ISSN: 0067-2904

Deep Learning Approaches for IoT Intrusion Detection Systems

Baseem A. kadheem Hammood[1]*, Ahmed T. Sadiq[2]
[1]*Department of Computer Science, Informatics Institute for Postgraduate Studies, Iraqi Commission for Computers and Informatics, Baghdad, Iraq*
[2]*Department of Computer Science, University of Technology, Baghdad, Iraq*

Received: 23/5/2023 Accepted: 5/9/2023 Published: 30/11/2024

Abstract:
The Internet of Things (IoT) has had a substantial impact on a number of industries, including smart cities, the medical field, the automotive industry, and logistics tracking. But along with the IoT's advantages come security worries that are proliferating more and more. Deep learning-based Intelligent Network Intrusion Detection Systems (NIDS) have been created to detect continually evolving network threats and trends in order to address this issue. Six alternative deep learning algorithms, including CNN, RNN, and DNN architectures, are used in this research to present a novel anomaly-based solution for IoT networks. Three distinct intrusion detection datasets were used to evaluate the algorithms, and the findings revealed that the hybrid method performed better than the others in terms of accuracy. In particular, the hybrid algorithm had a 99.13% accuracy on the UNSW-NB15 dataset, an 89.01% accuracy on the IoTID20 dataset, and a 90.83% accuracy on the BoTNeTIoT-L01-v2 dataset. These results point to the suggested hybrid algorithm as superior to previous algorithms and a potential intrusion detection method for Internet of Things networks.

Keywords: IDS, Deep Learning, CNN, Intrusion Detection, IoT.

كشف اختراق الانترنت الأشياء بواسطة التعلم العميق

بسيم عبد الكاظم حمود[1]*, احمد طارق صادق[2]
[1]القسم علوم الحاسوب، معهد المعلوماتية للدراسات العليا، الهيئة العراقية للحاسبات والمعلوماتية، بغداد، العراق
[2] القسم علوم الحاسوب ، جامعة التكنلوجيا، بغداد، العراق

الخلاصة
ان نمو السريع في تقنية الانترنت الأشياء إثر كبيرا على مختلف الصناعات مثل المدن الذكية والمهن الصحية والسيارات والخدمات اللوجستية. مع فوائد الكبيرة لتقنية الانترنت الأشياء تأتي مخاوف امنية التي أصبحت سائدة بشكل متزايد. لمعالجة هذه المشكلة تم تطوير أنظمة الكشف عن الاختراق الشبكة الذكية باستعمال ست خوارزميات مختلفة للتعلم العميق. تم استعمال معماريات CNN, RNNو DNN. حيث تم اختبار الخوارزميات على ثلاث قواعد بيانات مختلفة. حققت خوارزمية الهجينة دقة قدرها 99.13% عند تطبيقها على مجموعة بيانات UNSW-NB15 و 89.01% عند تطبيقها على مجموعة بيانات IoTID20

* Email: ms202130656@iips.icci.edu.iq

ودقة قدرها 90.83% عند تطبيقها على مجموعة بيانات 2BoTNeTIoT–L01–v. تشير هذه النتائج الى ان الخوارزمية الهجينة المقترحة تتفوق على باقي الخوارزميات ويمكن ان تكون خط فعالا لاكتشاف التسلل في الشبكات الانترنت الأشياء.

1. Introduction

Deep learning has been suggested as a cutting-edge way to find IoT intrusions because it uses data-driven, anomaly-based methods and can find new attacks that haven't been seen before. Examine the deep learning models put forth for IoT intrusion detection in this study [1].

Due to the exponential growth of cyberattacks, intrusion detection is essential to the security architecture of the Internet of Things (IoT). The significant improvements in communication, cloud computing, and IoT have created significant security challenges. Due to these advancements and the inadequacy of current security measures, cyberattacks are increasing rapidly [2].

A trustworthy IoT intrusion detection system has been proposed using a deep transfer learning-based approach. The suggested method is based on a deep transfer learning model that can be trained on a large dataset of labeled data from numerous sources. The model can then be adjusted using a smaller batch of labeled data from the target domain [3].

Another proposed IoT intrusion detection system makes use of deep learning and enhanced transient search optimization. The suggested system employs a new transient search optimization technique to maximize the hyperparameters of the deep learning model. The results show that the recommended method outperforms other state-of-the-art intrusion detection systems in terms of accuracy and false alarm rate [3].

For IoT intrusion detection systems, deep learning algorithms have shown tremendous potential. They are able to provide cutting-edge solutions thanks to their data-driven, anomaly-based technique and their ability to identify newly emerging, unrecognized attacks. Deep transfer learning-based techniques and improved transient search optimization algorithms have been called reliable intrusion detection systems for the IoT [1, 4].
The remainder of the article is organized as follows: Related works are explained in Section 2. IoT intrusion detection systems and deep learning approaches are presented in sections 3 and 4. The methodology is described in detail in Section 5. The experimental results are given in Section 6. Finally, Section 7 includes the conclusion and future work.

2. Related Work

A wireless IDS system utilizing a wrapper-based feature extraction unit and a feed-forward deep neural network (FFDNN) was suggested in [5]. The WFEU extraction method generates a condensed optimum feature vector using the Extra Trees methodology. The UNSW-NB15 and AWID datasets for intrusion detection are used to examine the efficacy and efficiency of the WFEU-FFDNN. Other machine learning (ML) techniques, including the k-Nearest Neighbor, Random Forest, Support Vector Machine, Naive Bayes, and Decision Tree, are also compared to the WFEU-FFDNN in this evaluation. Attacks on binary and multiclass targets are part of the experimental study. The findings demonstrate that the proposed WFEU-FFDNN has higher detection precision than existing methods. The WFEU created an ideal feature vector with 22 properties for the UNSW-NB15.

A robust framework system for detecting intrusions based on the IoT environment was presented in [6]. To develop the suggested method, the IoTID20 dataset, a recent dataset from the IoT infrastructure, was attacked. In this framework, the intrusion was categorized using convolutional neural networks (CNN), long short-term memories (LSTM), and a hybrid convolutional neural network with the long short-term memory (CNN-LSTM) model. The particle swarm optimization approach (PSO) was used to choose pertinent characteristics from the network dataset in order to enhance the suggested system while reducing the dimensions and complexity of the network dataset. Deep learning methods were applied to the features that were recovered.

The UNSW-NB15 dataset was prepared as a number of distinct files, and binary classifications were utilized for labeling, according to [7]. To test models once rather than once for each file in this inquiry, the entire dataset was pooled into one file. The attack families from the dataset were then used as a new label to create a multi-classification-labeled dataset. Using the larger dataset, they examined the effectiveness of deep learning within two categorization groups. They evaluated the outcomes of using relevant studies. The effectiveness of deep learning and machine learning models was assessed using accuracy and loss utilizing the extended dataset. The classification method used was CNN-LSTM, with an accuracy of 98.80%.

[8] came up with a deep, multi-layer classification method for intrusion detection. This method combines the two-phase detection of the existence and type of an intrusion with an oversampling strategy. This makes the classification results more accurate. The research shows that the proposed technique works best with intrusion type identification label (ITI) oversampling and 150 neurons for the single-hidden layer feedforward neural network (SLFN) and with two layers and 150 neurons for the LSTM, with an accuracy of 86%.

