mathematics

MDPI

Article

Intelligent Deep Learning for Anomaly-Based Intrusion Detection in IoT Smart Home Networks

Nazia Butt [1], Ana Shahid [1], Kashif Naseer Qureshi [2], Sajjad Haider [1], Ashraf Osman Ibrahim [3,*], Faisal Binzagr [4] and Noman Arshad [5]

[1] Department of Computer Science, Faculty of Engineering and Computer Science, National University of Modern Languages, Islamabad 44000, Pakistan
[2] Department of Electronic & Computer Engineering, University of Limerick, V94 T9PX Limerick, Ireland
[3] Faculty of Computing and Informatics, University Malaysia Sabah, Kota Kinabalu 88400, Malaysia
[4] Department of Computer Science, King Abdulaziz University, P.O. Box 344, Rabigh 21911, Saudi Arabia
[5] Department of Computer Science, Bahria University, Islamabad 44000, Pakistan
* Correspondence: ashrafosman@ums.edu.my

Abstract: The Internet of Things (IoT) is a tremendous network based on connected smart devices. These networks sense and transmit data by using advanced communication standards and technologies. The smart home is one of the areas of IoT networks, where home appliances are connected to the internet and smart grids. However, these networks are at high risk in terms of security violations. Different kinds of attacks have been conducted on these networks where the user lost their data. Intrusion detection systems (IDSs) are used to detect and prevent cyberattacks. These systems are based on machine and deep learning techniques and still suffer from fitting or overfitting issues. This paper proposes a novel solution for anomaly-based intrusion detection for smart home networks. The proposed model addresses overfitting/underfitting issues and ensures high performance in terms of hybridization. The proposed solution uses feature selection and hyperparameter tuning and was tested with an existing dataset. The experimental results indicated a significant increase in performance while minimizing misclassification and other limitations as compared to state-of-the-art solutions.

Keywords: internet of things; smart homes; machine learning; intrusion; attacks; detection

MSC: 68T07

Citation: Butt, N.; Shahid, A.; Qureshi, K.N.; Haider, S.; Ibrahim, A.O.; Binzagr, F.; Arshad, N. Intelligent Deep Learning for Anomaly-Based Intrusion Detection in IoT Smart Home Networks. *Mathematics* **2022**, *10*, 4598. https://doi.org/10.3390/math10234598

Academic Editors: Daniel Ramotsoela, Adnan M. Abu-Mahfouz, Bruno Silva, Umair Mujtaba Qureshi and Zuneera Umair

Received: 18 October 2022
Accepted: 28 November 2022
Published: 4 December 2022

1. Introduction

The Internet of Things (IoT) is the interconnection of sensors, machines, objects, or other computing devices over the internet to communicate with the least human interference. Specific types of sensors are involved in obtaining information from physical entities, and after analysis, it is stored in local storage, which is then sent to cloud storage, where appropriate action is taken according to the information. The smart home is one example of connected home devices that can be controlled from anywhere at any time [1]. These networks are implemented as a global technology and have gained popularity among users. These networks have suffered from various challenges, among which security is one of the top challenges. Some of the common security attacks on these networks are Denial of Service (DoS), brute force, and ransomware. Intrusion detection is the concept of monitoring the traffic and classifying it as benign or malign. Intrusion detection in IoT networks can be signature-based, anomaly-based, or specification-based. In an anomaly-based intrusion detection system, normal behavior is recorded and stored as patterns and then compared with traffic patterns to see whether noise and other potential intrusions are anomalous or normal [2]. There are multiple techniques for anomaly-based intrusion detection systems (IDSs), such as data mining, statistical models, rule models, payload

models, protocol models, and signal processing models. Machine learning (ML) and deep learning (DL) techniques are used for anomaly detection to tackle attacks on a network with significant performance [3,4]. Figure 1 shows an IDS in a smart home network.

