A Novel Method to Detect Cyber Attacks in IoT/IIoT Devices on the Modbus Protocol using Deep Learning

Thierno Gueye
Northwestern Polytechnical University
Yanen Wang (wangyanen@126.com)
Northwestern Polytechnical University
Mudassar Rehman
Northwestern Polytechnical University
Ray Tahir Mushtaq
Northwestern Polytechnical University

Research Article

Keywords: Intrusion Detection System, Internet of Things, Deep Learning, Cyber Security, Cyber Attacks, Artificial Intelligence

Posted Date: June 22nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1762940/v1

Abstract

The dominant Intrusion Detection Models in the field of IoT/IIoT cybersecurity use network-based datasets. This paper proposes a way to perform Intrusion Detection with just the values in the registers of a device over a Modbus communication protocol. The proposed method of performing Intrusion Detection is by introducing an Embedding Layer into an otherwise simple neural network. Three neural networks were designed for the experiment and trained for both a binary classification of whether an attack occurred or not, and a multi-class classification of different types of attacks. The models performed better than the results previously reported without the use of Embedding Layers. This paper shows that a neural network with an Embedding Layer can effectively be used to model not only whether an attack occurred on a device, but also the class of attack that occurred.

1. Introduction

Internet of Things (IoT) are devices that contain sensors, some form of processors, and an active wireless connection (whether it be the internet, or a private network) [1] [2]. They come in the form of lights, doors, watches, speakers, and many other devices that are kept around the home. IoT devices have been specifically designed to make life easier for the home [3].

There is a new subtype of IoT devices that are called Industrial Internet of Things (IIoT). These are special IoT devices that are used in the industrial sense. It is defined as, “A system comprising networked smart objects, cyber-physical assets, associated generic information technologies and optional cloud or edge computing platforms, which enable real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, so as to optimize overall production value. This value may include; improving product or service delivery, boosting productivity, reducing labor costs, reducing energy consumption, and reducing the build-to-order cycle.” [4].

Both Internet of Things, and Industrial Internet of Things make collecting data, and monitoring systems much easier. The downfall of this is that there is a higher risk of cyber-attacks due to the lack of security measures for IoT and IIoT devices. This exposes the devices to malicious attacks from both inside and outside of the enterprise networks [5].

One of the latest news on IoT cybersecurity vulnerability is one where a security researcher tracked an iPhone user (without their consent) using a custom-made AirTag clone. Apple’s security measure to detect unwanted AirTags was easily circumvented. The custom-made AirTag also did not have any speakers so the person who was being stalked did not hear any beeping sounds [6].

The common ways in which computer systems battle cybersecurity concerns are by using firewalls, encryption, and intrusion detection systems, but since IoTs are fundamentally different from computers, the various security measure cannot be directly ported onto IoT devices [7] [8] [9]. There has also been a boom of IoT devices with about 23.14 billion connected devices in 2018, and it is projected to be as high as 75.44 billion connected devices in 2025 [10].

Along with the boom in the IoT space, there has also been a boom in the cybersecurity field concerning IoT devices. Kaspersky, the cybersecurity company, reported that in the first half of 2021, the number of breaches in IoT devices more than doubled to 1.51 billion from 639 million in the year of 2020 [11]. Kaspersky also reported that about 64% of organizations globally use IoT solutions, but 43% do not protect them completely [12].

One of the main reasons for such a high number of successful attacks is due to the fact that IoT devices are mostly resource-constrained, with limited computation power and storage capacity [7] [9].

Intrusion Detection Systems (IDSs) are software applications that detect intrusions in the form of network policy violations or malicious activities [13]. IDSs are used as a second line of defense to monitor the network that an IoT is connected to. An IDS detects any malicious activity that successfully evaded security perimeters like a firewall [14]. An effective intrusion detection system is required for keeping all the IoT/IIoT devices connected to the network in good health without any attacks.

There have been many datasets published to help with creating good intrusion detection systems. Datasets such as [15] [16] [17] are network-based datasets that contain packet-level and flow-level information or a combination of both to detect attacks on the IoT network. The main thing missing from these datasets is the actual readings from the IoT devices. The dataset from [18] is a relatively new dataset that contains logs of the sensor data.

The aim of this paper is to use the register values generated by the Modbus communication protocol to detect attacks on the IoT/IIoT network. This paper proposes the use of an Embedding Layer in a neural network to model the register values, which then is used to detect attacks in the network. There are three basic neural network models, namely, multi-layered perceptron (MLP), convolutional neural network (CNN), and recurrent neural network (RNN). All of the networks are used to model a binary-classification problem of whether an attack occurred or not, and a multi-class classification to detect what type of attack occurred.

The main contributions of this paper are listed as follows:

1. The application of an Embedding Layer in neural networks to improve its performance in detecting attacks on an IoT/IIoT network.
2. Improvement in the performance of simple, and light-weight models in both binary and multi-class classification of cyber-attacks on a network.

2. Related Works

The paper [18] first introduces the dataset that this paper will be working with. The authors of the paper also used various machine learning methods to model their own dataset. They used a logistic regression model, a linear discriminant analysis algorithm, a k-nearest neighbor algorithm, a classification and regression trees algorithm, a random forest, the Naïve Bayes algorithm, a support vector machine, and finally a long short-term memory variant of the recurrent neural network. They performed both a binary classification as well as a multi-class classification on the dataset. Most of their models performed really well for the binary

classification problem. The performance of their multi-class classification model is not as good as their binary classifier model. The highest accuracy, precision, recall, and f-score they obtained was 0.77, 0.77, 0.77, and 0.75 respectively. This paper will serve as a benchmark for this paper.

When it comes to machine learning based intrusion detection systems, there are many different papers available. Take for example the paper titled “An Evaluation of Machine Learning-based Anomaly Detection in a SCADA System Using the Modbus Protocol” [19]. This paper used the dataset simulated by [20], which is a simulated dataset of a gas pipeline used to move petroleum products to the market. They trained various machine learning models that worked almost perfectly in every situation. There is a fundamental difference in the type of data they used in their paper and the type of data used in this paper. The authors used a dataset that had many supplementary data like pump states, and solenoid states which are not applicable to the dataset that this paper uses.

Another paper titled “Evaluation of Machine Learning-based Anomaly Detection Algorithms on an Industrial Modbus/TCP Data Set” [21] also uses machine learning models to detect anomalous data. The dataset they used was presented in [22], where the authors simulated a controller network, consisting of a number of Master Terminal Unit (MTU) and a number of controllers. This paper is interesting due to the fact that when performing some preliminary dataset processing, the authors found a loophole where all the anomalous data had a lower packets per sec rate of data transfer. Hence, it made all the algorithms obsolete because a simple check of the data transfer rate would give a perfect performance. Nonetheless, they trained many different machine learning models which performed really well on the dataset.

A paper titled “A Deep Learning Approach for Intrusion Detection System in Industry Network” [23] simulated their own dataset which was a network traffic dataset. They trained a linear neural network with an accuracy of 99.9%.

In summary, there are many machines learning based intrusion detection models. They are mostly trained on network data, and almost all data available on this is simulated in a laboratory. There is a lack of actual deep learning models that perform on par or better than traditional machine learning models. The previous works show that it is possible to get good performing intrusion detection systems using machine learning models.

3. Materials And Methods

This section contains the materials and methods used in the experimentation, which includes the dataset, the neural network models, and all the hyperparameters (optimization function, and loss function) used in the training of the neural network.

3.1. Dataset

The dataset used in this paper is obtained from the TON_IoT Datasets which were publicly made available by the University of New South Wales at the Australian Defense Force Academy. This group of datasets is described in their website as a new generation of Industry 4.0/Internet of Things (IoT) and Industrial IoT

(IIoT) datasets for evaluating the fidelity and efficiency of different cybersecurity applications based on Artificial Intelligence (AI) [18].

