Journal of Theoretical and Applied Information Technology
15th July 2023. Vol.101. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A TABTRANSFORMER BASED MODEL FOR DETECTING BOTNET-ATTACKS ON INTERNET OF THINGS USING DEEP LEARNING

ARCHANA KALIDINDI[1], MAHESH BABU ARRAMA[2]

[1]Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Hyderabad ,Telangana, India

[2]Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Hyderabad ,Telangana, India

E-mail: [1]archana.buddaraju@gmail.com, [2]maheshbabu.a@klh.edu.in

ABSTRACT

The Internet-of-Things (IoT) has revolutionized with the increase in data and serves as a link for the ecosystem of various tools, actuators, and sensors, and this is one of the disruptive technologies. IoT enjoys numerous advantages by accelerating communication between various smart objects around us Hence, IoT has become not only an essential part by serving as a medium for growth in advancing technology; the most crucial part is protecting the data from malicious attacks. To resolve Intrusion Attacks ranging from cyber to rule-based, the new advancing AI plays a significant role. Hence, it is well established in the literature that such attacks can be solved by using Machine Learning Algorithms. We analysed various Machine Learning and Deep Learning frameworks for tackling these attacks on the standard dataset N-BaIoT. This extensive analysis has confirmed that the TabTransformer model with SGD, Adam, and Avg & Sub optimizers has exhibited excellent performance, achieving at least 92.33% accuracy. It was observed that deep learning approaches such as Conv-Net or LSTM-based methods have come close to achieving similar results. Furthermore, we have compared the accuracy of our classification method with that of other studies, and the TabTransformer has demonstrated superior performance. This is the first study to conduct a series of experiments on the N-BaIoT dataset using a range of traditional machine learning and deep learning techniques. Our study has achieved state-of-the-art accuracy of 92.330% for the Provision PT-737E device using the TabTransformer model with Adam optimizer.

Keywords: *N-BaIoT; Transformer; Internet of Things; Machine Learning; Deep Learning.*

1. INTRODUCTION

The Internet-of-Things (IoT) has revolutionized with the increase in data and serves as a link for the ecosystem of various tools, actuators, and sensors, and this is one of the disruptive technologies. IoT enjoys numerous advantages by accelerating communication between various smart objects around us. Hence, IoT has become not only an essential part by serving as a medium for growth in advancing technology, but the most crucial part is also protecting the data from malicious attacks. To resolve Intrusion Attacks ranging from cyber to rule-based, the new advancing AI plays a significant role. Several security solutions, including protection, authentication, and detection, have been developed for the IoT. Using machine learning (ML) techniques in conjunction with the IoT may be able to address privacy and security concerns. It is crucial to choose the appropriate environment for automated decision-making, such as the thin layer, the cloud, or the fog. The IoT decision-making process is decelerated when all ML judgments are made in the cloud, though. Due to a lack of facilities, such as energy, bandwidth, and computation, ML solutions are difficult to apply with other layers, such as the thin or fog layer.

Botnets are highly evasive and often remain undetected due to their ability to propagate and update their behaviour autonomously and their capacity to reside on devices without negatively impacting performance. This makes it immensely challenging to identify botnets from IoT device traffic, making it essential for security measures to be taken [1]. Misuse-based detection is a method used to track network activity and look for similarities to

known attack signatures. It is widely used in public detection systems but is limited in its ability to recognize unknown attacks. Anomaly detection systems are more adept at identifying new threats, but they also have a high tendency to produce false-positive alarms. Additionally, they are difficult to implement in IoT environments due to the intricate nature of devices. Machine learning-based detection may be the most viable detection mechanism, as it can distinguish between the features of various attacks. Even though many studies have been conducted using machine learning methods, they are often based on outdated datasets, such as KDDCUP 99 and KDD NSL. These are insufficient for accurately addressing modern IoT attack records. N-BaIoT [2], a more recent dataset, which contains ten attack classes and one benign class, can be utilized to build a detection model because it was collected from IoT devices running malicious software like Gafgyt and Mirai.

The proliferation of BASHLITE and Mirai [3] attacks is similar to distributed denial-of-service (DDoS) attacks and has become increasingly widespread across connected devices. In accordance with several reports [4-6], Owari, Mirai, and BASHLITE are types of botnet attacks, which are used to control and manipulate devices connected to the Internet by exploiting command and control (C&C) networks [7]. These botnet assaults have demonstrated a capacity for rapidly spreading throughout the Internet, making them particularly menacing to the IoT infrastructure, due to its susceptibility to vulnerabilities and known authentication protocols. Such was the case in 2016 when Mirai managed to infect over 2.5 million devices. In light of these facts, there are considerable gaps in existing solutions for safeguarding Internet of Things (IoT) devices from botnet attacks. The intrusion detection system (IDS) is one such approach, utilizing artificial intelligence to identify novel botnet attack patterns. IDS can be further divided into two primary methods; anomaly and misuse that are signature-based. Examples of prominent IDSs include Suricata [8] and Snort [9].

At present, AI methods are utilized to ascertain malicious botnet attacks with greater accuracy. This technology can even detect discrepancies between methods of attacks. One of the impediments faced by security protocols in handling IoT attacks is that hackers make slight modifications to previous attacks that are undetectable to security solutions. To prevent any potential threats to the IoT environment, developers and researchers have integrated AI technologies to assess network traffic [10]. For this purpose, deep learning and machine learning have been integrated into security systems to identify attacks proficiently.

