us/sample - loss: 0.0524 - acc: 0.9837 - val_loss: 0.0464 - val_acc: 0.9861		
Epoch 5/5		
400500/400500 [=====] - 269s 671us/sample - loss: 0.0516 - acc: 0.9841 - val_loss: 0.0470 - val_acc: 0.9852		

Loss: 0.048755

Accuracy: 0.984627

4. The "Sigmoid" function, which makes use of the "categorical_crossentropy" loss function, is utilized to get the multiclass classification result.
5. The IoT-23 dataset, which regularly outperforms Text Data, scored the usual execution measurement scores of RNNs.
6. By distinguishing between 5 distinct classes, the result is summed by using the summary of the simple RNN model.

5.9 CNN (Convolutional Neural Network)

Convolution is a function used by CNN. The concept is supported by the fact that just two of the resulting neuron's neural nodes are associated with, rather than each neuron being interfaced with every single resultant neuron [34] (Table 8).

The steps followed to design a CNN Algorithm

1. Similar to previous models, bias and weights are applied to input units and fed into the model's initial input layer before it is shifted to the hidden layer.

Table 8 CNN model summary with execution time and 5 epochs

Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 83, 1)	0
conv1d (Conv1D)	(None, 81, 64)	256
max_pooling1d (MaxPooling1D)	(None, 41, 64)	0
flatten (Flatten)	(None, 2624)	0
dense_5 (Dense)	(None, 32)	84000
dense_6 (Dense)	(None, 5)	165

Total params: 84,421
 Trainable params: 84,421
 Non-trainable params: 0
 Train on 400500 samples, validate on 100126 samples
 Epoch 1/5
 400500/400500 [=====] - 56s 141us/sample - loss:0.0521 - acc: 0.9848 - val_loss: 0.0299 - val_acc: 0.9889
 Epoch 2/5
 400500/400500 [=====] - 59s 147us/sample - loss:0.0259 - acc: 0.9942 - val_loss: 0.0181 - val_acc: 0.9958
 Epoch 3/5
 400500/400500 [=====] - 56s 141us/sample - loss:0.0207 - acc: 0.9952 - val_loss: 0.0171 - val_acc: 0.9954
 Epoch 4/5
 400500/400500 [=====] - 55s 136us/sample - loss:0.0184 - acc: 0.9957 - val_loss: 0.0165 - val_acc: 0.9961
 Epoch 5/5
 400500/400500 [=====] - 56s 141us/sample - loss:0.0170 - acc: 0.9960 - val_loss: 0.0137 - val_acc: 0.9965

Loss: 0.015372

Accuracy: 0.996317

2. Conv1D, a filter with 64 nodes, was utilized to model the tabular dataset to get a much higher score.
3. Maxpooling1D to reduce the dimensional space.
4. The model is fitted to the testing and training datasets using the "Adam" optimizer to minimize loss across 5 iterations, and the best accuracy score is then determined.
5. The "softmax" generates five different types of IoT attack categories: "DoS," "Mirai," "Scan," "MITM-ARP," & "Normal."

5.9.1 MLP Model (Multilayer Perceptron)

A popular network model for classification is the multilayer perceptron, in which each neuron is coupled with every other neuron in the following layers and whose specified structure is known as a perceptron [35]. The ability to identify nonlinear correlations between the input data is provided by an MLP.

The steps followed in MLP Algorithm

1. The constructed MLP model is given the whole dataset to fit the model to the input data.
2. For feature extraction, a hidden layer with size-functioning nodes is activated.
3. It is fitted like other comparable models, and the "Adam" is used for loss reduction across five iterations, as well as backpropagation to lessen errors.
4. It produces the result by classifying data into 5 different categories, including "Mirai," "DoS," "Scan," "MITM-ARP," & "Normal."
5. Two ideal charts, including the 'Learning Curve' graph represents a declining slope of loss rate and the "Model_Accuracy" graph with excellent accuracy rate (Table 9).

6 Comparative Assessment of Foremost Pros & Cons of Top 10 Deep Neural Network Techniques

Let's discover a short look at of foremost benefits and downsides of those Ten Deep studying strategies earlier than finalizing the experimented results (Table 10).