The TON_IoT datasets contain data collected from several normal and cyber-attack events in a realistic experimental setup that was designed at the Cyber Range and IoT Labs to the UNSW Canberra [18]. The TON_IoT datasets are a group of many different datasets within it, including the Telemetry data of IoT/IIoT services, the Operating Systems logs, and Network traffic of the IoT network.

The datasets are labelled into two classes of whether an attack occurred, or if it is a normal operation. In the category of an attack happening, it is further divided into sub-classes of attack. There are nine different types of cyber-attacks are, scanning, DoS, DDoS, ransomware, backdoor, data injection, Cross-site Scripting (XSS), password cracking attack, and Man-in-The-Middle (MITM) attacks.

Out of all the data collected by the lab, this paper focuses only on the IoT/IIoT dataset. In the IoT/IIoT dataset, two main folders are named “Processed_datasets”, and “Train_Test_datasets”. The “Processed_datasets” contains all the processed, and filtered version of the datasets with their standard features and label in CSV format. The “Train_Test_datasets” folder contains the samples of the datasets to be used for training and testing new cybersecurity applications and machine learning applications.

This paper uses the “Train_Test_datasets” to train a neural network model. Inside the “Train_Test_datasets” folder, there are seven different IoT devices and the respective data collected from each of them. The IoT devices include Fridge, Garage Door, GPS Tracker, Modbus, Motion Light, Thermostat, and Weather module. The experiment used to collect all the data used a combination of physical and simulated IoT/IIoT devices. From all the datasets, the Modbus dataset is used for this paper.

The Modbus dataset was chosen because it is found in most smart manufacturing and industrial applications. The experimenters extracted the register type features from the Modbus service. The Modbus data that were extracted are described in Table 1. The Modbus data communications protocol will be discussed in more detail in the next section.

Table 1
Features of the Modbus Dataset

Features	Description
ts	Timestamp of sensor reading data
date	Date of logging Modbus register's data
time	Time of logging Modbus register's data
FC1_Read_Input_Register	Modbus function code that is responsible for reading an input register.
FC2_Read_Discrete_Value	Modbus function code that is in charge of reading a discrete value.
FC3_Read_Holding_Register	Modbus function code that is responsible for reading a holding register.
FC4_Read_Coil	Modbus function code that is responsible for reading a coil.
label	Identify normal and attack records, where '0' indicates normal and '1' indicates attacks.
type	A tag with normal or attack sub-classes, such as DoS, DDoS, and backdoor attacks.

The Modbus dataset does not contain all the classes previously described. It contains six different classes which are Backdoor, Injection, Normal, Password, Scanning, and XSS. A brief explanation of each of these cyber-attacks is described below.

- **Backdoor** refers to any method by which an unauthorized user is able to get around normal security measures and gain high level user access (which is also known as root access) on a computer system, network, or software application [24]. This attack can be used by attackers to steal any type of data (personal, financial) that is stored in the computer system, or network. The attackers can also use this attack to further install additional malware, and it can also be used to hijack devices.
- **Injection** attacks are the type of attack where the attacker injects a code into a program or query, or the attacker injects malware onto a computer in order to execute remote commands that can read or modify a database. It can also be used to change data on a website [25].
- **Password Cracking Attack** is an attack where the attacker tries to crack the password of a computer system by either a brute-force attack, or dictionary attacks [26]. This attack can allow the attacker to bypass the authentication procedure and hence, compromise the IoT or IIoT device.
- **Scanning Attacks** are attacks where the attackers scan the devices to gather network information of the devices before launching sophisticated attacks. Some common scanning techniques include IP address scanning, port scanning, and version scanning [27].
- **XSS Attacks**, also called Cross-Site Scripting attacks, are a type of injection, in which malicious scripts are injected into benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user [28]. This attack is capable of compromising the data and authentication procedures between IIoT devices and a remote Web server.

An overview of the Modbus dataset is shown in Fig. 1.

It can be concluded from Fig. 1 that the dataset is very imbalanced. Even in binary-classification tasks, the ratio of class 1-to-class 2 will be 68:32.

3.2. Modbus Communication Protocol

Modbus is a data communications protocol that was published by Modicon in 1979. It was initially used in their programmable logic controllers, but now it has become the standard communication protocol [29].

The Modbus protocol is very popular in industrial environments because it is openly published and royalty-free. Part of its popularity is due to the fact that it is easier to maintain compared to other standards, and it has less number restrictions on the format of data to be transmitted.

The Modbus protocol uses character serial communication lines, Ethernet, or the Internet protocol suite to transmit data. Data can be transmitted between multiple devices connected to the same cable or Ethernet network. The ease of use of the Modbus protocol is apparent in its use in IoT applications where, for example, a motion sensor module and temperature module can both be connected to the same computer, via Modbus.

It is a master/slave protocol, meaning a device operates as a master with one or more devices operating as a slave. A master device can read and write the data on a slave device's register. Each slave in the network has a unique 8-bit device address that is used by the master to transmit data. The slaves do not respond unless its address is recognized. The response time of the slave should also be within a specified time, or the master will call it a "no response error".

The data packet transmitted always starts with the slave address, a function code, followed by a function code, and parameters defining what is being asked for.

Table 2 lists the object types that a Modbus server provides to a Modbus client device [30].

Table 2
The object types that a Modbus server provides to a Modbus client device.

Object Type	Access	Size	Address Space
Coil	Read-Write	1 bit	00001–09999
Discrete input	Read-Only	1 bit	10001–19999
Input register	Read-Only	16 bits	300001–39999
Holding register	Read-Write	16 bits	400001–49999

The address space of the object types is called Entity Address. The Entity Address is the starting address, which is a 16-bit value in the data part of the Modbus frame. It ranges from 0–65,535 (0000 – FFFF in hexadecimal numbers).

The Coils are 1-bit registers, that are used to control discrete outputs. It can be read or written. Discrete inputs are also 1-bit registers that are used as inputs, they can only be read. Input registers are 16-bit registers that are used for input, and it can only be read. Holding registers are the most universal registers, they are 16-bit registers where data can be read or written. This register can be used to hold many things including, but not limited to, inputs, outputs, configuration data, or any requirement that calls for the “holding” of a data [31].

The Modbus dataset that is used in this paper contains the four object types being logged in time.

3.3. Neural Network

This paper uses 3 different types of neural networks to learn the mapping between the Modbus Register inputs and the Attack type output.

Neural networks, also called artificial neural networks, fall under the topic Deep Learning. Deep Learning further fall under the topic of Machine Learning, and Machine Learning is a type of an implementation under Artificial Intelligence.

Neural networks contain layers of nodes (also called artificial nodes), and each of those nodes contain parameters for the engineer to tune. The parameters in the node are the weights, and biases. Engineers do not manually tune them, rather they run an algorithm to tune the weights and biases for them. For a neural network to work, it needs a lot of data, and the data should be all encompassing with regards to the use cases.

This sub-section will discuss about the embedding layer, different types of neural networks used, as well as the optimization function and loss function used.

3.4. Embedding Layer

The main contribution of the paper is the use of Embedding Layers to model the Modbus register values. The Embedding layer is almost exclusively used in the Natural Language Processing subject. The benefit that an Embedding layer provides with NLP tasks is that it provides a dense and low-dimensional vectors with its main benefit being generalization power [32].

This layer provides a simple way to convert words into vectors of real numbers. Before the Embedding layer had become so popular, there were various methods to convert words into vectors but none of them worked as well as the Embedding layer. This is due to the simple fact that the Embedding layer is trainable [33] whereas the other vectorization methods could not be differentiated with respect to the loss function, hence, it could not be trained.