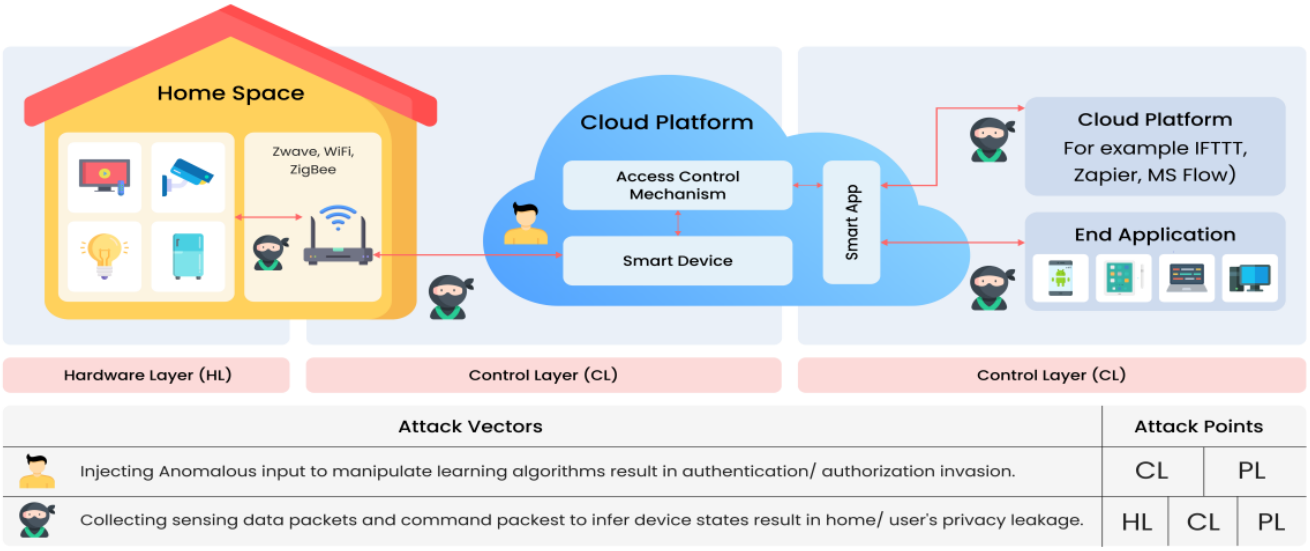

Figure 1. IDS in a Smart Home Network.

In signature-based techniques, detection is performed by matching signatures stored in the database with the signatures of the traffic flow. Anomaly-based IDSs with ML techniques are designed for attack detection [5,6]. However, the existing solutions are limited in function due to a lack of real-time dataset usage or an updated dataset, which leads to a degradation in performance, as attacks usually change their patterns and methods. Public datasets are available but not specifically created for smart home IoT networks. Some of the existing solutions suffer from overhead issues or increased time complexity [7]. Moreover, there are various issues observed, such as noise, overfitting, underfitting, complexity, and dimensionality, which lead to carelessness in data cleaning, feature extraction, selection, and normalization techniques. Accuracy is one of the significant parameters for measuring the model performance and needs to be maximized. However, there are differences in the test and training accuracy of many existing models, which suffer from overfitting or underfitting. As a countermeasure to these issues, appropriate methods are selected for data cleaning and hyperparameter selection. Hybrid ML/DL-based classifiers are used on recent real-time datasets for intrusion detection. To address the issues of existing solutions, the main objectives of this paper are as follows:

- To design a hybrid ML/DL model for intrusion detection in IoT-based smart home networks;
- To test the model on a real-time smart home IoT network dataset;
- To evaluate the performance of the proposed IDS by using appropriate evaluation techniques.

The rest of the paper is organized as follows: Section 2 provides a literature review on different methods for intrusion detection. Section 3 provides the complete methodology and the details of the design and development of the proposed model. Section 4 discusses the results and analysis of the proposed scheme and its comparison with different benchmark schemes. Section 5 concludes the research with potential future directions.