2. LITERATURE REVIEW

The authors [1] introduce a system for detecting IoT botnets using machine learning. The system relies on two kinds of features: static, which come from packet headers, and dynamic, which come from payloads. The data is then classified into either normal or anomalous classes using Support Vector Machine and Random Forest algorithms. According to their experiments, the system can accurately detect botnets with a 99% success rate. Furthermore, the system can avoid false alarms. The system is also robust against false alarms and provides good scalability. [11] In this study, the authors introduced a network-based approach designed to detect botnet attacks on IoT devices. It utilizes advanced deep autoencoders to recognize malicious network behavior associated with botnet attacks. The performance of the system is evaluated on two different IoT datasets, where it attains an average detection accuracy of 91.2% while maintaining a low false positive rate of 4.6%. [12] The study proposed a hybrid deep learning system named CNN-LSTM to identify botnet attacks on nine different commercial IoT devices. The system comprised a convolutional neural network and long short-term memory, and it successfully detected malicious activities with high accuracy. The accuracies achieved by the system were up to 90.88% for doorbells, 88.53% for thermostats, and 89.64% for security cameras [13]. A deep autoencoding strategy to detect botnets on the Internet of Things. This approach incorporates pre-processing, feature extraction, classification, and visualization into one step by utilizing deep autoencoding. The paper explores the capability of this technique to detect botnets on the Internet of Things and assesses its efficiency on the N-BaIoT dataset. By utilizing the all-in-one encoding method, the authors achieved an F1 score of 99.1%. Furthermore, they achieved an F1 score of 99.9% for the Provision PT-737E security camera device by utilizing a distinct encoding method [14]. The authors presented a study on how machine learning techniques can be employed to identify botnet attacks in an IoT setting. The suggested design employed the N-BaIoT dataset and CART algorithm. The study findings indicated that the CART classifier was more effective than the Naïve Bayes classifier, with a detection accuracy of up to 99% overall [15]. In this study, unsupervised machine learning techniques for anomaly detection compare different models' performance on different datasets. The OC-SVM model achieved the best performance, with an accuracy score of 85%, while

the K-means model was the fastest, completing inference on a single sample in just 0.04 milliseconds, larger datasets provide the largest room for improvement for LSTM models [16]. The more recent research has higher levels of accuracy compared to previous ones by using SVM algorithms. The ROC scores obtained from these studies were 89% and 55%. However, these algorithms took too long to predict, which made them unsuitable for real-time detection [17]. The paper presents an approach called LGBA-NN that aims to identify botnet attacks on nine different commercial IoT devices using neural networks. The proposed algorithm outperformed other advanced methods like Particle-Swarm-Optimization (PSO-NN) and BA-NN, achieving an accuracy of 90% on NBaIoT data [18]. Using the NBaIoT dataset, the authors conducted a study employing various types of deep neural networks, including CNN, RNN, and LSTM. The results showed that LSTM provided an accuracy of 62%, RNN produced an accuracy of 41%, and CNN produced an accuracy of 91%.

3. DATASET DESCRIPTION

The N-BaIoT dataset is a cutting-edge and highly sophisticated collection of data poised to revolutionize Internet of Things (IoT) research. Comprised of a vast array of sensor readings and other crucial metrics, this dataset offers a comprehensive and detailed snapshot of the state of IoT devices and their behavior in real-world environments. The present study comprehensively explores the highly advanced and intricate N-BaIoT dataset, which boasts an impressive 7,062,606 records of malevolent and innocuous network traffic, gathered from a simulated organizational environment. This expansive dataset comprises nine diverse IoT devices connected through Wi-Fi to several access points and a central switch, with activity being monitored via the Wireshark4 tool. In particular, the dataset includes two botnets which are Mirai and BASHLITE [19-20]. The botnets have been specifically engineered to seek out and infect vulnerable IoT devices. Notably, this dataset comprises an assortment of five BASHLITE attacks, including Scan, Junk, Flooding (TCP/UDP), and COMBO, as well as five Mirai attacks, such as Scan, Ack, Syn, UDP Flooding, and UDP Plain. Each device's data is presented in CSV files, which have 115 dimensions or instances for each device (shown in figure 1), and also it is subcategorized into an attack type. Crucially, each data point is characterized by 115 distinct features, including 23 statistical features and 5 values of decay factor, and it is designated as λ. Having these class labels, one can determine their predictions based on the CSV filename and depicting a TCP attack in either a benign or a malignant one. Further, this dataset also concentrates on botnet infection at its terminal stage. Here, IoT bots initiate attacks, and one can analyze them according to their choice of algorithm. Hence, the N-BaIoT dataset represents a significant contribution to the field of IoT research and is poised to enable more comprehensive and in-depth insights into the behavior of IoT devices. It could aid in determining the threats in real-world settings using emerging AI.

According to the authors of [10], it is never sufficient just only to determine the initial stages of infection but, a lot more can be inferred from it.

Device Type	Device Model Name
Doorbell	Danmini Ennio
Thermostat	Ecobee
Baby monitor	Philips B120N/10
Security camera	Provision PT-737E Provision PT-838 SimpleHome XCS7-1002-WHT SimpleHome XCS7-1003-WHT
Webcam	Samsung SNH 1011 N

Figure. 1 Devices used in the N-BaIoT Dataset.

4. PROPOSED METHODOLOGY (FRAMEWORK)

Our framework (Figure 2) incorporates this IoT dataset for training models, and we provide an opportunity to even detect using those models. We have chosen the Provision PT-737E security camera device, where 10 attack samples are collected from the N-BaIoT. The dataset's size and the number of samples are obviously for a certain device type are shown in Table 1.

Table 1. Different botnets and type of attacks

Botnet	Type of Attack	Security Camera
	Benign	62,154
Bashlite	Combo	61,380
	Junk	30,898
	Scan	29,297
	TCP	104,510
	UDP	104,011
Mirai	ACK	60,554
	Scan	96,781
	Syn	65,746
	UDP	156,284
	UDP Plain	56,681
Total		**828,296**

Now we'll further move into each section in detail and for readers' ease, we have dived into different stages. In Section 4.1, we will discuss the machine learning models which are commonly used in most of the previous work. Section 4.2, firstly will produce

some deep learning methods, and not to misguide the readers we have provided all the hyperparameters along with the architectural details Also, here we will see how their performance is considered Next in section 4.3, we will move to TabTransformer model with varying the optimization techniques. Here, we include both basic SGD, Adam and Avg. and Sub. Based Optimizer and prove their significance in detecting the botnet attacks. Then the results and conclusion of the paper is summarized.