The Embedding layer is a simple lookup table that stores the embeddings of a fixed dictionary and size [34]. In simple terms, it just maps a number to a table of vectors, then these vectors represent the number in the neural network. The representation vectors can be trained, so it will keep on getting better in representing the input vector. An ideal Embedding layer will encode the meaning of the word, and similar words will be closer in the vector space [35].

The inspiration to use the Embedding Layer is due to the type of data the Modbus dataset provides. Each register has a 16-bit number, which in turn represents a value between 0 and 65535. This dataset looks very similar to the Natural Language Processing datasets if the words are encoded with numbers ranging from 0 to the length of the total unique words. The Embedding layer will be able to map each register value to a vector space which is unique to it. The neural network will then be used to learn whether an attack has occurred by taking the 4 vector spaces of the 4 different registers. The Embedding layer will be able to map all the combination of register vectors that constitute an attack closer to each other in the vector space.

3.5. Multi-Layer Perceptron

A multi-layer perceptron (MLP) is a feedforward artificial neural network, and the first type of neural network this paper will use in the experiment.

It is a very simple fully connected class of feedforward neural network that maps an input vector to an output vector [36]. It will contain the following linear transformation:

$$y = x \cdot W + b$$

where, y is the output vector,

x is the input vector,

W is the weight vector,

b is the bias.

A MLP layer can linearly map an input to the output. Activation functions can be used to make the whole network learn non-linear functions. The activation function used in the hidden layer of our network is the Rectified Linear Unit (ReLU), except for the output layer where a log softmax function is used.

3.6. Convolutional Neural Network

Convolutional Neural Network (CNN) is a class of artificial neural network that is used mostly in the field of computer vision [37]. CNNs were inspired by the biological visual cortex [38] [39] where the neurons resemble the organization of neurons in the brains of animals.

For CNN to be most effective, the vectors should have a spatial aspect to them. This means that the vectors do not provide any information by themselves, but a group of vectors mean something. Think of a pixel of color, it does not have any meaning, but put millions of pixels together and you get an image. If the pixels of an image are moved in any way, it loses meaning again. This is the main reason why other neural networks could not map images as effectively. CNNs inherently takes the position of the input vector with respect to its neighbors into consideration when training the model.

Convolutional Neural Networks have shown to be good enough in NLP tasks, so this paper will also use a CNN network to test whether it is good enough to map the Embedded vectors from the Modbus dataset.

There are other layers used in conjunction with the CNN layer, which are the Max-Pooling Layer, and the Dropout Layer.

The Max-Pooling Layer applies max pooling over an input signal. This means that it takes a kernel size and within that window it only allows the maximum value to move forward while reducing the dimension of the vector. This makes sure that only the best represented features from the CNN are used by the neural network to learn. Max pooling is shown by the following equation:

$$out\left(N_i, C_j, h, w\right) = \max_{m=0, \ldots kH-1} \max_{n=1, \ldots, kW-1} input\left(N_i, C_j, stride[0] \times h + m, stride[1] \times w + n\right)$$

where, N is the number of batch size,

C is the number of channels,

h is the hidden vector,

w is the weight vector

The Dropout Layer is used to randomly zero some of the elements of the input tensor with the probability p that the user sets. This is used to decrease the co-adaptation of the neurons in the neural network [40]. We call the neurons as co-adapted when more than one of its neurons have highly correlated behavior.

The three neural network layers are used in the following order: Convolutional Neural Layer, Max-Pooling Layer, and Dropout Layer.

3.7. Long Short-Term Memory

A long short-term memory (LSTM) is a type of Recurrent Neural Network (RNN) [41]. RNNs, in particular, were designed for time series data [42]. Time series data, for example the stock market data, are the types of data where it is collected over time. This type of data is highly correlated to its own previous data points in time. To map all the intricacies that come with mapping a time series data, the recurrent neural network was invented.

The inspiration to use RNN for this dataset is because RNNs like GRUs, and LSTMs are very good at modelling natural language [43]. Since the embedding layer was also primarily used for Natural Language Processing, it is only logical that RNNs should also work very well with the Embedded vectors of the Modbus dataset.

Out of the RNN, GRU, and LSTM layers that fall under the category of RNN layers, the LSTM layer is chosen because it is the most effective in modelling long sequences of data. This is due to the fact that LSTM has an exclusive Cell State that is not directly affected by the input data.

3.8. Loss Function

Loss function is one of the most important aspects of designing a neural network. This is because the loss function, also called a cost function, is a way in which the performance of the model is gauged. Lower the

loss, the better the model performs. If the loss of the model is low with the training dataset, but higher in the validation or testing dataset, it means that the model has overfit the dataset.

The loss function used in this experiment is the Negative Log Likelihood Loss (NLLLoss) [44], and the Binary Cross Entropy Loss (BCELoss) [45]. Both the losses are good at training a classification problem with their respective number of classes.

The Negative Log Likelihood Loss is given by the equation below:

$$l(x, y) = \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n$$

Where, x is the input (predicted output),

y is the target (real output),

w is the weight,

N is the batch size.

The Binary Cross Entropy Loss is given by the following equation:

$$l(x, y) = L = \{l_1, \ldots, l_N\}^T$$

$$l_n = -w_n \left[y_n \cdot log x_n + \left(1 - y_n\right) \cdot log\left(1 - x_n\right) \right]$$

Where, x is the input (predicted output),

y is the target (real output),

w is the weight,

N is the batch size.

3.9. Optimization Function

The optimization function is used to update the weights and biases in the network. This function decreases the loss of the overall model gradually over many iterations. The optimization algorithm used is the Adam Optimization algorithm. It had been introduced in this paper [46].

4. Experimental Setup

For the experimental setup there are three different neural network architectures namely, The Linear Network, The CNN Network, The RNN Network. They are all simple, and easy-to-train models.

The aim of the experiment is to showcase how versatile the Embedding Layer is, not only in Natural Language Processing tasks, but also in tasks requiring embedding the register values of a microprocessor.

4.1. Dataset Cleaning

As mentioned in the previous section, the dataset used is the ToN_IoT Datasets. The Modbus Dataset from the ToN_IoT Datasets is used.

Since the data was taken from the “Train_Test_datasets” folder, it has already been filtered and cleaned beforehand. The dataset did not have to be cleaned because there were no missing values, nor were there any errors in the dataset.

The data was not normalized because the Embedding Layer takes in integer vectors that are not normalized. The classes in the dataset are shown in Table 3.

Table 3
Frequency of Classes in the dataset.

Attack Type	Number in each Class
Backdoor	5000
Injection	5000
Normal	35000
XSS	577
Password	5000
Scanning	529

It can be noted that the classes are very imbalanced, with about 68.5% of the dataset without any attacks. Since the data is very small already, there was no attempt to balance the classes. If the classes were balanced, there would only be about 3000 data points, which is not enough to be divided into the train, and test datasets.

The dataset was then shuffled, which did cause it to lose the time series element of the data but the models were meant to predict the attack based on just the register values at a current time. Then the dataset was divided into “train” and “test” datasets. The dataset had 80% of the total elements in the “train” dataset, and 20% of the elements in the “test” dataset. The train dataset will be used to train the model, and the test dataset will be used to test the final trained dataset. There will be no intermixing of the data points between the datasets. There is also no validation dataset because the total number of elements in the original dataset was already too small to divide it into further three datasets.

To make the experiment as legitimate to compare the different models, a seed value of 0 was chosen for all the libraries (PyTorch, NumPy, random, Pandas DataFrame Sampling) that had a random element to it.