2. Related Work

The authors of [8] used different Support Vector Machine (SVM) techniques, namely, Linear, Quadratic, Fine Gaussian, and Medium Gaussian SVM, on the NSL-KDD dataset. Linear SVM involves a linear kernel and is used when data are linearly separable. If zs and

zt are data points, then the kernel in this scenario is Fine Gaussian SVM, which showed a clear difference between classes with the kernel sqrt(P)/4 (P is a predictor). Medium Gaussian showed fewer differences between classes when the kernel was sqrt (P). The analysis was conducted through ROC and a confusion matrix. Fine Gaussian SVM achieved high performance among other SVM techniques with a minimal error rate. However, this technique was not tested on a real-time dataset. A real-time dataset is needed for the future of IoT security in terms of intrusion detection. The authors of [9] used multinomial NB, a continuous dataset with discrete data, whereas, in Bernoulli NB, both discrete and categorical data are used, but the feature vector should be binary. The experimental setup used Gaussian NB, as the aim was to deal with more than two groups of attacks. Moreover, the sklearn library of Python was used in this technique to evaluate all parameters over the KDD dataset. PCA was also used to reduce the attributes and execution time of the KDD dataset over the KDD dataset, which exhibits better performance than the traditional Naïve Bayes. However, if the number of components is increased, then it affects the accuracy, which could be a challenge to overcome in the future.

The authors of [10] used four ensemble learning ML models on the RPL-NIDIDS17 dataset to overcome routing attacks. The ensemble learning models were Boosted Trees, Bagged Trees, Discriminant, and RUS Boosted Trees, and the dataset contained packet traces of Sybil, Clone ID, Black Hole, and Hello Flooding. The preprocessing of data was performed through cleaning, one-hot encoding, and scaling methods. Missing values were handled through cleaning, where one-hot encoding converted the nominal data into numerical form, and scaling was used to scale them between 0 and 1. After preprocessing, the data were converted into training and test samples, and four ensemble learning models were trained on the training set and then tested to see the expected outcomes in terms of attack detection (normal or an attack class), which indicated the good performance of EL ML models. However, lightweight solutions for securing smart nodes in IoT networks will be a target in the future.

The authors of [11] used the CART algorithm based on decision trees (DTs) to split the parent and child nodes based on the Gini index criterion. Ensemble classifiers utilized the results of multiple DTs through voting. This means that multiple classifiers are used for the selection of sample classes through voting rather than a single model. CART Decision Tree is famous for classification and regression. Combinations of three decision trees were used along with the NSL KDD dataset, which resulted in improved performance and accuracy while detecting a variety of attacks, such as DoS, R26, and Probe. However, the time required for modeling was increased due to the combination of trees, which could be ignorable or manageable after further research and testing.

The authors of [12] used KNN and LSTM for protection against illegitimate users in IoT networks. The proposed technique is based on three phases. In preprocessing, the normalization of data is conducted in R [0,1] through the min–max function. After preprocessing, feature selection is completed, through which the best features for intrusion detection are selected. Finally, KNN and LSTM are implemented to detect intrusion. The grouping of instances is managed according to the value of K and the distance measured. LSTM is used to minimize the error rate by calculating the difference between the expected outcome and the original outcome and then adjusting these calculations by varying the values of weights and biases accordingly. Simulations were run in MATLAB, and the BOT-IOT dataset was used. The mean detection time and Kappa stats were evaluated as parameters for the performance check. The detection time is the time needed to recognize the attack, whereas the mean value is used to set and balance TPR and TNR. A comparison of KNN and LSTM was also made, which determined that LSTM was not underfitted or overfitted, so it is a better-performing algorithm in the scenario. More attacks and a high number of instances in real-time IoT scenarios needed to be extended from this work. The authors of [13] improved the feature sets and association rule mining techniques, such as the FP growth algorithm, for the improvement of feature sets through the FP growth algorithm. The CNN model was implemented for Botnet attack detection with higher

accuracy than existing features. However, several attacks, a larger sample size, and more ML/DL models with tested thresholds are issues and challenges for future work.

The authors of [14] used Gray Wolf Optimization (GWO) and Particle Swarm Optimization (PSO) for feature extraction and selection. Random Forest (RF) was used as a classifier for intrusion detection through simulations in Python language on KDD 99, NSL-KDD, and CIC IDS 2019 datasets. RF has a high variance and low bias, but with the GWO-PSO-RF problem, the biasing problem was solved. Hence, it showed optimal results, but it needs to be implemented in a real learning environment implementing IoT security to ensure its performance, and a distillation technique needs to be applied to enhance its performance. The authors of [15] used the Q learning model to predict cyberattacks, and the problem of QoS control was managed by the RL learning algorithm. The RL-based model was also compared in terms of accuracy and precision with other DL models with significant performance, and AUC was also improved. However, an increased number of epochs caused a decrease in precision. More DL models with different calculations need to be trained to develop an effective IDS.