.4.1 Machine Learning Models

Machine learning (ML) [21-22] models are computer algorithms that are trained to learn patterns and make predictions based on data. These models are an integral part of modern data science and are used in a wide range of applications, including image recognition, speech recognition, natural language processing, and predictive analytics. The most common types of machine learning models are supervised [23], unsupervised [24], and reinforcement learning [25] models. In supervised learning, the model is trained on labeled data and learns to make predictions on new data based on the patterns it has learned. In unsupervised learning, the model is trained on unlabeled data and discovers patterns and relationships on its own. Reinforcement learning is a type of machine learning that involves the model learning to make decisions based on feedback from its environment. Below are some traditional ML which are trained and tested on the dataset.

4.1.1 Logistic regression

Logistic regression (LR) is important for data analysis and predictive modelling and has shown its productivity for utmost all the classification tasks. This is also implied in medical research, credit scoring, and insurance. Logistic regression [26] is particularly useful when the dependent variable is binary, and the independent variables are categorical or continuous. Due to its simplicity in handling huge datasets, resistance to outliers, and lack of presumptions on the distribution of the data, it is an extensively used technique. But its applicability is not limited it can also be used to predict the probability of an event occurring and can be used to assess the impact of different independent variables on the dependent variable. Thus, for this work, we implied LR on all the coefficients that are obtained for the complete feature-set (attributes) of the N-BaIoT dataset, and this has implied the identification of which features are significant for the benign class. By sorting the coefficients in descending order, the most important features can be identified. Still, the LR model did not perform well on the Provision PT-737E surveillance camera, and just gave us an accuracy score of 79.07% and which we have reported in table 2.

4.1.2 Linear discriminative analysis

Linear Discriminant Analysis (LDA) [27] is a linear machine learning model implied in a supervised setting. its utility is seen both in classification and dimensionality reduction. LDA is commonly applied to understand anomaly patterns as its ability to identify outliers in data by looking at the statistical spread of the data points and their relative distance to the class boundaries. This makes it an ideal tool for detecting anomalies in data sets with many features. It can identify outliers that don't belong to any class or points that are quite abnormal compared to the other data points which do belong to the same class. The purpose of using LDA on an N-BaIoT dataset is to identify the most important features that can be used to accurately classify objects or events. To make this happen we need to find the various combinations linearly to discriminate each featured pattern into a unique class. But, with our observation, we have seen that LDA did not improve in accuracy compared to LR but, it's better compared to naïve bayes.

4.1.3 Naïve Bayes

The Naïve-Bayes (NB) [28] is a fundamental probabilistic model used in machine learning for classifying patterns like spam text. It also works better on continuous data using the gaussian naïve bayes algorithm. Even ND works with a supervision principle for learning representations with a probability of certain distribution relying on a definite class and thus it categorizes the data into different classes. NB assumes that the representations of the feature space for a certain kind of data are completely independent of each other and thus aid to simplify the calculation of the probability of each class. This makes the algorithm quick and effective and thus can be used for large datasets without any additional tuning. NB is relatively simple to understand and implement, which makes it a popular choice for many machine-learning applications. But the performance of the NB algorithm is the lowest for our dataset and it achieved an accuracy score of 64.71%. Also, the NB algorithm can be used in many different scenarios, such as email filtering, text classification, sentiment analysis, and more.

Table 2. Detection of accuracies in fundamental ML methods.

Method	Accuracy
Logistic Regression	79.07
Linear Discriminative Analysis	68.48
Naïve Bayes	64.71
SVM	79.84

4.1.4 Support vector machines

Support Vector Machines or SVMs are certain kernel-based algorithms in ML that introduced the novel concept of supporting hyperplanes and with their versatility, it has broader application ins supervised learning both for solving classification and regression problems. It works by plotting data points in an n-dimensional space and then drawing a hyperplane (a line or plane) that best separates the data points into different categories or classes. The best hyperplane is determined by finding the largest margin between the two classes of data. The data points which are parallel and close to the hyperplane are certainly called 'support vectors' and they eventually determine the hyperplane. These 'support vectors' not only support the decision boundary (the hyperplane) but also, maximize the distance between the two unique classes from the data points. During this process, they help is reducing the misclassification and improve the discriminability between two different classes. In addition, SVM algorithms can also be used to perform non-linear classification by using the kernel trick. SVM is also not the best model for the N-BaIoT data because SVM has Poor performance with overlapping classes [29]. In our case, it has achieved an accuracy score of 79.84 shown in Table 3.

4.2 Deep Learning Models

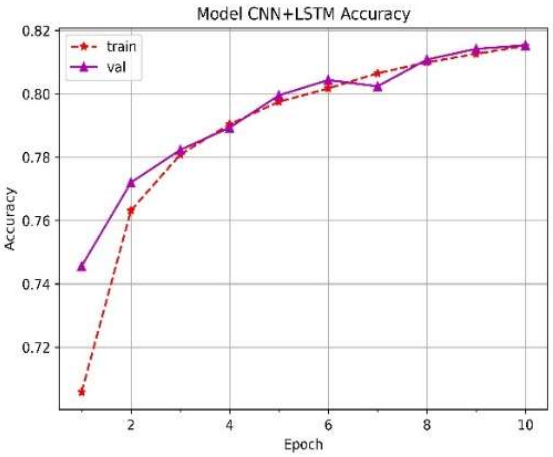
Deep learning models [30-31] are a type of machine learning algorithms that involve the use of neural networks with multiple layers. These models are particularly well-suited for solving complex problems that require high levels of accuracy and precision, such as image recognition, speech recognition, and natural language processing. Deep learning models are able to automatically learn hierarchical representations of data, which enables them to extract more meaningful features from input data than traditional machine learning models. These models are typically trained using large datasets and can require significant computational resources. Despite their complexity, deep learning models have become increasingly popular due to their superior performance in many real-world applications, such as autonomous vehicles, medical imaging, and recommender systems. As research in deep learning continues to advance, it is likely that these models will play an even larger role in shaping the future of artificial intelligence.