Researchers in [9] suggested that SDN-based smart homes may be protected using machine learning and deep learning settings for smart houses that use software-defined networking and Mininet to provide immediate virtual networks for IoT. In this study, ML and DL experiments were run on the first Software-Defined Networking (SDN) dataset, which was obtained from smart homes by conducting real attacks and creating regular traffic, and the second IoTID20 dataset, freely available online.

[10] described a 5-layered method for spotting intrusions in large datasets. In this study, the learning rate and perceptions of the machine model were increased by adding more unique properties of recurrent neural networks with short-term long memory in both directions (RNNBiLSTM). These properties are used in the proposed Accessible Contiguous Attribute Assessment and Selection (ACAAS) solution (Access Control-as-a-Service), which is meant to protect IoT networks, recognize attacks, and improve prediction performance.

3. IoT Intrusion Detection System

An Internet of Things system's network layer is where an intrusion detection system, or IDS, typically functions [11]. IoT Network Intrusion Detection Systems (NIDS) keep track of Internet traffic between networked devices. In order to identify dangers and safeguard the network from unauthorized users and harmful attacks, it serves as an additional layer of defense [12].

A typical intrusion detection system comprises instruments or processes that look for attacks or unauthorized access to system activity. Sensors are often part of an IDS, and data verification and intrusion detection tools are used to examine the data collected from these sensors. The IoT invasion is the technique for IoT intrusion detection systems; they are illegal actions or activities that damage the IoT ecosystem. In other terms, an intrusion is any act that jeopardizes the integrity, availability, or confidentiality of information [13].

An IoT system's deployed IDS should be able to scan data packets and respond in real time, analyze data packets at different IoT network tiers utilizing different protocol stacks, and adjust to different threats [14, 15].

4. Deep Learning Approaches

Due to its rapidly rising applications in atomistic, image-based, spectral, and textual data modalities, deep learning (DL) is one of the areas of materials data science that is expanding at the fastest rate. This study provides a solid theoretical framework for understanding deep neural networks and their applications. This study shows how to describe the output of trained networks using layer-to-layer iteration equations and nonlinear learning dynamics, starting from a first-principles component-level view of networks [16, 17].

5. Methodology

Three different kinds of datasets, namely UNSW-NB15, IoTID-20, and BotNetIoT, were used in the aforementioned study. Six different kinds of deep learning architectures were examined in order to ascertain how well they performed on these datasets. The data underwent a process known as "data preparation" prior to being used to train the models on the datasets. The IoTID-20 dataset was split into training and testing sets in an 80:20 ratio, whereas the UNSW-NB15 and BotNetIoT datasets were divided into training and testing sets in a 70:30 ratio. Following that, the training data was fed into a variety of deep learning algorithms, including DNN, CNN1, CNN2, GRU, LSTM, and a CNN/LSTM hybrid model. Finally, as shown in Figure 1, the performance of the trained models was assessed using the test data.

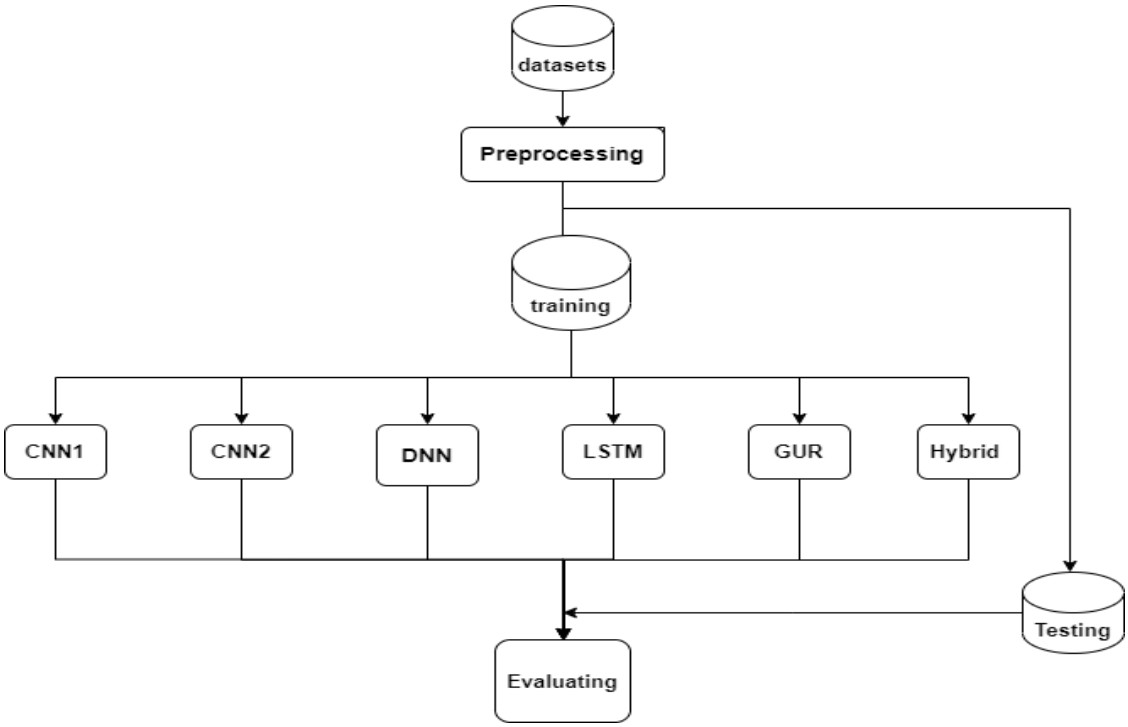


Figure 1: Applying various DL algorithms to different datasets

5.1. Datasets

Three distinct IoT intrusion detection datasets were employed in this study. A labeled network traffic dataset called UNSW-NB15 [18] is the first dataset that has been applied in studies on IoT intrusion detection. It has nearly two million records of network traffic, both benign and malicious, that were recorded in a real-world network environment. The dataset has labels that say whether the traffic is malicious or not, and it also gets 49 network

properties from each network flow. This makes it a useful tool for figuring out how well intrusion detection methods work for IoT networks. Second, the IoTID-20 [19, 20] dataset is a labeled dataset that is openly accessible and was created exclusively for IoT intrusion detection research. It includes network traffic information gathered from an IoT environment with 20 different kinds of IoT devices in a real-world setting. With a total of 15 assault scenarios produced by using various network attacks, such as brute-force attacks, DoS attacks, and malware infections, the dataset contains both benign and malicious traffic.

The dataset contains 27 network parameters, including source and destination IP addresses, port numbers, protocol kinds, and packet sizes, which are retrieved from each network transaction. Additionally, labels indicating whether a network flow is malicious or benign are included for each flow.

The IoTID-20 dataset can be used to assess how well different intrusion detection systems and machine learning algorithms work at spotting assaults that are specifically targeted at IoT devices.