4.2. Evaluation Metrics

The evaluation metrics used in the experiment are Accuracy, Precision, Recall, and the F_1-Score. There will also be a Macro Averaging done with each of the metrics.

The accuracy metric is the fraction of the correctly identified objects to the total number of objects. It is shown by the following equation:

Accuracy=\frac{TP+TN}{TP+TN+FP+FN}

Where, TP stands for True Positive,

TN stands for True Negative,

FP stands for False Positive,

FN stands for False Negative.

Precision shows the correctly identified objects to the total objects of that specific class. It is shown below:

Precision=\frac{TP}{TP+FP}

Recall is the correctly identified objects to the total objects that the model identified as being in that class.

Recall=\frac{TP}{TP+FN}

F_1-Score is the harmonic mean of both precision and recall. It is shown by the following equation:

{F}_{1} Score=2\times \frac{Reacll\times Precision}{Recall+Precision}

The Macro Averaging of Accuracy, Precision, Recall, and F_1-score is better for imbalanced datasets because it weights each of the calculations with the total percentage of the class appearing in the dataset.

4.3. Linear Model

The architecture of the Linear Model is shown in Fig. 2.

Figure 2 Architecture of the Linear Model.

Figure 2 shows all the layers in the model, as well as the dimensions of the dataset when it comes out of each layer. There is a ReLU activation function after each linear layer, except for the final linear layer where there is either a sigmoid activation function for a binary classifier or a log softmax activation function for a multi-class classification.

4.4. Long Short-Term Memory Model

The architecture of the LSTM Model is shown in Fig. 3.

Figure 3 Architecture of the LSTM Model.

Figure 3 shows the architecture of the LSTM Model. There is an Embedding Layer, two LSTM Layers, and two Linear Layers. It should be noted that an LSTM Layer outputs a Cell State and a Hidden State. The cell state is a vector representing all the cell states in the LSTM layer, while the hidden state is the final cell state from the last time-step. These two vectors are not used in this model, only the output vector from the LSTM is used.

In the LSTM Model, all the layers have a ReLU activation function, with only the Output Layer having either the sigmoid activation function for the binary classification, or the log softmax activation function for the multi-class classification.

4.5. Convolutional Neural Network Model

The architecture of the CNN Model is shown in Fig. 4.

Figure 4 Architecture of the CNN Model.

Figure 4 shows the architecture of the CNN Model. There is an Embedding Layer with three layers of CNN, Dropout, and Max Pool Layer in that order. There is a ReLU activation function after every Max Pool Layer, and in the final Output Layer there is either a log softmax activation function for multi-class classification or a sigmoid activation function for the binary-class classification.

5. Results

5.1. Binary-Class Classification

The first experiment carried out was a binary-class classification. The models were made to predict whether an attack occurred or not.

5.1.1 Linear Model

The Linear Model was trained for 10 epochs, with a batch size of 64. It trained fairly fast, and it can be trained and tested on a CPU if a GPU is not available.

The Confusion Matrix of the Linear Model is shown in Fig. 5.

Table 4 shows the results of the Linear Model.

Table 4
Results of the Linear Model.

Class	Precision	Recall	F_1-Score	Total No. Data Points
Normal	0.98	1.00	0.99	7074
Attack	0.99	0.96	0.98	3148
Macro Average	0.99	0.98	0.98	10222
Accuracy	0.9861			

The Linear Model performs really well on the whole dataset. Even though the classes were heavily imbalanced, it performed well. There is a lower score in detecting the attacks when compared with the “Normal” class, and this might be due to the imbalance of the classes.

The macro averages of the precision, recall, and f_1 score are 0.99, 0.98, and 0.98, respectively.

5.1.2 LSTM Model

The Long Short-Term Memory Model was also trained for 10 epochs, with a batch size of 64. The model can be trained on a CPU fairly fast.

The Confusion Matrix of the LSTM Model is shown in Fig. 6.

The Table 5 shows the results obtained by the LSTM Model.

Table 5
Results of the Binary LSTM Model.

Class	Precision	Recall	F_1-Score	Total No. Data Points
Normal	0.98	1.00	0.99	7074
Attack	0.99	0.96	0.98	3148
Macro Average	0.99	0.98	0.98	10222
Accuracy	0.9860			

The LSTM Model and the Linear Model perform identically in detecting the attacks on the network.

The good performance of this model might suggest that the input vector containing the register values in time series format might have encoded information that is capable of detecting attacks.

5.1.3 CNN Model

The Convolutional Neural Network Model was also trained for 10 epochs, with a batch size of 64. The model has less parameters to be trained, so it can be trained and tested on a CPU or a GPU.

The Confusion Matrix of the CNN Model is shown in Fig. 7.

The results of the CNN Model are shown in Table 6.

Table 6
Results of the Binary CNN Model.

Class	Precision	Recall	F_1-Score	Total No. Data Points
Normal	1.00	0.99	0.99	7074
Attack	0.97	0.99	0.98	3148
Macro Average	0.99	0.99	0.99	10222
Accuracy	0.9891			

The CNN Model performed the best out of all the models experimented with. The model has an almost perfect score in most of the metrics used to gauge the model.

In the binary-class classification problem, all the models performed really well. Even though the classes were unbalanced, it did not hold the model back that much. The best performing model is the CNN Model, with the Linear, and RNN Models performing almost similarly. This is proof that adding an Embedding Layer improved the performance of the whole model.

The results from this paper [18] which also used an LSTM model to train a binary classifier on the same dataset is shown in Table 7 for comparison.

Table 7
Results from [18]

Dataset	Accuracy	Precision	Recall	F-Score
Modbus	0.68	0.47	0.68	0.55

When comparing the two results, the proposed model of this paper performs much better. This can be attributed to the addition of a single Embedding Layer in the model. This shows that the Embedding Layer is good at embedding whether an attack occurred or not.

5.2 Multi-Class Classification

The second experiment carried out was a multi-class classification. The models were made to predict the 6 classes (Backdoor, Injection, XSS, Password, Scanning, and Normal) in the dataset at the same time.

Initially the motivation was to use neural networks to identify the type of attack while excluding the “Normal” cases but when the experiment was conducted there was no difference between the results obtained. It was

then decided to just keep all the data and train a multi-class classifier with the “Normal” cases included.

5.2. 5.2.1 Linear Model

The Linear Model was trained for 10 epochs, with 64 batch size. It is a small model so training can easily be done on a CPU if a GPU is not available.

The Confusion Matrix for the Linear Model is shown in Fig. 8.

The results from the Linear Model are given in Table 8.

Table 8
Results from the Multi-Class Classification Linear Model

Class	Precision	Recall	F_1-Score	Total No. of Data Points
Backdoor	0.97	0.94	0.96	986
Injection	0.98	0.98	0.98	974
XSS	0.96	0.89	0.92	105
Password	0.96	0.95	0.95	983
Scanning	0.91	0.63	0.75	100
Normal	0.99	1.00	0.99	7074
Macro Avg	0.96	0.90	0.92	10222
Accuracy	0.9806			

Firstly, the model performed really well even though the classes were heavily imbalanced. The model performed the best in identifying the “Normal” class, this may be due to the fact that there is a lot of data in the normal class to train the model.

The model performed the worst in identifying the “Scanning” class, this might be due to scanning being a very complicated attack which can easily hide their tracks. If that was not such as case, then the class “XSS” would have a similarly bad performance, but this is not so. Even though there is equally a smaller number of XSS attacks as scanning attacks, it the model performs much better at identifying the XSS attacks.

The Macro Average values were 0.96 for precision, 0.90 for recall, and 0.92 for F_1 Score. The averages were brought down due to the model not performing well on the “Scanning” class.