4.2.1 Convolutional neural networks

In this section, we describe Convolutional Neural Networks (CNNs) or also termed as ConvNets. These CNNs are a certain kind in neural networks where they imply conv-layers to extract the patterns and propagate the features from one layer to another. The conv-layer is followed by a pooling layer which reduces the size of the feature map while preserving important features. The output of the pooling layer is fed into a fully connected layer which classifies the features. The output of the fully connected layer is then fed into a layer that calculates the probabilities of multiple classes using a multi-sigmoid or a SoftMax function and then assigns a probability to each class and predicts the final output of the CNN [32]. The parameters of a convolutional layer include the stride, padding, kernel size, and activation function.

So, we found that 1D CNN can fit into the learning of features for the N-BaIoT dataset. Thus, these 1DCNNs help to detect and classify various activities of daily living (ADLs). This can be done by extracting features from the sensor data and then using a 1D CNN to classify the activities. The 1D CNN learns the representations from the features of the sensor associated with each activity, allowing it to accurately classify the activities. Additionally, the 1D CNN can be used to detect anomalies in the sensor data, which can be used to alert the user of any potential problems.

(a)



(b)

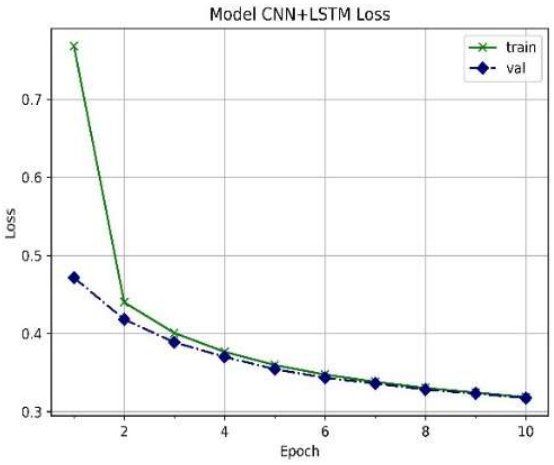
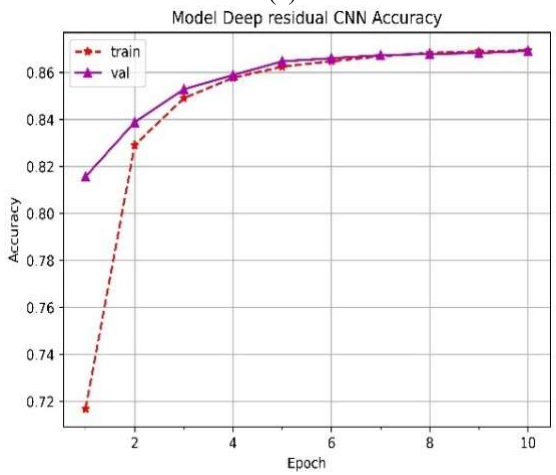


Figure. 3 (a) illustrates the CNN+LSTM models' accuracy, and (b) shows a loss calculated per epoch

We applied a 10-layer 1D CNN using the ReLU activation function. We fed 512 numbers of samples as a batch. The optimizations are carried out using the Adam optimizer and initialized the learning with a rate of 10^{-3}. To minimize the loss, we used the cross-entropy objective function. The model achieved an accuracy score of 81.53%, as shown in Table 3. The learning of the 1D CNN for the N-BaIoT data set can be understood from the accuracy and loss calculated per epoch which is visualized in figure 4.

(a)



(b)

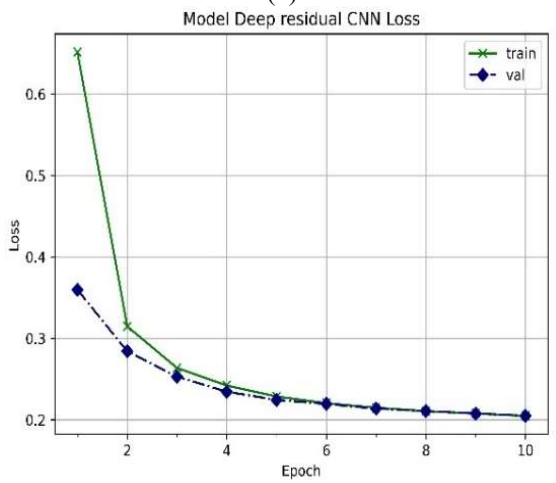


Figure 4 (a) illustrates the Deep Residual CNN (1D CNN) models' accuracy, and (b) shows the loss calculated per epoch.

Table 3. The accuracy scores for the standard deep learning Models

Model	Accuracy
ANN	46.36
CNN + LSTM	81.53
Deep Residual CNN	86.92

4.2.2 Long short-term memory

LSTM (Long Short-Term Memory) [33] is a type of artificial recurrent neural network (RNN) architecture that is designed to handle the problem of vanishing gradients, which occurs when training RNNs on long sequences. LSTM networks use specialized memory cells and gates to selectively remember and forget information over time, allowing them to effectively capture long-term dependencies in sequential data. They are particularly well-suited for tasks such as speech recognition, language translation, and time-series forecasting. The basic building block of an LSTM network is the LSTM cell, which contains three main components: a memory cell that stores information, an input gate that controls the flow of new information into the cell, and an output gate that controls the flow of information out of the cell. By adjusting the weights and biases of these components during training, an LSTM network can learn to selectively retain or discard information over time.

$$\textit{Input: } x_t = \sigma(W_x.[h_{t-1}, x_t] + b_x) \quad (1)$$

$$\textit{Forget Gate: } i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \quad (2)$$

$$\textit{Output Gate: } y_t = \sigma\big(W_y.[h_{t-1}, x_t] + b_y\big) \quad (3)$$

$$\textit{Memory Cell: } m_t = i_t.m_{t-1} + x_t.tanh(W_m.[h_{t-1}, x_t] + b_m) \quad (4)$$

$$h_t = y_t.\tanh(m_t) \quad (5)$$

Where σ is the sigmoid activation function W_x, W_i, W_y, W_m are weight matrices, b_x, b_i,b_y, b_m are bias vectors and h_t is standardized as a hidden state at definite time t.