Third, the malicious botnet dataset (BotNetIoT), which is made up of data files collected during the detection of IoT botnet attacks on a cybersecurity system, is focused on using an IoT dataset for intrusion detection systems. On Kaggle [21], this dataset is accessible to everyone.

Researchers collected network traffic data from nine distinct Internet of Things (IoT) devices connected to a local network using the software Wireshark to build this dataset. The information was gathered in a file format called packet capture (PCAP), which is frequently employed in network analysis. 23 statistics features for the network's main switch are among the data packets from the network that are included in the PCAP file.

The BotNetIoT dataset includes traffic that is both beneficial and harmful as a result of various IoT-specific attacks like botnets and infiltration campaigns. The information is helpful for evaluating intrusion detection systems' performance in identifying IoT-specific threats and determining the health of networks. Additionally, the dataset can be used to develop and evaluate machine learning algorithms for IoT intrusion detection.
Three datasets' specifications are listed in Table 1.

Table 1: specification of datasets

Dataset	No of instances	No of instances	No of classes
IoTID20	*625783*	*86*	*9*
UNSW_NB15	*2540047*	*49*	*10*
BoTNeTIoT-L01-v2	*7062606*	*27*	*9*

5.2. Data Pre-processing:

Pre-processing is an important stage before the processing of the data, and it includes the following four stages:

5.2.1. Data Cleaning

In this stage, duplicated and not-useful data and features like objects and IDs are removed after being identified to enhance and present the best results of the prediction system.

5.2.2. Handling Missing Values

There are some missing values in the dataset that were replaced with zero values.

5.2.3. Normalization

The feature normalization stage of data pre-processing is crucial. Data normalization is a useful strategy for increasing DL and ML accuracy. Data from the three datasets are scaled using the Standard Scaler to a range between 0 and 1. The following equation represents the process of normalization with min-max normalization:

$$x' = \frac{x - (x)}{(x) - min(x)} \quad (1)$$

5.2.4. Data correlation

A statistical technique used in research to determine the association between two variables and gauge the strength of their linear relationship is correlation analysis. The magnitude of the change in one variable as a result of the change in the other is determined using correlation analysis.

Because all algorithms require numeric values, this work transforms categorical values into numerical values.

5.3. Deep Learning in IoT Intrusion Detection System

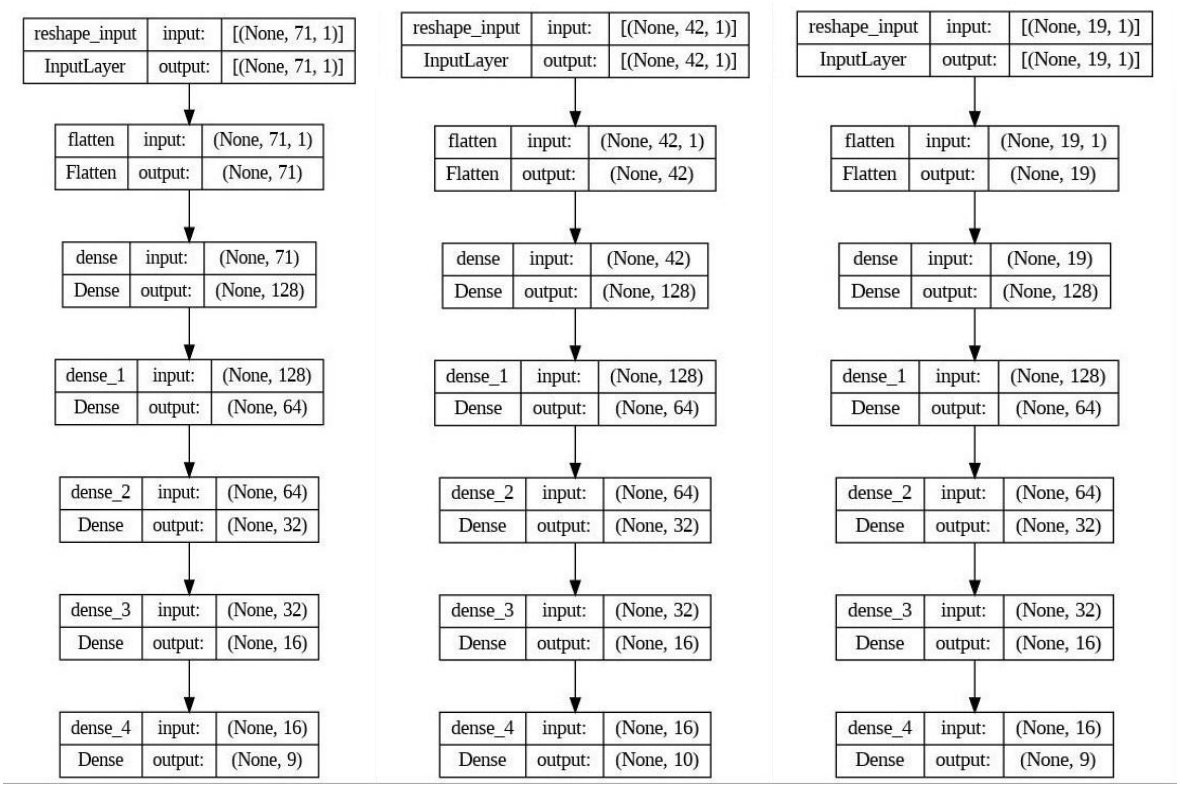
Deep learning techniques were used on three different datasets in this research because it is a modern technology. Six different architectures were used in deep learning. It relied on the algorithms DNN, Recurrent Neural Network (RNN), and CNN in these architectures, as shown in the following algorithms:

5.3.1. DNN

The LSTM, GRU, or convolution layers are not used in the straightforward DNN approach. In comparison to the other algorithms mentioned in this paper, it is one of the simplest and has a quicker training time because it only has dense layers. Dense layers, an essential component of many artificial neural networks used in deep learning, are included in the five-layer DNN technique. Every neuron in the layer underneath is linked to every neuron in the layer above by a thick layer, resulting in a fully interconnected network of neurons. In a dense layer, each neuron receives input from every other neuron above it, processes it, and then sends output to every other neuron above it. A weighted sum of the inputs is calculated by the neuron, and then it is subjected to an activation function.

The backpropagation algorithm is employed during training to determine the weights of each input link. Errors made by the output layer are retransmitted across the network using backpropagation, which modifies the weights to make the error less. Four layers of the Rectified Linear Unit (ReLU) activation function make up the DNN algorithm. There are 128 neurons in the top layer, 64 and 32 neutrons in the two middle layers, respectively, and 16 neurons in the bottom layer. When the IoTID20 or BoTNetIoT datasets are used, the fifth layer has nine neurons, while the UNSW_NB15 dataset uses 10 neurons. The SoftMax activation function is utilized in the fifth layer.

BoTNetIoT, IoTID20, and UNSW_NB15 are three different types of datasets, and each one of them was subjected to the DNN method in Figure 2. The DNN is a straightforward method that has been found to work well in a variety of tasks, including audio and picture recognition, natural language processing, and predictive analytics.