5.3. 5.2.2 LSTM Model

The Long-Short Term Memory Model was trained for 10 epochs, with 64 batch size. It is also a small model that can easily be trained on a GPU or a CPU. Each input data was in the sequence of Input Register, Discrete Value, Holding Register, and Read Coil. This is important because an LSTM layer reads the input as a serial input, which means that when it is processing the data of Holding Register, it will have already seen the data of the Input Register, and the Discrete Value and it will have updated its Cell State accordingly.

The Confusion Matrix of the LSTM Model is shown in Fig. 9.

The results obtained from the LSTM Model is shown in Table 9.

Table 9
Results of the Multi-Class Classification LSTM Model.

Class	Precision	Recall	F_1-Score	Total No. of Data Points
Backdoor	0.98	0.95	0.97	986
Injection	0.99	0.98	0.98	974
XSS	0.99	0.87	0.92	105
Password	0.98	0.93	0.96	983
Scanning	0.76	0.65	0.70	100
Normal	0.98	1.00	0.99	7074
Macro Avg	0.95	0.90	0.92	10222
Accuracy	0.9795			

The LSTM Model also performs really well. Most of the patterns seen in the Linear Model's results can also be seen here. The model performed the best in the "Normal" class just like the Linear Model.

The model again performed the worst in identifying the Scanning attacks. This model is more precise in identifying the XSS attack than the Linear Model, but it also has a lower recall score in identifying the XSS attack when compared to the Linear Model. Overall, the model performed very similarly to the Linear Model.

The fact that the performance of the LSTM Model is so good suggests that there is a time series relationship between the four registers that helps the model identify the types of attacks, which is what was originally assumed to be the case.

The Macro Averages were 0.95 for Precision, 0.90 for Recall, and 0.92 for the F_1 Score. Again, the averages were brought down due to the model not performing better in identifying the scanning attacks.

5.4. 5.2.3 CNN Model

The Convolutional Neural Network Model was trained for 10 epochs, and it had a batch size of 64. This model was also a light-weight model that can easily be trained and tested on a CPU if GPUs are unavailable.

The Confusion Matrix of the CNN Model is shown in Fig. 10.

The Table 10 shows the results of the CNN Model.

Table 10
Results from the Multi-Class Classification CNN Model

Class	Precision	Recall	F_1-Score	Total No. of Data Points
Backdoor	0.78	0.95	0.86	986
Injection	0.94	0.97	0.96	974
XSS	0.00	0.00	0.00	105
Password	0.67	0.93	0.78	983
Scanning	0.00	0.00	0.00	100
Normal	1.00	0.94	0.97	7074
Macro Avg	0.57	0.63	0.59	10222
Accuracy	0.9247			

This result is the perfect example of what can go wrong when the training dataset is not balanced. This model performs almost perfectly in identifying the “Normal” class, but its performance significantly deteriorates when the training size of the classes decreases. It should also be noted that the CNN Model took the longest to converge.

The model did not even recognize a single attack of the “XSS” and “Scanning” classes. It performed bad, but not as much as in the “XSS” class, in the identifying the “Password” class with a precision, recall, and f_1 score of 0.67, 0.93, 0.78, respectively. It performed just good enough in identifying the “Backdoor” class with a precision, recall, and f_1 score of 0.78, 0.95, and 0.86, respectively.

The results of the CNN Model either suggests that the imbalance of the training dataset caused the difference in its performance, or that the register values cannot be mapped into a vector space with image-like features. To further prove which one is the correct answer, a new experiment can be conducted but this paper will not go into that experiment.

The models trained to classify all the different types of attacks perform very well. The best performing model is the Linear Model by a very small margin. The CNN Model does not perform that well on multi-class classification as it did on the binary classification problem. Apart from the CNN Model, both the Linear Model, and LSTM Model performed consistently across both the datasets.

The multi-class classification performed by [18] in their paper is a bit different from what is done in this paper. They concatenated all the different IoT/IIoT devices to create a single dataset which was then fed into their model. This paper did not do that because most of the data points were binary so it would not have been useful in the Embedding Layer which is the main focus of this paper. Nonetheless, the results from the [18] paper, where they have trained a LSTM multi-classification model is shown in Table 11.

Table 11
Results from [18]

Dataset	Accuracy	Precision	Recall	F-Score
Combined IoT/IIoT	0.68	0.64	0.68	0.63

Comparing the results of this paper, and the one from TABLE shows a very remarkable increase in accuracy. This might also be attributed to adding a single Embedding Layer in the model, or it might be because their dataset contained more noise in the form of other IoT/IIoT devices. Further investigation is warranted.

6. Conclusion

This paper looked at the feasibility of applying an Embedding Layer to embed the register values of devices connected over the Modbus Communication Protocol. The Embedding Layer is the most common way to embed words or characters in the field of Natural Language Processing (NLP). In the NLP field, a word is represented by an index, and that index is the input for the Embedding Layer which then embeds the word into a vector space.

The dataset used in this paper is the ToN_IoT Datasets created by the University of New South Wales at the Australian Defense Force Academy. Out of all the datasets in the ToN_IoT Datasets, this paper deals with the Modbus Dataset.

Three simple neural networks, namely, the Linear Model, the CNN Model, and the LSTM Model were designed with an Embedding Layer to embed all the register values of the four registers, namely, the Coil Register, Discrete Input Register, the Holding Register, and the Input Register.

The neural networks were then trained in binary-classification, and multi-class classification tasks. The binary classification consisted of “Normal” and “Attack” classes, while the multi-class classification consisted of “Backdoor”, “Injection”, “Normal”, “XSS”, “Password”, and “Scanning” classes of attacks.

After each of the neural networks were trained, the results are presented in this paper. All the models performed exceptionally well in the binary-classification task with the CNN Model performing just slightly better than the other two models. In the multi-class classification task, the Linear Model, and the LSTM Model performed exceptionally well, but the CNN Model did not perform as well.

When comparing the results with a previous paper [18] that did a similar experiment with neural networks, but without the Embedding Layer, the model proposed by this paper performed much better.

It can safely be concluded that embedding the register values of an IoT/IIoT device with an Embedding Layer can effectively be used to predict not only if an attack has occurred at that time, but also the type of attack that has occurred.

Declarations

Conflict of interest

Dear Editor-in-Chief,

A Novel Method to Detect Cyber Attacks in IoT/IIoT Devices on the Modbus Protocol using Deep Learning

The authors declare that they have no conflicts of interests or personal relationships that could have appeared to influence the work reported in this review paper. We affirm that the submission represents original work that has not been published previously and is not currently being considered or submitted to another journal, until a decision has been made. Also, we confirm that each author has seen and approved the contents of the submitted manuscript.

Corresponding author*

Name: Yanen Wang

Affiliation: Department of Industry Engineering, School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an China, 710072.