The authors of [16] used a protocol-based DID dataset, which reduced the number of features compared with the UNSW-NB15 and BoT-IoT datasets, and LSTM was used as a classifier with promising results. However, the misclassification of DoS and DDoS occurred due to similarities between their features, which needs to be mitigated in the future. The authors of [17] used AAE and GAN to prevent noise in the data and latent representation of the data, whereas KNN was used as a classifier on the IoT 23 dataset. Both techniques showed good performance, but GAN outperformed in terms of accuracy. Instances of minority classes were increased, and the feature resemblance method was used for the detection of new attacks. Table 1 shows the comparison of the discussed schemes.

Table 1. Comparison of ML/DL schemes for IDS.

Algorithm/Model	Dataset	Classification	Attacks	Performance Metrics	Achievements	Demerits
Nonlinear SVM [8] 2019	UNSW NB 15	Binary, multiclass	Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, worms	ACC, DR, FPR	Performed well in attack classification.	The difference in training and test accuracy reveals low bias and high variance, which may indicate overfitting.
PCA+NB [9] 2019	NSL-KDD	Multiclass	Probe, DoS, U2R, R2L	ACC, confusion matrix	Decreases the execution time by minimizing the number of components.	Degradation in accuracy if the number of components is increased.
EL [10] 2019	RPL-NIDDS17	Binary	The sinkhole, Blackhole, Sybil, Clone ID, Selective Forwarding, Hello Flooding, and Local Repair	ACC, AUC	Ensemble learning showed good performance in mitigating routing attacks.	High accuracy.
Hybrid DT [11] 2020	NSL-KDD	Binary	DOS, Probe, U2R, R26	ACC, precision, recall, F1 score	Improved accuracy.	Value of recall is lower than benchmark schemes.
KNN, LSTM [12] 2020	Bot-IOT	Binary	DDoS, DoS, OS, Service Scan, Keylogging, data exfiltration	ACC, DR, Kappa Stats, Geometric Mean	Counters overfitting and underfitting issues and has a faster learning rate with LSTM.	Difficult to select the suitable value of K when using KNN, and LSTM takes more time and power to train.

Table 1. *Cont.*

Algorithm/Model	Dataset	Classification	Attacks	Performance Metrics	Achievements	Demerits
FP growth algorithm, CNN [13] 2020	N-BaIOT	Binary	IoT Botnet attacks	ACC, precision, recall, F1 score	Method for improvement of the original feature set is given, and ACC is also good.	The number of classes and data size are small. The threshold value could have been tested. Only one attack class is countered.
Linear SVM [5] 2021	NSL-KDD	Multiclass	DoS, Probe, U2R, R26	ACC, error, ROC, confusion matrix	Fine Gaussian SVM performed well with the lowest error rate.	Lack of optimization of SVM.
GWO-PSO-RF [14] 2021	KDDCUP99, NSLKDD99	Binary class, multiclass	DoS, DDoS, Heartbleed, Botnet, Infiltration	ACC, precision, recall, F1 score, Support, confusion matrix	Balanced GWO-PSO-RF reduced biasing problem and DR of minority classes.	A real-time dataset was not used.
MDP [15] 2021	NSL-KDD	Multiclass	DDOS, DOS	ACC, precision, sensitivity, AUC	Gave the best precision and AUC curve.	Low accuracy and high epoch number could cause overfitting.
LSTM [16] 2022	UNSW-NB15, Bot-IoT	Multiclass	Dos, DDos	ACC, confusion matrix	Imbalanced data issues and overfitting were countered.	More methods for noise removal in data are used.
BiGAN + KNN [17] 2022	IOT23	Multiclass	Dos, Botnet Attacks	ACC, precision, recall, F score	Efficient detection of Zero-Day Attack.	Shared features in the data were not analyzed and required more instances of the minority class.