(a) **(b)** **(c)**

Figure 2: Shows the layers of the DNN algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

5.3.2. CNN1

A deep learning method that mixes dense and convolutional layers is called CNN1. For processing high-dimensional inputs like photos or videos, convolutional neural networks (CNNs) are a common type of neural network. The convolutional layer, a critical part of CNNs, uses a network of filters to extract local features or patterns from the input data. These filters are applied to the input data by sliding over the input data and computing a dot product between the filter values and the input data values at each place. They are learned during training via backpropagation. The final product of the convolution is a new matrix reflecting the filter output applied to the input data at that location. A collection of feature maps is produced by applying various filters to the input data, and the convolutional layer can use these feature maps to train itself to recognize local features like edges, corners, and textures.

Three convolutional layers, each with 20 filters, three kernel sizes, and padding of the same kind, make up the CNN1 algorithm. All layers employ the ReLU activation function, which is followed by four dense layers that each contain 128, 64, 32, and 16 neurons, respectively. The first four layers are of the dense type and use the ReLU activation function, while the final layer makes use of the SoftMax activation function. For the BoTNetIoT and IoTID20 datasets, the last layer has nine neurons, while for the UNSW_NB15 dataset, it has ten neurons. The application of the CNN1 algorithm to various datasets is shown in Figure 3.

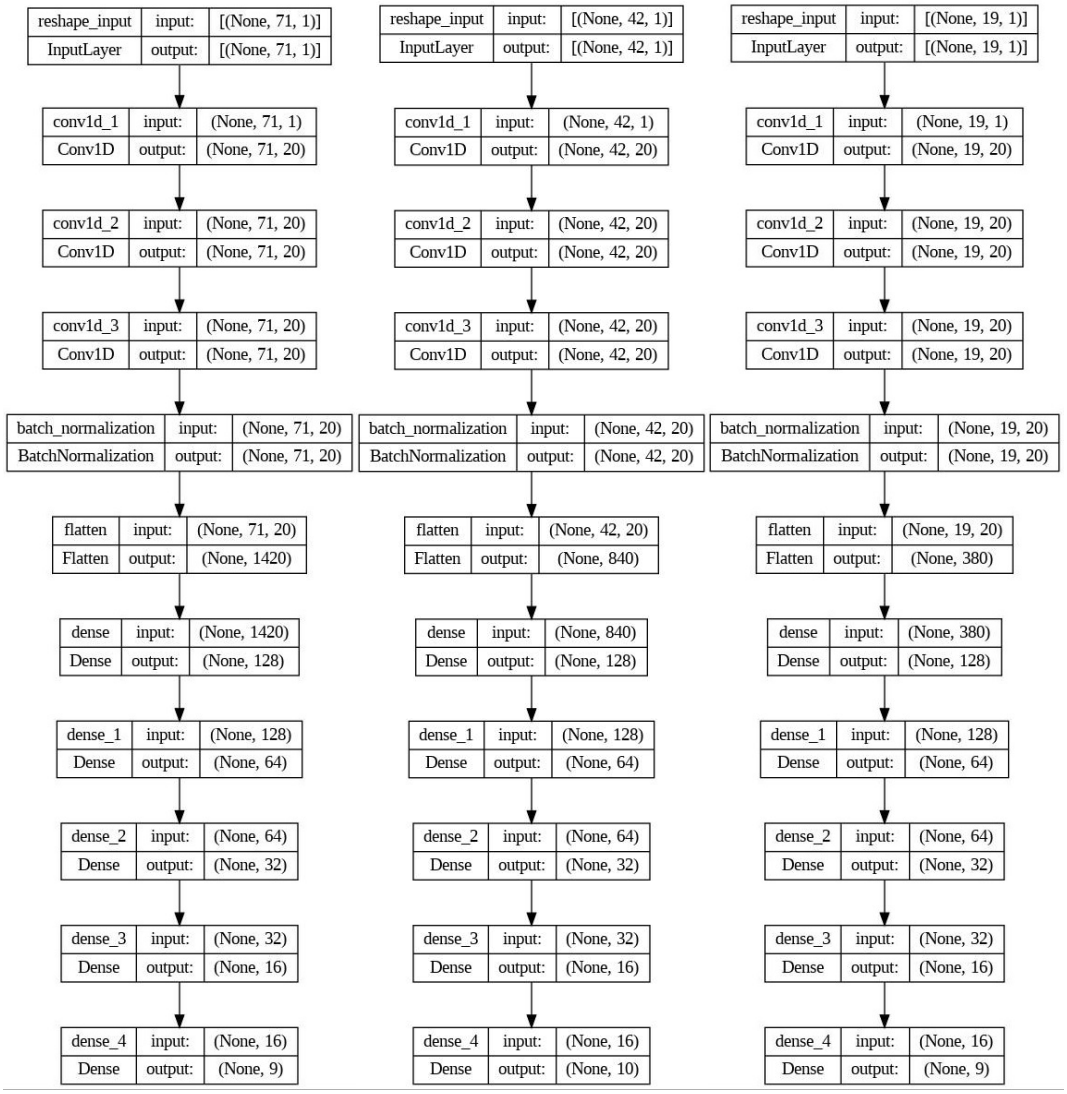


Figure 3: Shows the layers of the CNN1 algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

5.3.3. CNN2

This study's suggested method makes use of a CNN architecture with four convolutional layers. With the same padding and three kernel sizes, each layer has 30 filters. All layers utilized the Rectified Linear Unit (ReLU) activation function, and then four dense layers were added, each containing 128, 64, 32, and 16 neurons. The ReLU activation function is used in the first four dense-type layers; the fifth and final layer uses the SoftMax activation function. Ten neurons make up the last layer for the UNSW NB15 dataset, compared to nine for the BoTNetIoT and IoTID20 datasets. Figure 4 displays how this strategy was applied to several datasets.