Table 9 MLP model output value with execution time and 5 epochs

```

Model: "model_7"
-----
Layer (type)      Output Shape      Param #
-----
dense_7 (Dense)   (None, 32)        2688
-----
dense_8 (Dense)   (None, 16)        528
-----
dense_9 (Dense)   (None, 5)         85
-----
Total params: 3,301
Trainable params: 3,301
Non-trainable params: 0

Train on 400500 samples, validate on 100126 samples
Epoch 1/5
400500/400500 [=====] - 29s 73us/sample - loss: 0.0633 - acc: 0.9809 -
val_loss: 0.0421 - val_acc: 0.9876
Epoch 2/5
400500/400500 [=====] - 22s 56us/sample - loss: 0.0358 - acc: 0.9899 -
val_loss: 0.0265 - val_acc: 0.9932
Epoch 3/5
400500/400500 [=====] - 19s 48us/sample - loss: 0.0248 - acc: 0.9944 -
val_loss: 0.0195 - val_acc: 0.9959
Epoch 4/5
400500/400500 [=====] - 19s 49us/sample - loss: 0.0213 - acc: 0.9951 -
val_loss: 0.0177 - val_acc: 0.9956
Epoch 5/5
400500/400500 [=====] - 19s 48us/sample - loss: 0.0196 - acc: 0.9953 -
val_loss: 0.0148 - val_acc: 0.9963
    
```

Loss: 0.017234

Accuracy: 0.996157

7 Comparative Results and Conclusion

The fact about the accuracy values, minimizing the model execution duration and loss value, and 10 AI-deep learning techniques to generate all such models is evident from the below-summarized data table. CNN is the most potent, reliable, and quick model to produce an accurate score for IoT abnormality identification and thus limit the execution time as well as loss, according to the primary supervised deep learning techniques of the best ten AI -Deep Learning models, which are followed by an MLP model in the IoT-23 true dataset that is supervised classification. The unsupervised method GANs offers excellent results, the quickest execution model, and is practical in identifying IoT attack abnormalities. The study paper that followed provided a substantial and new addition to the multi-class classification of IoT anomalies by identifying five different IoT class types and summarising the output from the top ten Deep Learning neural network models (Table 11).

Table 10 Comparison of major advantages and disadvantages of Ten Deep Learning Techniques

S. No	Advantages	Disadvantages
1	<p>MLP</p> <p>Functions admirably with the massive input file and conjointly lower input data</p> <p>Frequently applied to complicated non-linear issues for fast forecasting on trained models</p>	<p>MLP</p> <p>The correct working of the model relies on the standard of model training. Calculations are delicate and overpowering time</p>
2	<p>CNN</p> <p>Without human intervention, it shares the weights and spies on the vital features precisely.</p> <p>Authentically high delicacy in furnishing results in problem issues</p>	<p>CNN</p> <p>Absence of ability to deal with spatial connections of data</p>
3	<p>RNN</p> <p>It is utilized in time series-related forecasting problems because it continues each data on a time basis.</p> <p>It's additionally known as LSTM due to the factor to remember its former input data sources</p>	<p>RNN</p> <p>Inability to handle lengthy sequences of data and facing problems of exploding and vanishing gradient issues</p>
4	<p>LSTM</p> <p>It turned out to be so beneficial because it takes care of the issue of vanishing slopes</p>	<p>LSTM</p> <p>They need a ton of resources to get prepared for true applications</p>
5	<p>GAN</p> <p>GAN produces data that appears just like unique inputs henceforward to comprehend the input data effectively</p>	<p>GAN</p> <p>This network finds it tougher to produce textual content or audio data</p>
6	<p>Autoencoder</p> <p>Autoencoders can be prepared in an unlabeled manner, and work as an unsupervised algorithm</p>	<p>Autoencoder</p> <p>It encodes the input configuration through a bottleneck layer, there are possibilities of lacking significant aspects assuming the bottleneck is excessively restricted</p>
7	<p>RBF</p> <p>RBF is the principal adversary of MLP and it catches up with training quicker than MLP</p>	<p>RBF</p> <p>Despite the fact, that RBF is simpler to train, however, after the preparation is exceptionally delayed being utilized is slower than MLP</p>
8	<p>SOM</p> <p>The visual representation of the results is appealing</p> <p>The execution speed is much higher</p>	<p>SOM</p> <p>It is hard to understand the outcomes from mapping results.</p> <p>It is computationally contrasted with K-Means</p>

Table 10 (continued)

S. No	Advantages	Disadvantages
9	<p>RBM It's a double-layered network engineering only with two layers, Input and hidden. The intralayer connection between the layers is not necessary so the performance is much faster</p>	<p>RBM The training of the Boltzmann machine is troublesome because of the complexity that arises in calculating gradient functions</p>
10	<p>DBN It is formed by a group of Boltzmann machines and also has high-quality overall performance in classification. DBN generates stylish performance results when the quantum of data is big. DBN is bendy to future techniques</p>	<p>DBN DBN needs higher data for training purposes. DBN is high-priced to educate as it has complicated information models. It is tough to apply with the aid of using much fewer professional people</p>