We incorporated multiple 1DCNNs and 2 LSTMs with similar settings that are used for ConvNets. This is done to provide a broad-level comparison in a fair perspective. It could not be fair to improve certain results of the algorithm and mention its performance is bad for incomparable parameters. As it is fundamentally important in deep learning to provide appropriate hyperparameters for better learning of representations. Thus, maintaining those standard hyperparameters for the LSTM-based architecture we obtain an accuracy score of 86.92%, which is

higher than that attained from CNNs, yet not the best possible result.

4.3 TabTransformer

In this paper, we have used TabTransformer [34] to classify the type of attacks on the N-BaIoT dataset. TabTransformer is a modified version of the Transformer architecture [35] designed specifically for tabular data. The main idea behind TabTransformer is to replace the self-attention layers in the Transformer with cross-attention layers that enable the model to attend to both the row and column features in the input data. This allows the model to capture the relationships between the different features in the tabular data and to learn feature interactions. TabTransformer takes as input a tabular dataset with m rows and n columns, where each column represents a feature, and each row represents a sample.

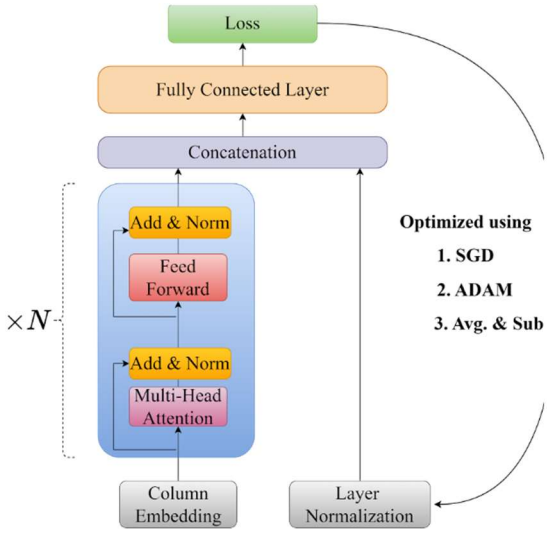


Figure 5. Architecture using optimization methods

Figure 5 illustrates the TabTransformer models' Architecture using various optimization methods that are SGD, Adam and Avg and Sub. Optimizer. Compared to standard transformer TabTransformer works best for the tabular data and also, I has two special embedding characteristics such as column and layer embedding's. Also, the number of transformer units can be scaled based on the depth of the dataset and thus this can be used significantly to extract underlying features from very large databases with millions of instances. Also, the Feed forward layer and fully connected layers are multi-layer perceptions used inherently. But, in order to distinguish the hierarchical structure where the layers have been used, we have separated with two different names and added different colour pallet to each of the components residing the TabTransformer Architecture.

The model first embeds the input data using a combination of categorical embeddings for the categorical features and continuous embeddings for the numerical features. The embeddings are then fed into a series of cross-attention layers, which compute attention weights for each feature based on the context of the other features in the input. The cross-attention layers in TabTransformer are similar to the self-attention layers in the Transformer, but with a few key differences. In the self-attention layer, the input is transformed into three different vectors, namely the query, key, and value vectors, and the attention weights are computed by taking the dot product between the query and key vectors.

We compute the row vectors Q_r, K_r, and V_r and the column vectors Q_c, K_c, and V_c by applying different linear transformations to the concatenated input X:

For row vectors $Q_r = * W_{rq}$, $K_r = X * W_{rk}$, $V_r = X * W_{rv}$ (6)

For column vectors $Q_c = * W_{cq}$, $K_c = X * W_{ck}$, $V_c = X * W_{cv}$(7)

We compute the row attention weights for each column feature by taking the dot product between the row query vector Q_r and the column key vector K_c:

$$A_r = SoftMax(Q_r * K_c^T/\sqrt{d_k}) \quad (8)$$

We compute the column attention weights for each row feature by taking the dot product between the column query vector Qc and the row key vector Kr:

$$A_c = SoftMax(Q_c * K_r^T/\sqrt{d_k}) \quad (9)$$

In the cross-attention layer, there are two types of input vectors, row vectors, and column vectors, and the attention weights are computed separately for each type of vector. The row vectors are used to attend to the column features, while the column vectors are used to attend to the row features. After the cross-attention layers, the model aggregates the attention-weighted features to compute a single vector for each row and each column. These row and column vectors are then concatenated and passed through a series of feedforward layers to produce the final output.

We have implemented various optimization techniques on the TabTransformer model. We have carefully selected the most suitable optimization algorithms to achieve the best possible outcomes. In order to enhance the model's performance, we have utilized a range of advanced techniques such as improved initialization methods, customized batch

sizes, and fine-tuned learning rates. Below, we have provided a detailed description of the optimization methods employed during the training process, which have been carefully selected to ensure that the model performs to the highest possible standard.

4.3.1 Stochastic gradient descent

Stochastic Gradient Descent (SGD) is a fundamental optimization algorithm, and its significance is implied both in machine and deep learning. It optimizes the loss function by updating the model parameters in the direction of the negative gradient of the loss with respect to the assigned parameters. Here is a detailed breakdown of how the SGD [36] optimizer works:

Initialization: The SGD optimizer starts by initializing the model parameters with some random values. These parameters will be updated during the training process to minimize the loss function.

Batch Selection: the optimizer selects a random data point in batches from the training set and this batch is generally a subset of the entire training set and it's considered due to computational constraints.

Forward Propagation: The selected set of batches for the data samples are then given as input to the model to obtain a set of predicted outputs related to the specified dataset.

Loss: At this stage, the predicted values or the outputs are then compared to the underlying ground true values for the selected batch of data points. This comparison produces certain loss values, and they are averaged for the total samples via a batch. Now, this obtained loss represents the model's deviation from the ground truth of the data points.