E-mail address: wangyanen@126.com

Submission date: 17/06/2022

All the codes used in this paper is available from

https://github.com/THIERNOGUEYE12/modbus_journal

The dataset is also publicly available from https://research.unsw.edu.au/projects/toniot-datasets

Author's Contribution

Author (Thierno Gueye) Conceptualization, Methodology, Validation, Data Curation, Formal Analysis, Investigation, writing – Original Draft Preparation.;

Author (yanen wang) Funding Acquisition, Project Administration, Supervision.;

Author (Mudassar Rehman) Data Curation, Visualization, Writing-Review & Editing

Author (Ray Tahir Mustaq) Data curration, analysis, Writing-Review & Editing

Funding:

The research is Funded by the current study was funded by The National Key Research and Development Program of China [Grant No.2019QY(Y)0502];

The Key Research and Development Program of Shaanxi Province [Grant No.2020ZDLSF04-07]

The National Natural Science Foundation of China [Grant No51905438];

The Fundamental Research Funds for the Central Universities [Grant No31020190502009]

The Innovation Platform of Bio fabrication[Grant No17SF0002];and

China postdoctoral Science Foundation[GrantNo2020M673471]

References

1. Gillis, A.S.: "What is the internet of things (IoT)?," 10 June 2022. [Online]. Available: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT
2. ITU:, "Internet of Things Global Standards Initiative," 14 July 2015. [Online]. Available: https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx
3. Kumar, S., Tiwari, P., Zymbler, M., "Internet of Things is a revolutionary approach for future technology enhancement: a review,"Journal of Big Data, p.111, (2019)
4. Boyes, H., Hallaq, B., Cunningham, J., Watson, T., "The industrial internet of things (IIoT): An analysis framework,"Computers in Industry,pp.1–12, (2018)
5. a., L.W., Raza, T.V.S., "SVELTE: Real-time intrusion detection in the Internet of Things,"Ad Hoc Networks, pp.2661–2674, (2013)
6. Bannister, A., "AirTag clone bypassed Apple's tracking-protection features, claims researcher," 22 February 2022. [Online]. Available: https://portswigger.net/daily-swig/airtag-clone-bypassed-apples-tracking-protection-features-claims-researcher
7. a., A.S.S.H.U.J., Sisinni, M.G.E., "Industrial Internet of Things: Challenges opportunities and directions,"IEEE Transactions on Industrial Informatics, pp.4724–4734, (2018)
8. a., W.H., Da Xu, S.L.L., "Internet of Things in industries: A survey,"IEEE Transactions on Industrial Informatics, pp.2233–2243, (2014)
9. R. S. M. C. T. K. a. S. C. d. A. B. B. Zarpelão, "A survey of intrusion detection in Internet of Things,"Journal of Network and Computer Applications, pp.25–37, (2017)
10. Alam, T., "A Reliable Communication Framework and Its Use in Internet of Things (IoT),"International Journal of Scientific Research in Computer Science, Engineering and Information Technology, pp.450–456, (2018)
11. Cyrus, C., "IoT Cyberattacks Escalate in 2021, According to Kaspersky," 17 September 2021. [Online]. Available: https://www.iotworldtoday.com/2021/09/17/iot-cyberattacks-escalate-in-2021-according-to-kaspersky/
12. Kaspersky, "43% of businesses don't protect their full IoT suite," 1 March 2022. [Online]. Available: https://www.kaspersky.com/about/press-releases/2022_43-of-businesses-dont-protect-their-full-iot-suite
13. Checkpoint, "Intrusion Detection System (IDS)," 10 June 2022. [Online]. Available: https://www.checkpoint.com/cyber-hub/network-security/what-is-an-intrusion-detection-system-ids/
14. a., J.H., Moustafa, J.S.N., "A holistic review of network anomaly detection systems: A comprehensive survey,"Journal of Network and Computer Applications, pp.33–55, (2019)
15. E. B. W. L. a. A. A. G. M. Tavallaee, "A detailed analysis of the KDD CUP 99 data set," in *IEEE Symposium on Computational Intelligence for Security and Defense Applications.*, Ottawa, ON, Canada, (2009)

16. Slay, N.M.a.J., : A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *The annual Military Communications and Information Systems (MilCIS) Conference*, Canberra, ACT, Australia, (2015)
17. A. H. L. a. A. A. G. I. Sharafaldin, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *The International Conference on Information Systems Security and Privacy*, (2018)
18. Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A., Anwar, A., "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems,"IEEE Access, pp.165130–165150, (2020)
19. Phillips, B., Gamess, E., Krishnaprasad, S., "An Evaluation of Machine Learning-based Anomaly Detection in a SCADA System Using the Modbus Protocol," in *ACM Southeast Conference*, Tampa, FL, USA, (2020)
20. Morris, R.V.Y.D.T., "A Testbed for SCADA Control System Cybersecurity Research and Pedagogy," in *Proceedings of the 7th Annual Workshop on Cyber Security and Information Intelligence Research*, Oak Ridge, TN, USA, (2011)
21. D., S.K., Simon Duque Anton, F.H.D.S., "Evaluation of Machine Learning-based Anomaly Detection Algorithms on an Industrial Modbus/TCP Data Set," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, (2018)
22. Antoine Lemay, J.M.F., "Providing SCADA Network Data Sets for Intrusion Detection Research," in *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, Austin, TX, USA, (2016)
23. Hijazi, A., Safadi, E.A.E., Flaus, J.-M., "A Deep Learning Approach for Intrusion Detection System in Industry Network," in *he first international conference on Big Data and Cybersecurity intelligence*, Beirut, Lebanon, (2018)
24. Malwarebytes, "Backdoor computing attacks," 7 June 2022. [Online]. Available: https://www.malwarebytes.com/backdoor
25. IBM:, "Injection attacks," 8 March 2021. [Online]. Available: https://www.ibm.com/docs/en/snips/4.6.0?topic=categories-injection-attacks
26. Nelso, T., Chaffin, M.: "Common Cybersecurity Vulnerabilities in Industrial Control Systems,". In: Control Syst. Secur. Program. Washington DC: Dept. Homeland Secur. (DHS) Nat. Cyber Secur. Division, Washington DC (2011)
27. Chen, Q.: "Chapter Three - Toward realizing self-protecting healthcare information systems: Design and security challenges,". In: Advances in Computers, pp. 113–149. Elsevier (2019)
28. A. J. A. S. J. K. I., J.W.D.W.A.W.R. E. Y. k. V. K. Jim Manico, "Cross Site Scripting (XSS)," 7 June 2022. [Online]. Available: https://owasp.org/www-community/attacks/xss/
29. Drury, B., Control Techniques Drives and Controls Handbook (2nd Edition), Institution of Engineering and Technology, (2009)
30. Modbus Organization, MODBUS APPLICATION PROTOCOL SPECIFICATION V.1.1b3, Modbus Organization, (2012)

31. Control, S., Minnesota, "Modbus 101 - Introduction to Modbus," 7 June 2022. [Online]. Available: https://www.csimn.com/CSI_pages/Modbus101.html
32. Goldberg, Y.: Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies). Morgan & Claypool Publishers (2017)
33. TensorFlow, "Word embedding," 10 June 2022. [Online]. Available: https://www.tensorflow.org/text/guide/word_embeddings
34. PyTorch, [Online]. Available: (2019). https://pytorch.org/docs/stable/generated/torch.nn.Embedding.html
35. Jurafsky, D., Martin, J.H.: Speech and Language Processing. Prentice Hall, Upper Saddle River, N.J. (2000)
36. Hastie, T.T.R.F.J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2009)
37. Valueva, M., Nagornov, N., Lyakhov, P., Valuev, G., Chervyakov, N., "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,"Mathematics and Computers in Simulation, pp.232–243, (2020)
38. Fukushima, K., "Neocognitron," 2007. [Online]. Available: http://www.scholarpedia.org/article/Neocognitron
39. Hubel, D.H., Wiesel, T.N., "Receptive fields and functional architecture of monkey striate cortex,"The Journal of Physiology, pp.215–243, (1968)
40. G. E. a. S. N. a. K. A. a. S. I. a. S. R. R. Hinton, "Improving neural networks by preventing co-adaptation of feature detectors," arXiv, (2012)
41. J. S. Sepp Hochreiter, "Long Short-Term Memory,"Neural Computation,pp.1735–1780, (1997)
42. Dupond, S., "A thorough review on the current advance of neural network structures,"Annual Reviews in Control,pp.200–230, (2019)
43. W. a. K. K. a. Y. M. a. S. H. Yin, "Comparative Study of CNN and RNN for Natural Language Processing," 7 February 2017. [Online]. Available: https://arxiv.org/abs/1702.01923
44. PyTorch, "N.L.L.L.O.S.S.," 10 June 2022. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.NLLLoss.html
45. PyTorch, "B.C.E.L.O.S.S.," 10 June 2022. [Online]. Available: https://pytorch.org/docs/stable/generated/torch.nn.BCELoss.html
46. D.P.a, Kingma, B.J., "Adam:A Method for Stochastic Optimization," arXiv, (2014).