Discussion of the Previous Research

After reviewing state-of-the-art schemes, it is observed that there is a need to design a more efficient hybrid scheme to address the existing issues and changes. The availability of good data is the most important aspect of any ML/DL-based IoT-IDS, which is lacking in most of the discussed previous studies. Many of the discussed studies applied NSL-KDD or older datasets, which ignored new patterns and signatures. Feature engineering, selection, and hyperparameter tuning play a significant role in enhancing performance, whereas there is a limited amount of work on these aspects. The correct selection of the method to find rich features that represent characteristics in the best way and assist in classifying attacks by ML/DL models is not a part of many studies. There must be methods to reduce the dimensionality of data from high to low; many studies have been conducted, but some of them did not consider it. Noise in the data caused overfitting/underfitting issues and imbalanced data, which is the built-in case; therefore, there must be methods to handle them. There were many methods to cope with noise in previous studies, but they still faced overfitting/underfitting issues due to imbalanced bias and variance. Time and budget constraints are also noticed. Generalizable ML/DL models are also less common in state-of-the-art studies. The performance of solutions in terms of evaluating parameters such as Acc and precision must be enhanced by splitting the data fairly into training and test sets, whereas many studies did not justify this aspect. The hybridization of ML schemes with DL schemes has been implemented by many researchers for NIDS security. However, it must be explored on a real-time dataset to ensure a promising solution for IoT security.

3. The Proposed Hybrid ML/DL for IDS-IoT in Smart Homes

This section presents the research methodology for designing the proposed novel hybrid ML/DL scheme for intrusion detection with a real-time dataset mainly designed for

smart homes. After a detailed review of the literature, two datasets were selected, namely, CIC-IoT 2022 and UNSW-NB15. The CIC-IoT 2022 dataset is a public dataset generated for behavioral and profiling analysis. This dataset is tested on IoT devices where mostly Z-wave, ZigBee, and IEEE 802.11 standards are used. Two NIC cards are deployed in a 64-bit system, where the first card is utilized as a network gateway, and the second card is connected to an unmanaged network switch. The Wireshark tool is used for data packet capturing in the form of pcap files. The IoT devices are connected to a switch, whereas a smart hub (Vera plus) is used and connected to an unmanaged switch and serves as an IoT device. This switch is compatible with Z-wave, ZigBee, Wi-Fi, and Bluetooth. Network traffic is captured through six different types of experiments. Power, idle, interactions, scenarios, active, and attack states are used to capture traffic. RTSP brute force and flood are the types of attacks launched. The experimental setup is divided into three phases, which are responsible for data conversion, data preprocessing, the training–test split, model training, classification by the model, and evaluation. In the data conversion step, the files are captured in pcap format by default. However, ML models usually work with csv files. For data conversion, first, pcap files are downloaded, then Wireshark is downloaded and installed, and downloaded pcap files are opened with Wireshark. Then, pcap files are converted into CSV files by choosing CSV conversion from the file menu.

Data preprocessing is a critical task that enhances the quality of data to promote meaningful extractions from the data. In ML, it mainly refers to cleaning and organizing the raw data to make them suitable for training ML and DL models. In this research, data cleaning, normalization, one-hot encoding, and data reduction were used. For data cleaning, "preprocessing. Labelencoder" was used to transform labels into numerical forms [18].

The feature is a dimension reduction process in which the data sequences are represented in such a way that interesting parts are represented more effectively, and it reduces the calculation time of the algorithm. In ML, the most important part of data feature extraction and pattern recognition is that, based on these sequences, the training and testing process is performed. For the proposed model, one-hot encoding with panda's technique is used for feature extraction from the dataset. One-hot encoding allows the representation of data as categorical features using $log2(D)$ vectors. Here, D is the dimensions that are associated with one-hot encoding [19]. In this type of encoding technique, each categorical value is assigned a binary value and converted into a new column. The one-hot vector v is a binary vector of length m, where the only single entry will be 1, and the others will be zero. Here, v is the one-hot encoder vector, and m is the length of the vector. The feature vector of one-hot encoding is represented by Equation (1) [20].