Backward Propagation: This is also known as back-prop and now, the optimizer calculates the set of gradients of the loss with respect to each of the model neurons using the chain rule. These accumulated gradients depict the way the model weights and in which the loss has to decreasing the most to update better weights.

Parameter Update: Finally, the optimizer updates each parameter of the model by subtracting a fraction of the gradient of the loss with respect to that parameter from its current value. This fraction is called the learning rate and determines how big the parameter updates are. The update equation is as follows,

new_parameter_value=old_parameter_value-learning_rate * gradient

For a certain fixed number of epochs this process is iterated, where individual epoch consists of numerous iterations over randomly collect the data-points from batches. The main role of the SGD-optimizer is to find the set of model parameters that minimizes the loss function over the entire training set. SGD is a simple and efficient optimization algorithm that can be easily implemented and scaled to large datasets. However, it can sometimes converge slowly or get stuck in local minima. To address these issues, several variants of SGD have been developed, such as momentum, AdaGrad, and Adam. Of this Adam has a significant reach with its property to converge at a faster scale.

4.3.2 Adaptive moment estimation

Adam (Adaptive Moment Estimation) is an optimization algorithm commonly used for training deep neural networks. It combines the ideas of both momentum and RMSprop [37] and is considered to be an extension of stochastic gradient descent (SGD) [38] with adaptive learning rates. Here is a detailed breakdown of how the Adam optimizer works:

Now, for Adam the steps involved in SGD such as, *Initialization, Batch Selection*, *Forward Propagation*, *Loss Calculation*, and *Backward Propagation* remains same. The new steps that are specifically involved in the Adam are detailed below.

Moment Estimation: Adam calculates the first and second moment estimates of the gradients using the following equations:

First Moment Estimate: $m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t$ (10)

Second Moment Estimate: $v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2$ where m_t and v_t are (11)

These momentum updates i.e., first and second moment estimates at each time step t, β_1 and β_2 are the decay rates for the first and second moment estimates, and g_t is the gradient at time step t.

Learning Rate Calculation: Adam calculates the effective learning rate for each parameter using the following equation: effective_learning_rate = learning_rate * sqrt(1 - β_2^t) / (1 - β_1^t) where t is the current time step, learning_rate is the user-defined learning rate, and β_1 and β_2 are the decay rates for the first and second moment estimates.

Parameter Update: Finally, Adam updates these parameters of the model by subtracting the effective learning rate times the first moment estimate divided by the square root of the second moment estimate from its current value. The update equation is as follows:

new_parameter_value=old_parameter_value-effective_learning_rate * m_t/ (sqrt (v_t) + epsilon) where epsilon is a small constant to prevent division by zero.

This process is repeated for multiple epochs, where each epoch consists of multiple iterations over

randomly selected batches of data points. The goal of the optimizer is to find the set of model parameters that minimizes the loss function over the entire training set. Adam is an efficient and effective optimization algorithm that can adaptively adjust its learning rate based on the gradient information. It has been shown to converge faster and achieve better performance than traditional SGD, particularly on large and complex datasets.

4.3.3 Average and subtraction-based optimizer

The average and subtraction-based optimizer is a type of optimization algorithm used in machine learning to find the optimal values for the parameters of a model. The algorithm works by keeping track of a running average of the gradients of the model with respect to each parameter [39]. This running average is then subtracted from the current gradient at each step of the optimization process. The idea behind this is to smooth out the gradient estimates and make the optimization process more stable. The basic steps of the algorithm are as follows:

- Initialize the running average for each parameter to zero.
- Compute the gradient of the model with respect to each parameter.
- Update the running average for each parameter using the current gradient and a smoothing factor (usually a value between 0 and 1).
- Subtract the running average from the current gradient for each parameter and use the resulting value to update the parameter.
- Repeat steps 2-4 until the optimization process converges.

The specific implementation details of the average and subtraction based optimizer can vary depending on the specific variant of the algorithm being used. One common variant is the AdaGrad algorithm [40], which uses a different smoothing factor for each parameter based on the historical gradient information for that parameter. Overall, the average and subtraction based optimizer is a popular choice for optimizing machine learning models due to its effectiveness and ease of implementation.

The performance of TabTransformers was enhanced by leveraging a range of optimizers, which yielded superior outcomes compared to both machine learning and deep learning models as depicted in table 4. The results section provides the detailed evaluation of the model's performance under diverse hyperparameter configurations, featuring comprehensive accuracy scores and data visualizations.

5. RESULTS

We have evaluated our performance of the data on TabTransformer with three different optimizers as previously mentioned (the accuracy results are shown in Table 5). To describe the results of the proposed models let me first briefly explain the parameters that were tuned in order to achieve such significant performance. The number of heads used in the attention mechanism is a key architecture parameter, and in this paper, we experiment with values of 1, 2, 4, and 8 heads and found that 4 to be optimal to achieve good accuracy. In addition to the attention heads, we use a feedforward neural network with 3 hidden layers.

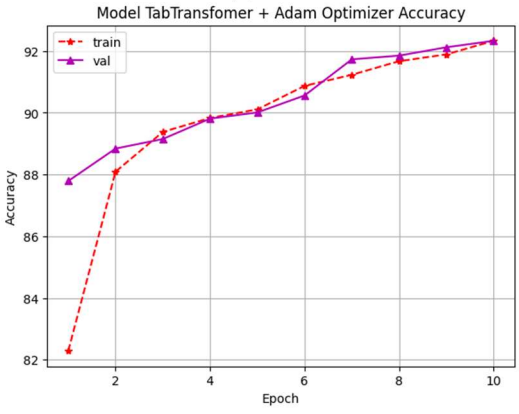
Also, we imply an embedding layer to transform the categorical features embeddings into continuous vectors and assigned a size of 64 to this embedding layer. Finally, the authors use a positional encoding to allow the model to consider the order of the input features. The length of the positional encoding is another architecture parameter and we set it as 32.