Table 11 Comparison of Experimented all Deep Learning model output results with loss value and accuracy score and Execution Time for 1 Epoch in seconds

Deep learning models	Loss value	Accuracy values	Execution time for 1 epoch/sec
GANs model	0.017877	0.995829	38 s
Autoencoders neural model	0.033690	0.996269	29 s
RBM model	0.159412	0.935050	16.4 s
RBFN model	0.159412	0.935050	38 s
LSTM network	0.051700	0.984400	646 s
DBN model	0.063500	0.936500	32 s
CNN model	0.015372	0.996317	56 s
RNN model	0.048755	0.984627	238 s
MLP model	0.017234	0.996157	29 s

Acknowledgements I appreciate Dr. T. Prem Jacob, my co-author, for his continual support and encouragement as well as for reviewing the final edit of this work. Additionally, I'd like to thank the doctorate committee members of mine, respectable review members, and also the honorable EiC, 'Ramjee Prasad', 'Tim Kersjes', 'Iratxe Puebla', who have supported my research and also helped me to promote my research publications.

Funding The authors have not disclosed any funding.

Data Availability Enquiries about data availability should be directed to the authors.

Declarations

Conflict of interest The author of the article titled "The Top 10 Artificial Intelligence -Deep Neural Networks for IoT Intrusion Detection System", declares that there is no conflict of interest in this paper.

References

- Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. *Computer*, 50(7), 80–84.
- Radanliev, P. (2018). Future developments in cyber risk assessment for the internet of things. *Computers in Industry*, 102, 14–22.
- Bertino, E., & Islam, N. (2017). Botnets and internet of things security. *Computer*, 50, 76–79.
- Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of a machine and deep learning methods for the internet of things (IoT) security. *IEEE Communications Surveys & Tutorials*, 22(3), 1646–1685.
- Mirzaei, A. & Najafi Souha, A. (2021). Towards optimal configuration in MEC Neural networks: Deep learning-based optimal resource allocation. *Wireless Personal Communications*.
- Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., & Ghogho, M. (2016). Deep learning approach for network intrusion detection in software-defined networking. In *2016 international conference on wireless networks and mobile communications (WINCOM)* (pp. 258–263). IEEE.
- A Gautam, S Singh. (2021). Deep learning based object detection combined with internet of things for remote surveillance. *Wireless Personal Communications*.
- Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion*, 42, 146–157.
- Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cybersecurity. *Information*, 10(4), 122.

10. Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234, 11–26.
11. Bhardwaj, S. & Dave, M. (2022). Crypto-Preserving investigation framework for deep learning based malware attack detection for network forensics. *Wireless Personal Communications*
12. Milenkoski, A., Vieira, M., Kounev, S., Avritzer, A., & Payne, B. D. (2015). Evaluating computer intrusion detection systems: A survey of common practices. *ACM Computing Surveys (CSUR)*, 48(1), 1–41.
13. Hussain, F., Abbas, S.G., Fayyaz, U.U., Shah, G.A., Toqeer, A., & Ali, A. (2020). Towards a universal features set for IoT botnet attacks detection. arXiv preprint [arXiv:2012.00463](https://arxiv.org/abs/2012.00463).
14. Ullah, I., & Mahmoud, Q. H. (2020, May). A scheme for generating a dataset for anomalous activity detection in IoT networks. In *Canadian conference on artificial intelligence* (pp. 508–520). Springer.
15. Foley, J., Moradpoor, N., & Ochen, H. (2020). Employing a machine learning approach to detect combined internet of things attacks against two objective functions using a novel dataset. *Security and Communication Networks*, 2020.
16. Ferrag, M. A., Maglaras, L., Moschoyiannis, S., & Janicke, H. (2020). Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419.
17. Saito, Y., Benjebbour, A., Kishiyama, Y., & Nakamura, T. (2013). System level performance evaluation of downlink non-orthogonal multiple access (NOMA), In *Proceeding IEEE annual symposium on personal, indoor and mobile radio communications (PIMRC)* (pp. 611–615).
18. Sakurada, M., & Yairi, T. (2014). Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis* (pp. 4–11).
19. Zenati, H., Foo, C. S., Lecouat, B., Manek, G., & Chandrasekhar, V. R. (2018). Efficient gan-based anomaly detection. arXiv preprint [arXiv:1802.06222](https://arxiv.org/abs/1802.06222).
20. Di Mattia, F., Galeone, P., De Simoni, M., & Ghelfi, E. (2019). A survey on gans for anomaly detection. arXiv preprint [arXiv:1906.11632](https://arxiv.org/abs/1906.11632).
21. Orr, M. J. (1996). Introduction to radial basis function networks.
22. Rapaka, A., Novokhodko, A., & Wunsch, D. (2003). Intrusion detection using radial basis function network on sequences of system calls. In *Proceedings of the international joint conference on neural networks, 2003*. (Vol. 3, pp. 1820–1825). IEEE.
23. Premkumar, M., & Sundararajan, T.V.P. (2021). Defense countermeasures for DoS attacks in WSNs using deep radial basis networks. *Wireless Personal Communications*
24. Cottrell, M., Fort, J. C., & Pagès, G. (1998). Theoretical aspects of the SOM algorithm. *Neurocomputing*, 21(1–3), 119–138.
25. Tian, J., Azarian, M. H., & Pecht, M. (2014). Anomaly detection using a self-organizing maps-based k-nearest neighbor algorithm. In *Proceedings of the european conference of the prognostics and health management society* (pp. 1–9). Citeseer.
26. Montúfar, G. (2016). Restricted Boltzmann machines: Introduction and review. In *Information Geometry and Its Applications IV* (pp. 75–115). Springer.
27. Pumsirirat, A., & Yan, L. (2018). Credit card fraud detection using deep learning based on auto-encoder and restricted Boltzmann machine. *International Journal of advanced computer science and applications*, 9(1), 18–25.
28. Abdel-Zaher, A. M., & Eldeib, A. M. (2016). Breast cancer classification using deep belief networks. *Expert Systems with Applications*, 46, 139–144.
29. Van, N. T., & Thinh, T. N. (2017). An anomaly-based network intrusion detection system using deep learning. In the *2017 International conference on system science and engineering (ICSSE)* (pp. 210–214). IEEE.
30. Zhou, X., Hu, Y., Liang, W., Ma, J., & Jin, Q. (2020). Variational LSTM enhanced anomaly detection for industrial big data. *IEEE Transactions on Industrial Informatics*, 17(5), 3469–3477.
31. Dave, A., Kuhar, S., and Chopra, R. (2021). Deep learning-based semi-blind tracking for aging wireless communication channels. *Wireless Personal Communications*
32. He, Y., Chen, R., Li, X., Hao, C., Liu, S., Zhang, G., & Jiang, B. (2020). Online at-risk student identification using rnn-gru joint neural networks. *Information*, 11(10), 474.
33. Radhika, R., Bhuvaneshwari, A., & Kalpana, G. (2022). An intelligent semanticification rules enabled user-specific healthcare framework using IoT and deep learning techniques, *Wireless Personal Communications*.
34. Pyakillya, B., Kazachenko, N., & Mikhailovsky, N. (2017). Deep learning for ECG classification. *Journal of physics: conference series*, 913(1), 012004.

35. Roy, S.S., Mallik, A., Gulati, R., Obaidat, M.S., & Krishna, P.V. (2017). Deep learning-based artificial neural network approach for intrusion detection. In *International conference on mathematics and computing* (pp. 44–53). Springer.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



V. Kanimozhi completed M.E, Computer Engineering, currently in a researcher position in Artificial Intelligence for more than 4.5 years in Sathyabama University which is a Tier I university, and also working as a Data Scientist, have good proficiency in Python Libraries (Numpy-Scipy, Pandas), Machine Learning, Deep learning AI algorithms, Artificial Intelligence Neural Networks, Tensorflow2, Python Implementation framework Jupyter Notebook and Spyder3. On the verge of Ph.D. (Data Science) in Big Data Analytics and Anomaly Detection of Large Scale Network-Based security and IoT network security using Python 3, Anaconda. Published both National and International journals in Artificial Intelligence, Data Science, Machine Learning and served as an editorial board member and reviewer of International reputed journals



T. Prem Jacob received the B.E degree in Computer Science and Engineering from C.S.I Institute of Technology, Manonmaniam Sundaranar University, Nagercoil, India in 2004, M.E degree in Computer Science and Engineering from Sathyabama University, Chennai, Indian 2006 and Ph.D. degree from Sathyabama University, Chennai, India. He works currently as an Associate Professor of Computer Science and Engineering in Sathyabama Institute of Science and Technology Chennai. He has participated and presented many Research Papers in International and National Conferences. The area of interests includes Software Engineering, Data mining & Data warehouse, Networks, Machine Learning, Deep Learning, and Cloud computing. Published more than 50 research articles in reputed journals.

Authors and Affiliations

V. Kanimozhi¹  · T. Prem Jacob¹

¹ Sathyabama Institute of Science and Technology, Chennai, Tamilnadu, India