$$\mathrm{v} \in \sum_{i=1}^{m} v1 = 1 \tag{1}$$

Here, v is the one-hot encoder vector, and m is the length of the vector. For the proposed study, there are five classes of data, namely, Power, idle, interactions, scenarios, and active, from the data sequences, which need to be converted into the type of attack the system is facing. The one-hot encoding method is used to convert the classes into a one-hot encoding vector. After completing data balancing and feature extraction, the data are split into training and test sets. The splitting of data prevents the model from underfitting and overfitting. The dataset input features are in numerical format, but the neural network has a problem when processing this type of data due to exploding and vanishing gradient problems [21]. This problem leads to poor model performance and low accuracy. To overcome this problem, the data features are scaled in the form of 0 and 1, where the maximum number will be 1, and the minimum number will be 0. The training–test split method is utilized to measure the estimated performance of ML/DL algorithms. A 70:30 split is the most commonly used ratio to train and test the model. The training set of a dataset is used to train the model, while the test set is used to evaluate the performance of the model based on different parameters.

For the proposed model, ML and DL algorithms are applied for intrusion detection, including Long Short-Term Memory network (LSTM), K nearest neighbor, and the decision tree (DT) algorithm. The LSTM model for deep learning consists of multiple layers. Each layer is inspired by the human neuron and processes the input. The input passes through various hidden layers and generates output. In the meantime, back-propagation algorithms take back the errors with them and learn from these errors. For every iteration of feed-forward and backward passes, the accuracy, precision, and recall are calculated. These learning features of machine learning algorithms allow them to learn by themselves using different learning procedures.

LSTM, KNN, and DT are hybridized with Adaboost for detecting the RTSP brute-force attack and UDP flood attack on CIC-IDS2022. The mathematics and logic behind the architectures of these methods are explained in the section below. LSTM is one of the most commonly used algorithms in artificial intelligence and deep learning methods. The algorithm is mostly used in the field of speech recognition, robotics, text recognition, handwriting recognition, etc. LSTM is the combination of the cell. Each cell contains an input gate, forget gate, and an output gate [22]. LSTM is used for IDS-IoT, where the input passes the information to the cell layers. It determines the extent of the information that is to be passed inside the cell. This gate obtains information from the previous cell. The forget gate in LSTM is responsible for carrying the information. This gate decides which information passes to the next layer and discards the information that is not very necessary for the cell. The output gate generates the output and passes the information to the next LSTM cell.

The algorithm of LSTM was developed to tackle the vanishing gradient problem. The vanishing gradient problem occurs in a DL model due to a greater number of layers. When there are a greater number of layers in the DL model, the product of derivation decreases, and the value of the loss function reaches zero. LSTM tackles this problem with the help of gates by increasing the space of the RNN model [23]. The gate in LSTM is responsible for the regulation of information from one cell to another cell. Different activation functions are applied in each gate. Figure 2 shows the LSTM architecture used for the proposed scheme.

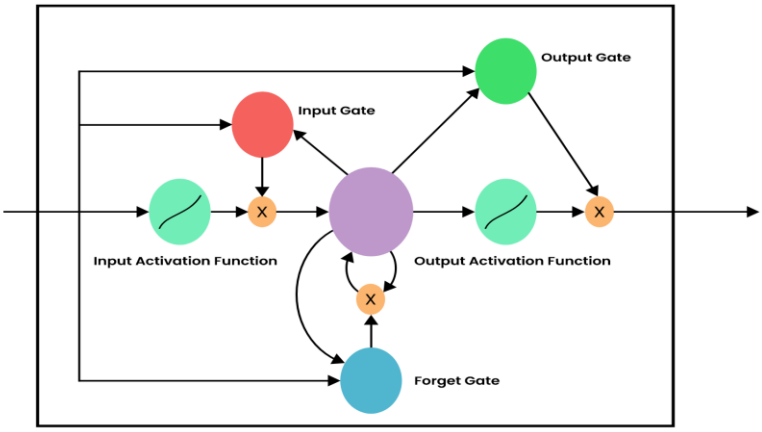

Figure 2. LSTM Architecture for IDS.