Table 4. Comparing our method TabTransformer with variants of optimizers

Model Used	Accuracy
TabTransformer + SGD	87.06
TabTransformer + Avg. & Sub.	89.67
TabTransformer + Adam	92.33

The optimizer leads a path by driving the training process to the global maximum and avoiding local maxima. To swiftly attain the global maximum, varying optimizers are used such as, Adam optimizer, SGD Optimizer and Avg and Sub. Optimizers and utilize a learning rate of 0.001 for a unbiased evaluation. The objective function aid in determining the training and thus assist in the problem of generalization.

(a)



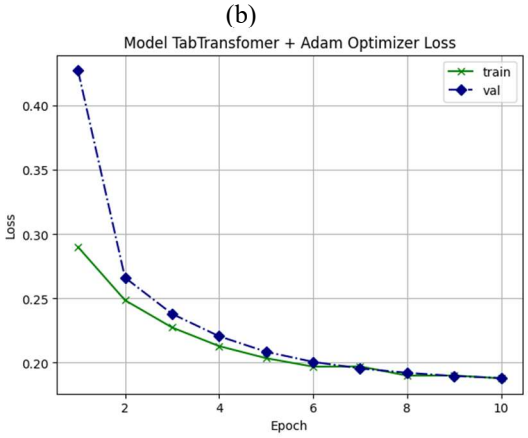


Figure 6 (a) illustrates the TabTransformer models' using Adam optimizer., and (b) the loss calculated per epoch.

In this treatise, the objective function is set as that of Categorical Cross entropy. Similarly, the number of batches are set to 256 for all the three models for providing a fair level analysis and all the three models were trained for 10 epochs. We trained only for 10 epochs as, the model researched saturations thus, not to overfit we use early stopping with a patience of 4. This was sufficient in providing better significant results without over-training and over-parameterization. The accuracy and loss curves of the model are shown in figure 6.

Table 5. Comparison with the existing standard literature and apply the 3 variants of optimizers

Reference	Model used	Accuracies
[11]	Autoencoders	91.2
[12]	CNN-LSTM	89.64
[14]	CART	90
[16]	SVM	89
[18]	LGBANN	90
[19]	CNN	91
ours	**TabTransformer + SGD** **TabTransformer + Avg. & Sub.** **TabTransformer + Adam**	**87.06** **89.67** **92.33**

6. CONCLUSION

Our work developed a seamless framework for detecting IoT botnet attacks not just limited to classical machine learning methods but extensively applied to various deep learning methods varying the optimization techniques. The botnet dataset (figure 1) N-BaIoT is implied on several Machine, Deep learning models and these classifiers ranging from LR to Naïve Bayes are extensively analysed. It is observed that, on the chosen device, TabTransformer model with Adam Optimizer are best-performing ones and the other DL models (ANN, LSTM, and Deep Residual CNN) to create a top-level baseline detection model were good enough but not on par with the TabTransformer. The experimental results demonstrate that TabTransformer method applied for various optimizers (SGD, Adam, Avg. & Sub) were most effective in detecting Bashlite and Mirai botnets in N-BaIoT. The accuracy of the remaining other models was significantly lagging however, the accuracy scores of the SVM and LR were quite high. For DL-based classification, the performance of the Deep Residual CNNs was much better than that of Artificial Neural Network (ANN) and LSTM, and the accuracy score of the LSTM was higher than that of the ANN.

The contribution of this work is the development of a seamless framework for detecting IoT botnet attacks using a variety of deep learning methods and optimization techniques. Additionally, the study highlights the importance of choosing the right deep learning method and optimizer for botnet detection, which can significantly improve the accuracy of the detection model. Our study shows that the performance of botnet detection is highly dependent on the choice of deep learning method and optimizer used. The TabTransformer model with Adam Optimizer achieved the highest accuracy score of 92.33% in detecting botnet attacks on various IoT devices.

REFERENCES

[1] Bagui S, Wang X, Bagui S. Machine learning based intrusion detection for IoT botnet. International Journal of Machine Learning and Computing. 2021 Nov;11(6):399-406.

[2] Subba B, Biswas S, Karmakar S. A neural network based system for intrusion detection and attack classification. In2016 Twenty Second National Conference on Communication (NCC) 2016 Mar 4 (pp. 1-6). Ieee.

[3] Angrishi K. Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. arXiv preprint arXiv:1702.03681. 2017 Feb 13.

[4] Enigmasoftware, "BASHLITE Malware Hits Over One Million IoT Devices," 2020, https://www.enigmasoftware.com/ bashlite-malware-hits-one-million-iot-devices/.

[5] Kumar M. IoT botnets found using Default Credentials for C&C server Databases. 2020, https://thehackernews.com/ 2018/06/iot-botnet-password.html.

[6] Bankinfosecurity, "Massive botnet attack used more than 400,000 IoT devices," 2020, https://www.bankinfosecurity.com/massivebotnet-attack-used-more-than-400000-iotdevices-a-12841.

[7] ingbots, “e Future of Botnets in the Internet of ings,” 2020, https://securityintelligence.com/thingbots-the-futureof-botnets-in-the-internet-of-things/.

[8] Baker, A. R, Esler, and J. Snort Ids, IPS Toolkit; 30 Corporate, Elsevier Inc., Burlington, MA, USA, 2007.

[9] Shah SA, Issac B. Performance comparison of intrusion detection systems and application of machine learning to Snort system. Future Generation Computer Systems. 2018 Mar 1;80:157-70.

[10] Meidan Y, Bohadana M, Shabtai A, Guarnizo JD, Ochoa M, Tippenhauer NO, Elovici Y. ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. InProceedings of the symposium on applied computing 2017 Apr 3 (pp. 506-509).

[11] Meidan Y, Bohadana M, Mathov Y, Mirsky Y, Shabtai A, Breitenbacher D, Elovici Y. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Computing. 2018 Oct 11;17(3):12-22.

[12] Alkahtani H, Aldhyani TH. Botnet attack detection by using CNN-LSTM model for Internet of Things applications. Security and Communication Networks. 2021 Sep 9;2021:1-23.

[13] Catillo M, Pecchia A, Villano U. Botnet Detection in the Internet of Things through All-in-one Deep Autoencoding. InProceedings of the 17th International Conference on Availability, Reliability and Security 2022 Aug 23 (pp. 1-7).