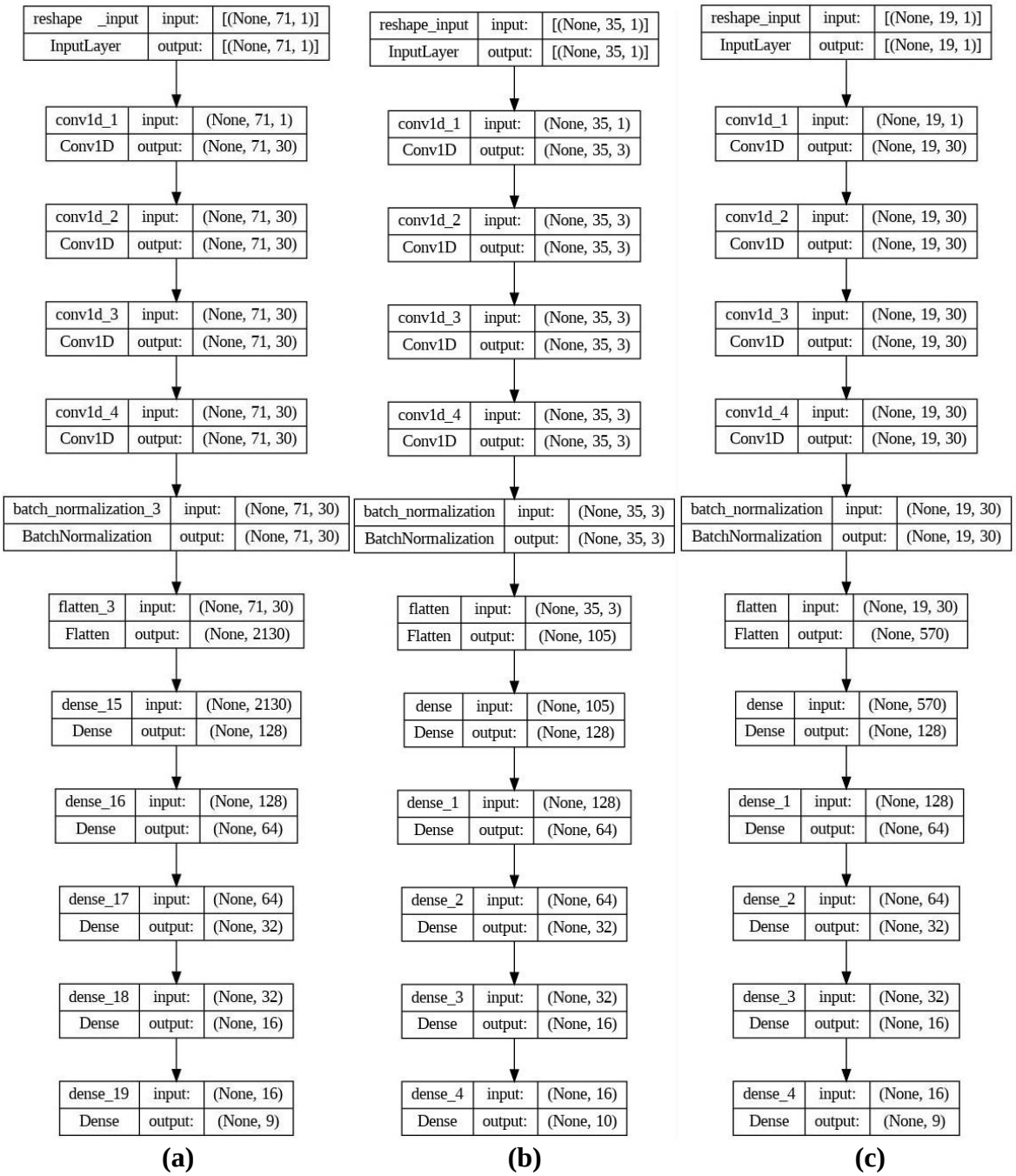


Figure 4: Shows the layers of the CNN2 algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

5.3.4. GRU

The GRU algorithm makes use of dense and gated recurrent unit (GRU) layers. Sequential data, such as time series or text written in natural language, is simulated using recurrent neural networks (RNNs) of the GRU form. The GRU has gating techniques to control the information flow between various time stages in the sequence. A hidden state, which acts as the model's internal memory, and an update gate, which determines how much of the current input and previous hidden state to use for updating the current hidden state, are the two fundamental components of the GRU cell. The update gate is created using a sigmoid activation function. It creates a probability between 0 and 1 by linearly integrating the initial hidden state and the current input. This probability is used to combine the current input with

the previous hidden state, with the current input receiving more weight as the gate approaches 1 and the previous hidden state receiving more weight as the gate approaches 0.

The GRU additionally features a reset gate that determines how much of the previous hidden state should be forgotten at the current time step, in addition to the update gate. The reset gate is calculated using a sigmoid activation function and a linear combination of the current input and the initial concealed state.

The reset gate's output is used to update a reset gate vector. The GRU avoids overfitting by generating a new hidden state by adding noise to a weighted sum of the output from the update gate and the old hidden state.

The GRU finds long-term relationships in the input sequence without running into the problem of vanishing gradients that can happen in traditional RNNs. This is done by controlling the flow of information between certain time steps using gating techniques.

The final layer of this method uses ten neurons for the UNSWNB15 dataset, nine neurons for the BoTNetIoT and IoTID20 datasets, and one GRU layer with 200 units. The use of the GRU technique on various datasets is shown in Figure 5.

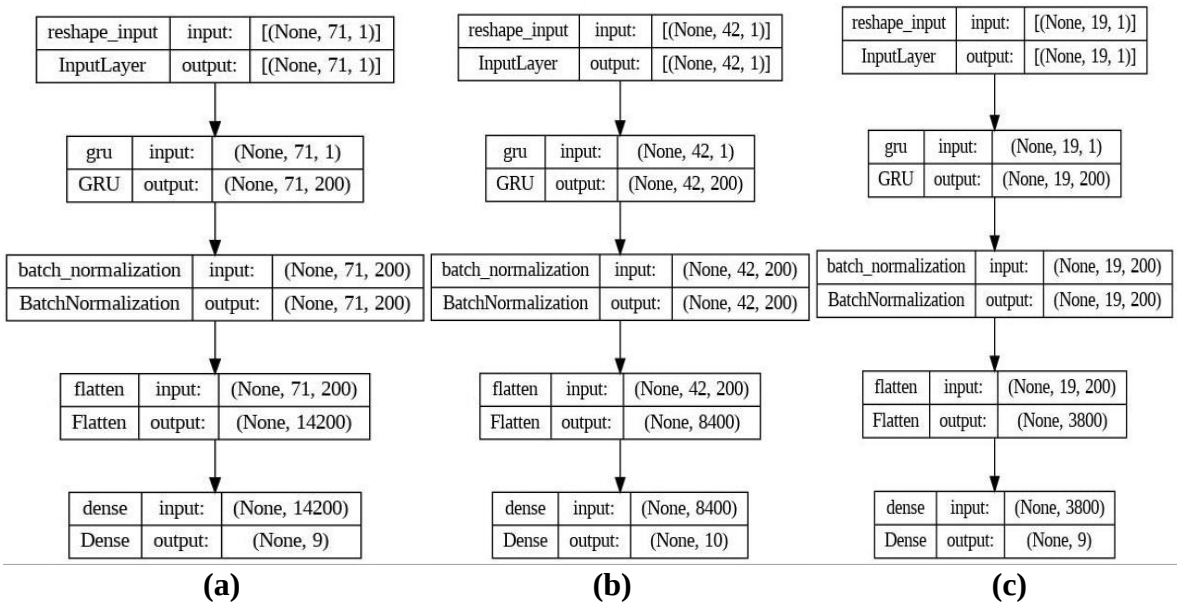


Figure 5: Shows the layers of the GRU algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

5.3.5. LSTM

The LSTM approach simulates sequential data, such as time series or text in natural language, using dense LSTM layers. A specific type of recurrent neural network called the LSTM uses memory cells to store data for a very long time and gating mechanisms to control the flow of information between different time steps in a sequence. The LSTM's main innovation is the use of a cell state, an input gate, and an output gate. The input gate controls how much current is input to update the cell state, and the output gate controls how much current is input to compute the output. The internal memory of the model is the cell state. The most recent hidden state and the current input are linearly combined to create the input gate, which is then transformed into a probability value between 0 and 1.