Figures

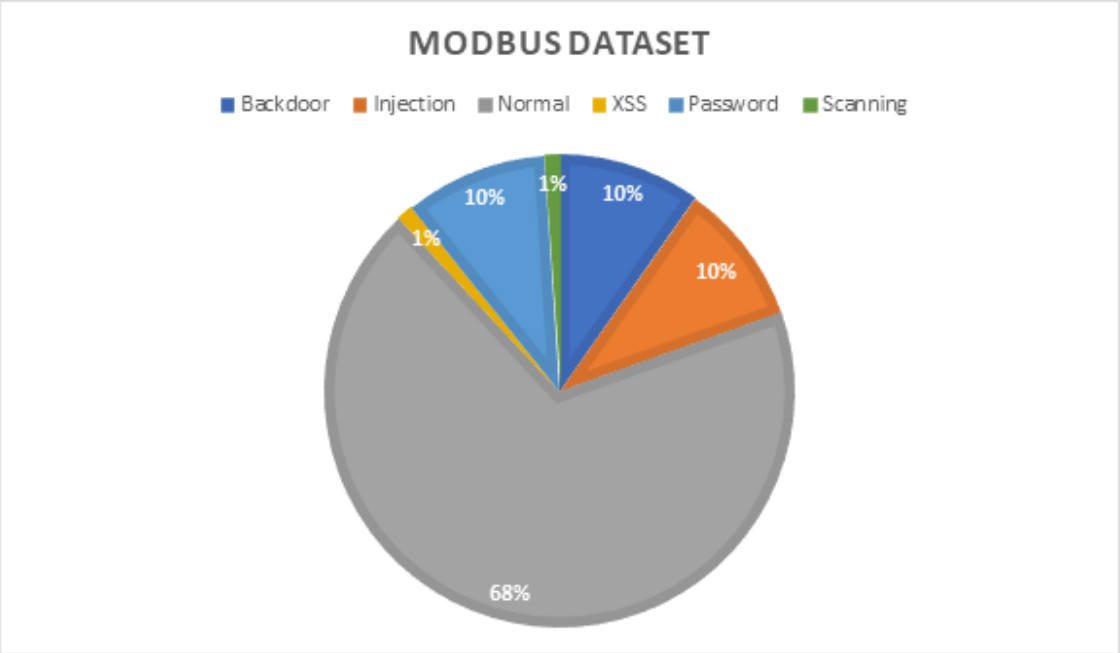

Figure 1

A pie chart showing the classes included in the Modbus dataset.

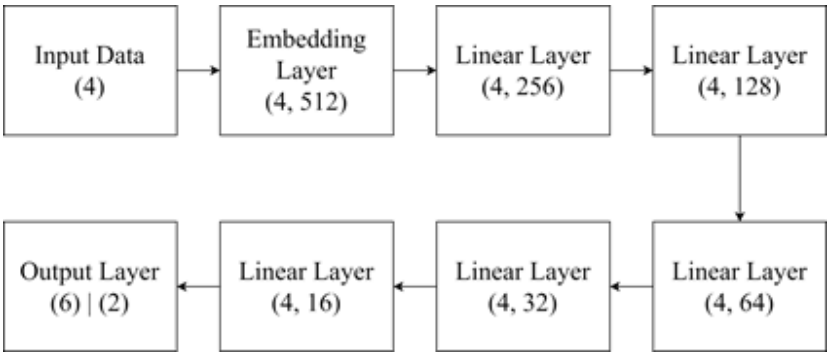

Figure 2

Architecture of the Linear Model.

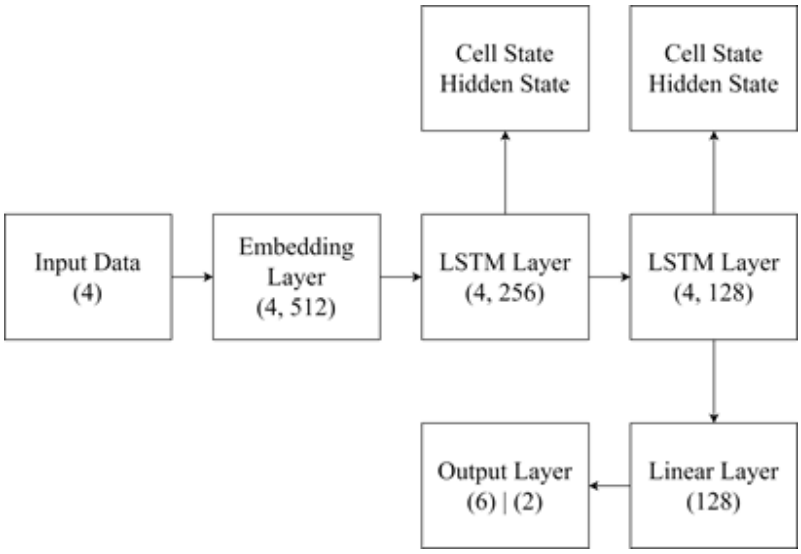

Figure 3

Architecture of the LSTM Model.

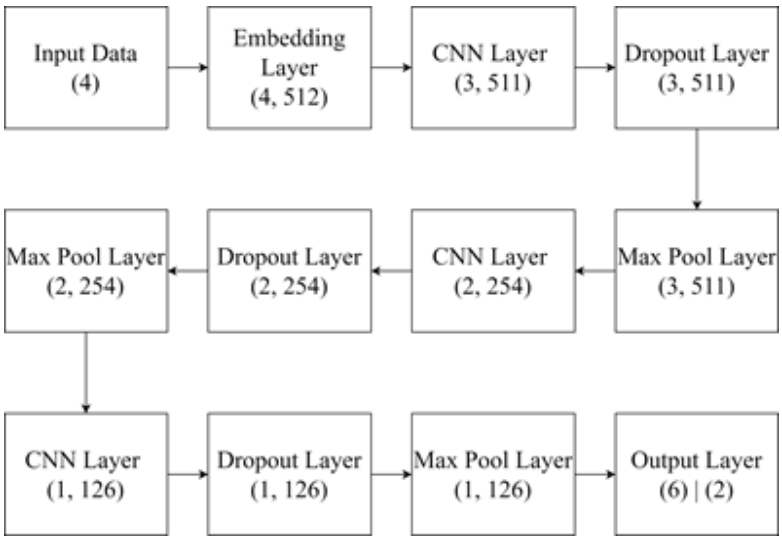

Figure 4

Architecture of the CNN Model.

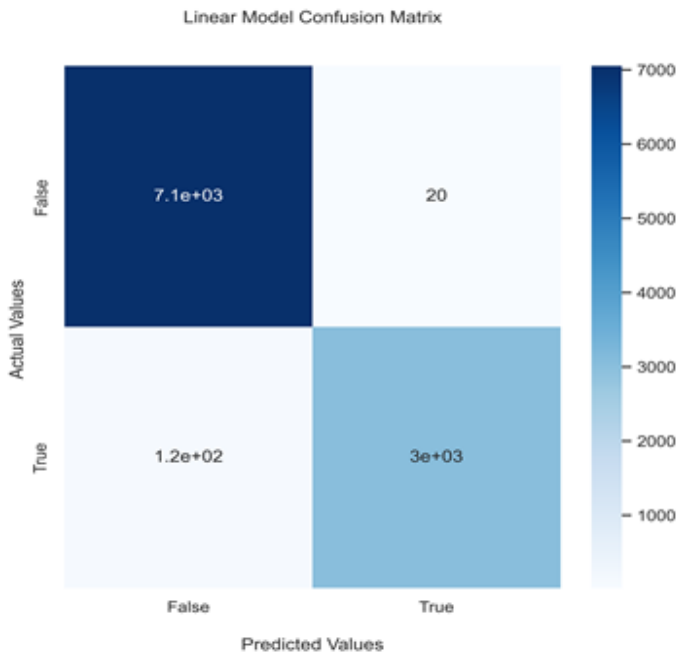

Figure 5

Confusion Matrix of the Binary Linear Model.