[14] Htwe CS, Thant YM, Thwin MM. Botnets attack detection using machine learning approach for IoT environment. InJournal of Physics: Conference Series 2020 Sep 1 (Vol. 1646, No. 1, p. 012101). IOP Publishing.

[15] Bracci, Lorenzo, and Amirhossein Namazi. "Evaluation of unsupervised machine learning models for anomaly detection in time series sensor data." (2021).

[16] Timčenko V, Gajin S. Machine learning based network anomaly detection for IoT environments. InICIST-2018 conference 2018.

[17] Alharbi A, Alosaimi W, Alyami H, Rauf HT, Damaševičius R. Botnet attack detection using local global best bat algorithm for industrial internet of things. Electronics. 2021 Jun 3;10(11):1341.

[18] Kim J, Shim M, Hong S, Shin Y, Choi E. Intelligent detection of iot botnets using machine learning and deep learning. Applied Sciences. 2020 Oct 8;10(19):7009.

[19] Marzano A, Alexander D, Fonseca O, Fazzion E, Hoepers C, Steding-Jessen K, Chaves MH, Cunha Í, Guedes D, Meira W. The evolution of bashlite and mirai iot botnets. In2018 IEEE Symposium on Computers and Communications (ISCC) 2018 Jun 25 (pp. 00813-00818). IEEE.

[20] Wang A, Liang R, Liu X, Zhang Y, Chen K, Li J. An inside look at IoT malware. InIndustrial IoT Technologies and Applications: Second EAI International Conference, Industrial IoT 2017, Wuhu, China, March 25–26, 2017, Proceedings 2 2017 (pp. 176-186). Springer International Publishing.

[21] Mitchell TM. Machine learning. Vol. 1. New York: McGraw-hill, 2007.

[22] Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. Vol. 2. New York: springer, 2009.Aug.

[23] Burkart N, Huber MF. A survey on the explainability of supervised machine learning. Journal of Artificial Intelligence Research. 2021 Jan 19;70:245-317.

[24] Barlow HB. Unsupervised learning. Neural computation. 1989 Sep 1;1(3):295-311.

[25] Kaelbling LP, Littman ML, Moore AW. Reinforcement learning: A survey. Journal of artificial intelligence research. 1996 May 1;4:237-85.

[26] Zhang Z, Chow WS. Tensor locally linear discriminative analysis. IEEE Signal Processing Letters. 2011 Aug 22;18(11):643-6.

[27] Murphy KP. Machine learning: a probabilistic perspective. MIT press; 2012 Sep 7.

[28] Cortes C, Vapnik V. Support-vector networks. Machine learning. 1995 Sep; 20: 273-97.

[29] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015 May 28;521(7553):436-44.

[30] Goodfellow, I, Yoshua B, and Aaron C. Deep learning. MIT press, 2016.

[31] Bishop, Christopher M. Neural networks for pattern recognition. Oxford university press, 1995.

[32] Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Communications of the ACM. 2017 May 24;60(6):84-90.

[33] Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997 Nov 15;9(8):1735-80.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[34] Huang X, Khetan A, Cvitkovic M, Karnin Z. Tabtransformer: Tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678. 2020 Dec 11.
[35] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Advances in neural information processing systems. 2017;30.
[36] Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016 Sep 15.
[37] Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747. 2016 Sep 15.
[38] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014 Dec 22.
[39] Dehghani M, Hubálovský Š, Trojovský P. A new optimization algorithm based on average and subtraction of the best and worst members of the population for solving various optimization problems. PeerJ Computer Science. 2022 Mar 7;8:e910.
[40] Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research. 2011 Jul 1;12(7).

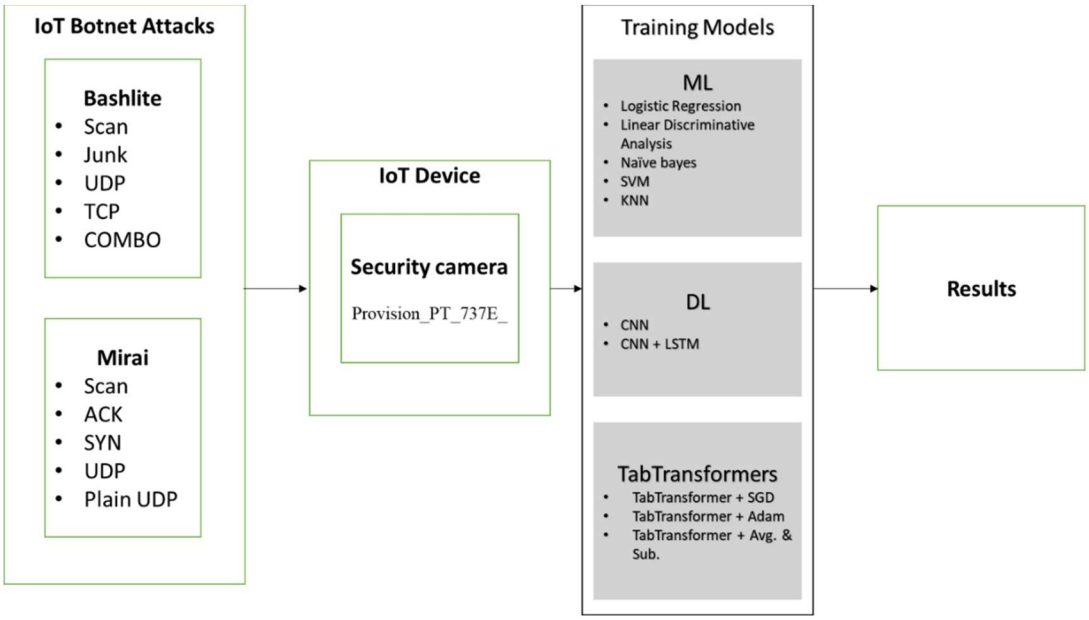


Figure 2. Proposed framework for the detection of the Provision PT-737E device.