The state of the cell is changed by adding a candidate value that is made by applying a hyperbolic tangent activation function to a linear combination of the current input and the previous hidden state. Additionally, the LSTM contains two gates: forget and output.

These gates regulate how much of the initial cell state should be forgotten and how much of the current cell state will be used to compute the output, respectively. The LSTM weights the output of the output gate and the last updated cell state to produce a new hidden state.

The LSTM can express long-term dependencies in the input sequence while avoiding the issue of vanishing gradients that could occur in conventional RNNs. Due to its ability to store information over a long period of time and control the flow of information between different time steps, it is a powerful tool for modeling sequential data in fields like language modeling, speech recognition, and machine translation.

A single LSTM layer with 200 units and a dense layer with nine neurons were used for the BoTNetIoT and IoTID20 datasets, while 10 neurons were used for the UNSWNB15 dataset. Figure 5 illustrates how these techniques were applied to distinct datasets.

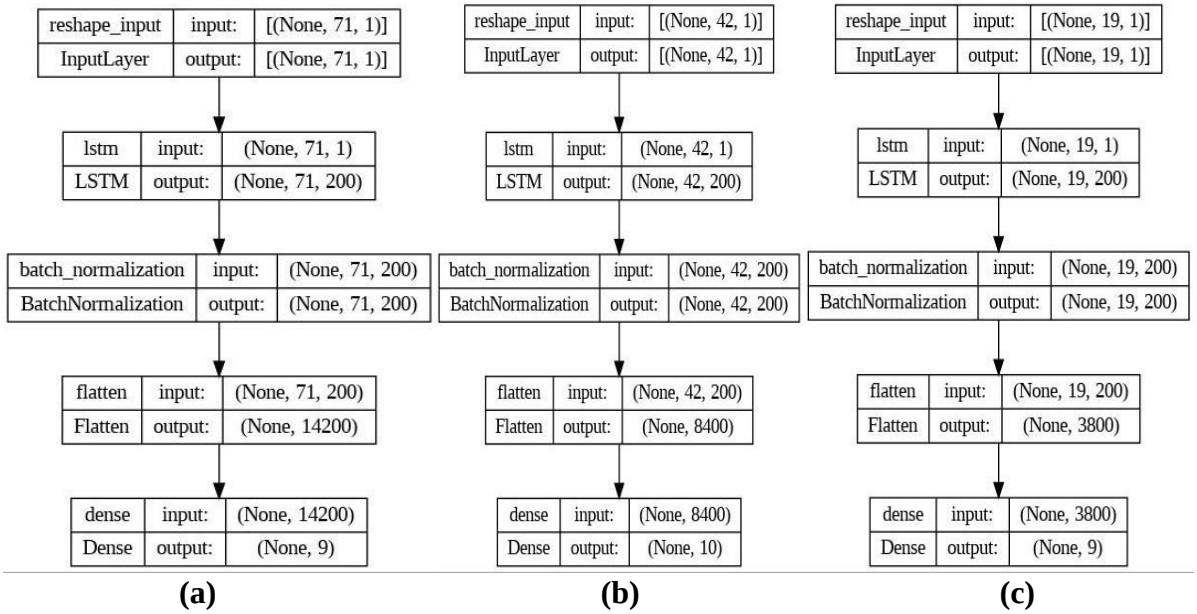


Figure 6: Shows the layers of the LSTM algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

5.3.6. Hybrid

The hybrid algorithm requires more time to execute because it is more sophisticated than the earlier ones. In this stage, two processes occur: the first is the feature selection process, and the second is the learning process. It has multiple layers, including four convolutional layers, three kernels, and 30 filters, each with the same padding. Rectified Linear Unit (ReLU) activation is used in all layers until the final one. These layers are responsible for the feature selection process; in addition, an LSTM layer with 200 units is used, which is responsible for the learning process. Finally, there are four dense layers with 128, 64, 32, 16, and 9 neurons (for the UNSWNB15 dataset) or 10 neurons (for the BoTNetIoT or IoTID20 datasets), and again, the ReLU activation function is used in all layers until the last one. The final layer is activated using the "SoftMax" function. The use of this technique on diverse datasets is shown in Figure 7.

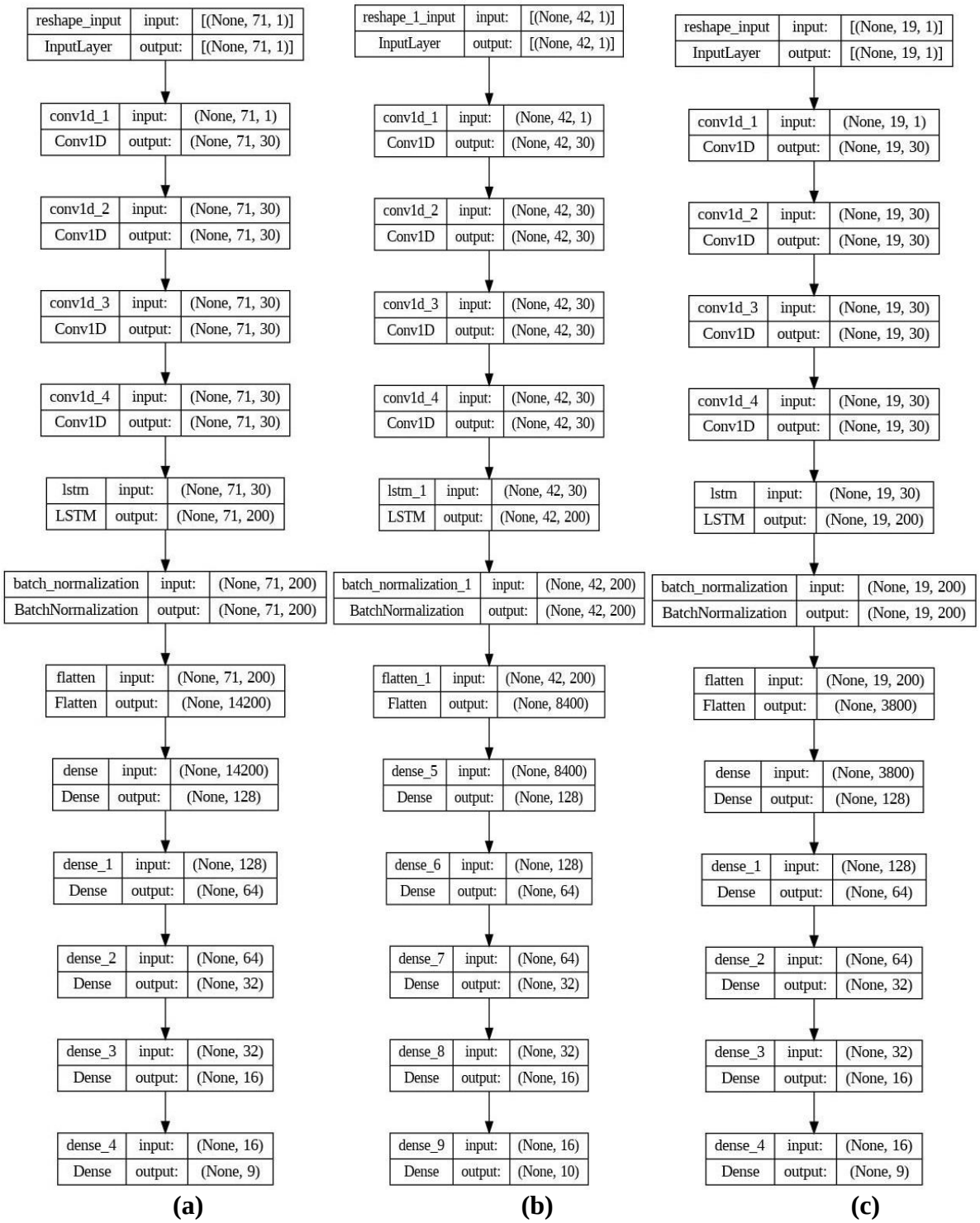


Figure 7: Shows the layers of the hybrid algorithm: (a) the IoTID20 dataset; (b) the UNSW_NB15 dataset; and (c) the BoTNetIoT dataset