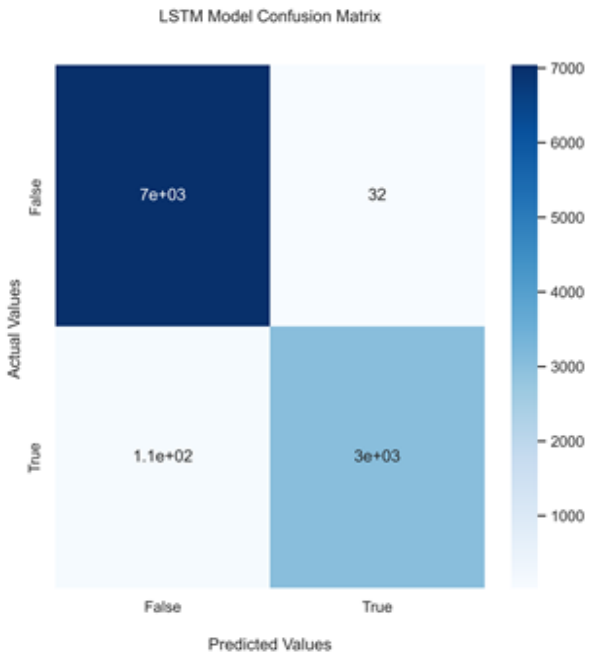

Figure 6

Confusion Matrix of the Binary LSTM Model.

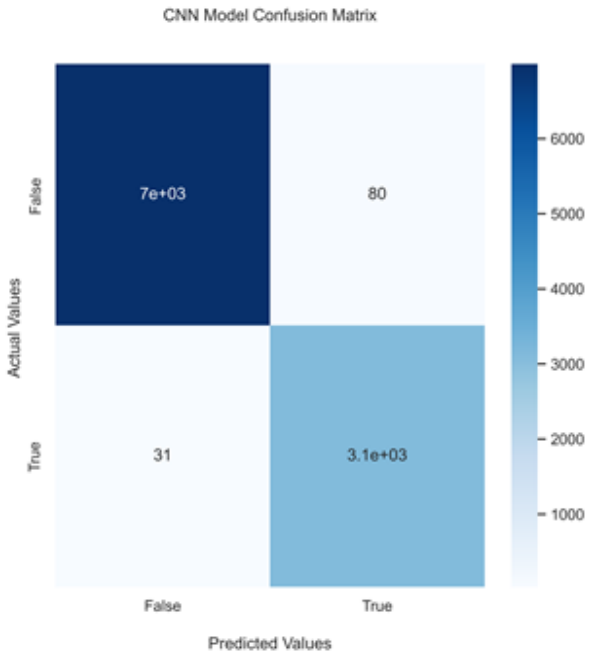

Figure 7

Confusion Matrix of the Binary CNN Model.

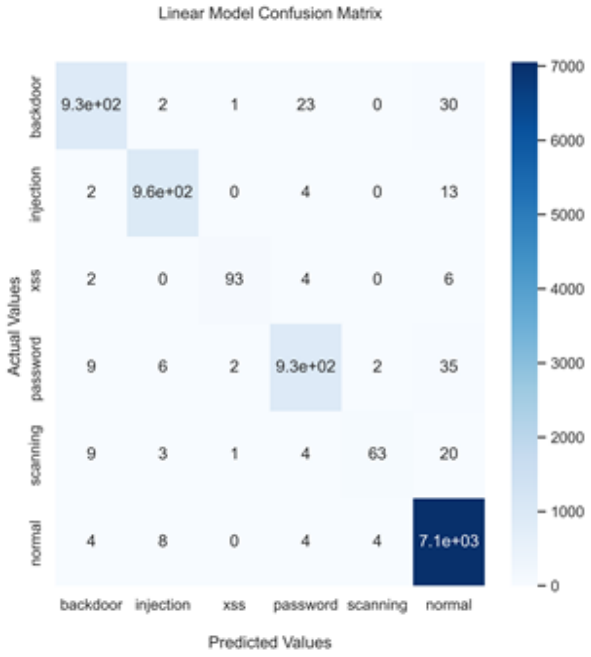

Figure 8

Confusion Matrix of the Multi-Classification Linear Model.

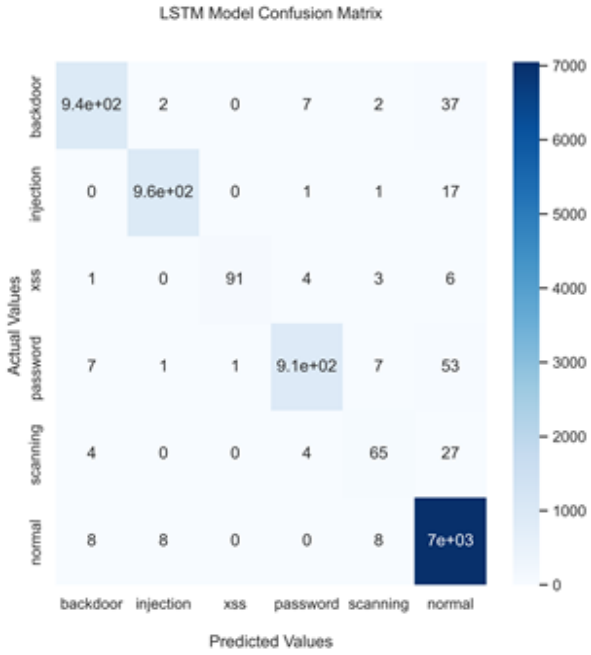

Figure 9

Confusion Matrix of the Multi-Classification LSTM Model.

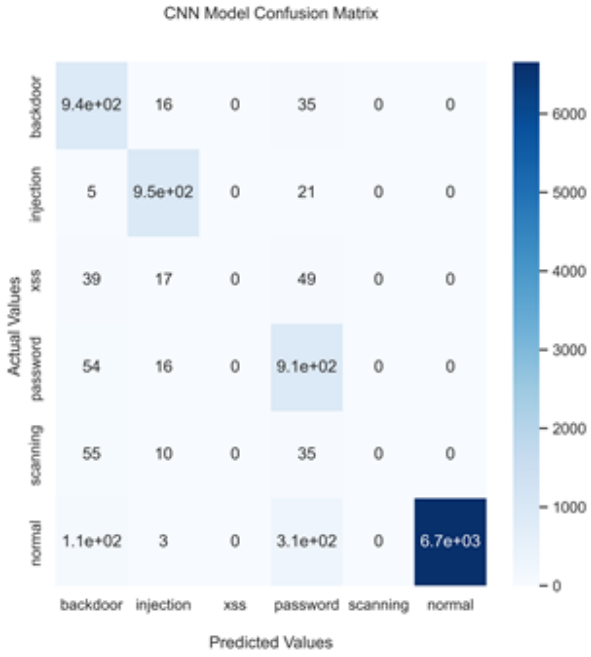

Figure 10

Confusion Matrix of the Multi-Classification CNN Model.