6. Experimental Results

This section includes confusion matrices as well as the results for multi-class classification. We assess the model's performance with the aid of accuracy, precision, recall, and the F1 score. In contrast to recall, which is determined by dividing the total number of true positive class values in the test data by the number of true positive predictions, precision is calculated by dividing the total number of true positive predictions by the total number of false negative predictions. The weighted average of recall and precision is the F1 score. By dividing the total number of correct predictions by the total number of predictions, accuracy is

determined. A low recall indicates a high proportion of incorrect negative predictions, and a low accuracy indicates a high proportion of false positive predictions. A high F1 score denotes precision and recall that are in balance, with few false negatives and positives. These measures were calculated using the corresponding equations, which were based on sources [22], [23], and [24].

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (5)$$

where TP = true positive, TN = true negative, FP = false positive and FN= false negative.

The studies were carried out using the Google Colab platform, which has a 2 GHz CPU, an NVIDIA Tesla T4 GPU, 12 GB of RAM, and 53 GB of hard drive. On the IoTID20 dataset, six deep learning methods were tested with a batch size of 1024. Table 2 displays the findings of these experiments.

Table 2: Performance metrics in the IoTID20 dataset

Algorithm	*Accuracy*	*Recall*	*Precision*	*F1Score*
DNN	*88.81%*	*86.68%*	*86.20%*	*82.93%*
CNN1	*88.64%*	*79.78%*	*88.31%*	*79.47%*
CNN2	*88.69%*	*79.22%*	*87.44%*	*79.23%*
GRU	*88.70%*	*82.53%*	*83.42%*	*82.86%*
LSTM	*88.92%*	*86.07%*	*91.25%*	*82.46%*
Hybrid	***89.01%***	*86.99%*	*92.44%*	*82.78%*

Table 2 shows the preference for hybrid algorithms over other algorithms, and the accuracy of this algorithm was 89.01%.

When testing six deep learning algorithms on the UNSW_NB15 dataset, batch_size equals 2048. The results are shown in Table 3.

Table 3: Performance metrics in the UNSW_NB15 dataset

Algorithm	*Accuracy*	*Recall*	*Precision*	*F1Score*
DNN	*98.17%*	*51.10%*	*64.70%*	*49.93%*
CNN1	*98.21%*	*49.38%*	*53.98%*	*49.86%*
CNN2	*98.09%*	*46.55%*	*62.83%*	*47.30%*
GRU	*98.25%*	*52.55%*	*57.34%*	*53.09%*
LSTM	*98.52%*	*57.30%*	*65.55%*	*58.93%*
Hybrid	***99.13%***	*69.23%*	*78.01%*	*70.77%*

Table 3 shows the preference for hybrid algorithms over other algorithms, and the accuracy of this algorithm was 99.13%.

When testing six deep learning algorithms on the BoTNeTIoT-L01-v2 dataset, batch_size equals 2048. The results are shown in Table 4.

Table 4: Performance metrics in the BoTNeTIoT-L01-v2 dataset

Algorithm	*Accuracy*	*Recall*	*Precision*	*F1Score*
DNN	*89.70%*	*85.83%*	*94.95%*	*84.68%*
CNN1	*89.85%*	*86.58%*	*93.85%*	*85.21%*
CNN2	*90.60%*	*88.08%*	*95.95%*	*86.64%*
GRU	*89.88%*	*86.14%*	*94.79%*	*85.04%*
LSTM	*90.03%*	*87.03%*	*95%*	*85.46%*
Hybrid	***90.83%***	*88.58%*	*95.79%*	*87.14%*

Table 4 shows the preference for hybrid algorithms over other algorithms, and the accuracy of this algorithm was 90.83%.
Table No. 2, Table No. 3, and Table No. 4 illustrate the advantages of the hybrid algorithm over the rest of the algorithms.
From Table 5, it can be concluded that the hybrid algorithm takes more time than the rest of the algorithms, unlike the DNN algorithm, which takes less time than the rest.

Table 5: Elapsed time

Dataset	*Algorithm*	*Epoch*	*Elapsed time*
IoTID20	*4Conv1D*	*45*	*3min 45s*
IoTID20	*3Conv1D*	*60*	*4min*
IoTID20	*DNN*	*100*	*3min 20s*
IoTID20	*GRU*	*50*	*15min*
IoTID20	*LSTM*	*130*	*45min 30s*
IoTID20	*Hybrid*	*230*	*1-hour 35min 50s*
UNSW_NB15	*4Conv1D*	*100*	*11min 40s*
UNSW_NB15	*3Conv1D*	*40*	*4min 40s*
UNSW_NB15	*DNN*	*100*	*7min 30s*
UNSW_NB15	*GRU*	*80*	*54min 40s*
UNSW_NB15	*LSTM*	*202*	*2 hours 58min 26s*
UNSW_NB15	*Hybrid*	*602*	*10 hours 42min 8s*
BoTNeTIoT	*4Conv1D*	*60*	*23min*
BoTNeTIoT	*3Conv1D*	*59*	*19min 40s*
BoTNeTIoT	*DNN*	*60*	*13min*
BoTNeTIoT	*GRU*	*40*	*42min 40s*
BoTNeTIoT	*LSTM*	*154*	*3 hours 9min 56s*
BoTNeTIoT	*Hybrid*	*112*	*3 hours 4min 48s*

The study contrasts the approach with some current studies. On the UNSW_NB15 dataset, Table 6 compares the overall accuracy performance across several classes. Table 7 compares the accuracy of studies performed on the IoTID20 dataset for subcategories. In terms of accuracy metrics, our method fared better than the alternatives.

Table 6: General comparison of multiple classification accuracy measures for the UNSW_NB15 dataset

Method	***Accuracy***
[5]	*77.16%*
[6]	*76.30%*
[25]	*66.30%*
Proposed method (Hybrid CNN+LSTM)	***99.13%***

Table 7: General comparison of subcategories with accuracy measures of the IoTID20 dataset

Method	***Accuracy***
[26]	*77.55%*
proposed method (Hybrid CNN+LSTM)	***89.01%***

7. Conclusion

In this study, three Internet of Things intrusion detection datasets (IoTID20, UNSW-NB15, and BoTNeTIoT-L01-v2) were used to assess how well six deep learning algorithms performed. With accuracy ratings of 89.01% on dataset IoTID20, 99.13% on dataset UNSW-NB15, and 90.83% on dataset BoTNeTIoT-L01-v2, the results show that the hybrid method performs better than the other techniques. The tests show how important the LSTM architecture is for enhancing outcomes, whether it is used alone or in conjunction with other algorithms like the CNN algorithm, which eventually led to the creation of the hybrid algorithm. Future work should compare the hybrid algorithm to others in the field of machine learning, including SVM, Random Forest, and XGBoost.

References

[1] S. Tsimenidis, T. Lagkas, and K. Rantos, "Deep Learning in IoT Intrusion Detection," *Journal of Network and Systems Management*, vol. 30, no. 8, pp. 1-40, Jan. 2022, Doi: 10.1007/s10922-021-09621-9.

[2] A. Fatani, M. Abd Elaziz, A. Dahou, M. A. A. Al-Qaness, and S. Lu, "IoT Intrusion Detection System Using Deep Learning and Enhanced Transient Search Optimization," *in IEEE Access*, vol. 9, pp. 123448-123464, 2021, Doi: 10.1109/ACCESS.2021.3109081.

[3] S. T. Mehedi, A. Anwar, Z. Rahman, K. Ahmed, and R. Islam, "Dependable Intrusion Detection System for IoT: A Deep Transfer Learning Based Approach," *in IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 1006-1017, Jan. 2023, Doi: 10.1109/TII.2022.3164770.

[4] A. Awajan, "A Novel Deep Learning-Based Intrusion Detection System for IoT Networks," *Computers*, vol. 12, no. 2, p. 34, Feb. 2023, Doi: 10.3390/computers12020034.

[5] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper based feature extraction for wireless intrusion detection system," *Computers & Security*, vol. 92, p. 101752, 2020, Doi: 10.1016/j.cose.2020.101752.

[6] M. K. Hooshmand and D. Hosahalli, "Network anomaly detection using deep learning techniques," *CAAI Trans Intell Technol*, vol. 7, no. 2, pp. 228-243, Jun. 2022, Doi: 10.1049/cit2.12078.

[7] H. Alkahtani and T. H. H. Aldhyani, "Intrusion Detection System to Advance Internet of Things Infrastructure-Based Deep Learning Algorithms," *Complexity*, vol. 2021, pp. 1-18, 2021, doi: 10.1155/2021/5579851.

[8] R. Qaddoura, A. M. Al-Zoubi, H. Faris, and I. Almomani, "A Multi-Layer Classification Approach for Intrusion Detection in IoT Networks Based on Deep Learning," *Sensors*, vol. 21, no. 9, p. 2987, Apr. 2021, Doi: 10.3390/s21092987.

[9] W. A. Alonazi, H. Hamdi, N. A. Azim, and A. A. A. El-Aziz, "SDN Architecture for Smart Homes Security with Machine Learning and Deep Learning," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 10, pp. 917–927, 2022, Doi: 10.14569/IJACSA.2022.01310108.

[10] D. V. Jeyanthi and B. Indrani, "IoT Based Intrusion Detection System for Healthcare Using RNNBiLSTM Deep Learning Strategy with Custom Features," *Soft Computing*, vol. 27, pp 11915–11930, 2023, Doi: 10.1007/s00500-023-08536-8.

[11] H. Alkahtani and T. Aldhyani, "Intrusion detection systems for IoT-based smart environments: a survey," *Complexity*, vol. 2021, pp. 1–18, 2021. Doi: 10.1155/2021/5579851.

[12] E. Gyamfi and A. Jurcut, "Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets," *Sensors*, vol. 22, no. 10, p. 3744, May 2022, Doi: 10.3390/s22103744.

[13] A. Khraisat and A. Alazab, "A critical review of intrusion detection systems in the Internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges," *Cybersecurity*, vol. 4, pp. 1–27, Dec. 2021, Doi: 10.1186/s42400-021-00077-7.

[14] H. J. Liao, C. H. Richard Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1. pp. 16–24, Jan. 2013. Doi: 10.1016/j.jnca.2012.09.004.

[15] M. J. Gatea and S. M. Hameed, "An Internet of Things Botnet Detection Model Using Regression Analysis and Linear Discrimination Analysis", *Iraqi Journal of Science*, vol. 63, no. 10, pp. 4534–4546, Oct. 2022, Doi: 10.24996/ijs.2022.63.10.36.

[16] H. K. Jameel and B. N. Dhannoon, "Gait Recognition Based on Deep Learning", *Iraqi Journal of Science*, vol. 63, no. 1, pp. 397–408, Jan. 2022, Doi: 10.24996/ijs.2022.63.1.36.

[17] Y. Hussain Ali *et al.*, "Optimization System Based on Convolutional Neural Network and Internet of Medical Things for Early Diagnosis of Lung Cancer," *Bioengineering*, vol. 10, no. 3, p. 320, Mar. 2023, Doi: 10.3390/bioengineering10030320.

[18] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, ACT, Australia, 2015, pp. 1-6, Doi: 10.1109/MilCIS.2015.7348942.

[19] A. Y. Hussein, P. Falcarin, and A. T. Sadiq, "Enhancement performance of random forest algorithm via one hot encoding for IoT IDS," *Original Research*, vol. 9, no. 3, pp. 579–591, 2021, Doi: 10.21533/pen.v9i3.2204.

[20] I. Ullah and Q. H. Mahmoud, "A Scheme for Generating a Dataset for Anomalous Activity Detection in IoT Networks," *In Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence*, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33 Springer International Publishing, pp. 508-520, 2020, Doi: 10.1007/978-3-030-47358-7_52.

[21] A. Alhowaide, I. Alsmadi, and J. Tang, "Towards the design of real-time autonomous IoT NIDS," *Cluster Comput*, vol. 26, pp. 2489–2502, 2021, Doi: 10.1007/s10586-021-03231-5.

[22] B. Roy and H. Cheung, "A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network," *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, NSW, Australia, 2018, pp. 1-6, Doi: 10.1109/ATNAC.2018.8615294.

[23] A. Kim, M. Park, and D. H. Lee, "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," in *IEEE Access*, vol. 8, pp. 70245-70261, 2020, Doi: 10.1109/ACCESS.2020.2986882.

[24] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, and A. Robles-Kelly, "Deep Learning-Based Intrusion Detection for IoT Networks," *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)*, Kyoto, Japan, 2019, pp. 256-25609, Doi: 10.1109/PRDC47002.2019.00056.

[25] N. Elmrabit, F. Zhou, F. Li, and H. Zhou, "Evaluation of Machine Learning Algorithms for Anomaly Detection," *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Dublin, Ireland, 2020, pp. 1-8, Doi: 10.1109/CyberSecurity49315.2020.9138871.

[26] S. Ullah *et al.*, "A New Intrusion Detection System for the Internet of Things via Deep Convolutional Neural Network and Feature Engineering," *Sensors*, vol. 22, no. 10, p. 3607, May 2022, Doi: 10.3390/s22103607.