In Figure 3, x_t is the input at a specific time, and y_t is the output at a specific time t. f_t represents the forget gate, and i_t and NOT represent the input gate and output gate, respectively. Every cell of LSTM has three inputs, x_t, A_{t-1}, and B_{t-1}, and two outputs, b_t and h_t. The Tanh function is used to regulate the flow of the network. It maintains the value of the network between -1 and 1 [24]. The Tanh function allows the values of the gates to remain inside the boundaries. When the values pass from the network, it changes due to the large number of mathematical functions implemented inside the LSTM cell. For the proposed model, LSTM is used for intrusion detection from a real-time dataset. The sequential DL model is used in the current scenario. For the proposed model, one input

layer, along with three dense layers, one LSTM layer, and one output layer, is used. In each of the dense layers, the Relu activation function is used. Relu is a rectified linear unit that works on the min–max principle. The Relu function is adopted to maximize the value, as it is a nonlinear activation function used in deep neural networks. The working method of Relu is explained in Equations (2) and (3).

$$\text{Relu} = -\text{ev}\ (-\text{ev}, 0) = 0 \tag{2}$$

$$\text{Relu} = +\text{ev}\ (+\text{ev}, 0) = +\ \text{ev value} \tag{3}$$

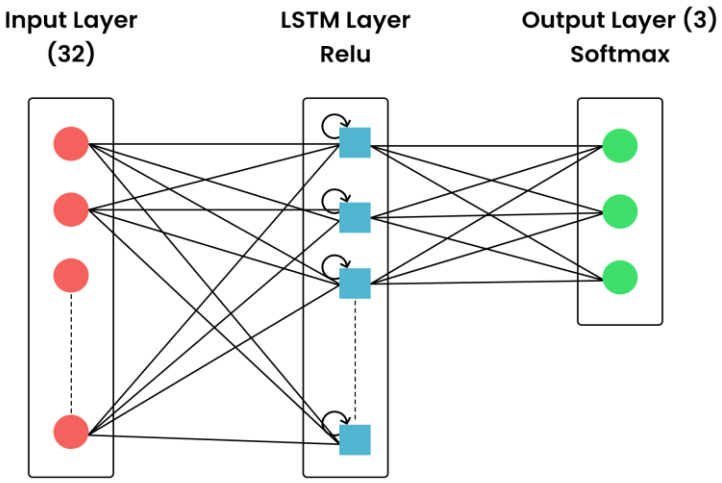

Figure 3. The probabilities of the instances of each class in LSTM (proposed).

The Relu activation function converts the classification into 1 if the value is greater than zero and makes the classification zero if the value is below zero. The dense layer of the model receives the input from all of the previous layers and classifies the output based on the output from the convolutional layers. The LSTM layer helps in the gradient flow. The LSTM layer is responsible for taking the data from the input layer, and the dense layers calculate the parameterized vector from these layers and apply activation functions element-wise on each gate. After applying the two dense layers to the dataset, the LSTM layer with RELU activation is applied. The SoftMax activation function is applied in the last dense layer. The SoftMax activation function is used to determine the probability of the class to which the input data belong. The output of the SoftMax activation is equal to the number of classes to which the data belong. This is also known as a probability distribution. The sum of all classes is equal to one. In our model, we have five output classes: Power, idle, interactions, scenarios, and active. The formula for calculating the SoftMax activation function is explained in Equation (4).

$$P\left(y = j \setminus \theta^{(i)}\right) = \frac{e^{\theta^{(i)}}}{\sum_{j=0}^{k} e^{\theta^{(i)}}} \tag{4}$$

In the equation, θ represents the one-hot encoding matrix, and j is the set of weights. Figure 3 shows the probability of each class in the proposed model.

KNN is an ML problem used for both classification and regression problems. This algorithm stores the data and classifies new data according to their similarity to the data classes. The algorithm for KNN is as follows:

1. Input different classes of sample data S, e.g., S(x) and S(y).
2. Select parameter k for the data.
3. Give a new data sample, x.

4. Determine the k-nearest neighbor of sample x by calculating the distance. It can be determined by the Euclidean distance.
5. Combine the classes of sample y into one class.
6. Find the output.

The decision tree (DT) is a supervised ML technique to address regression and classification problems. In DT, root nodes are used for input, which is also filtered through decision and leaf nodes to obtain the desired output [25]. Entropy is used for splitting control in DT and determines the respective classes of features from the information. The overall flow of the proposed model is shown in Figure 4.

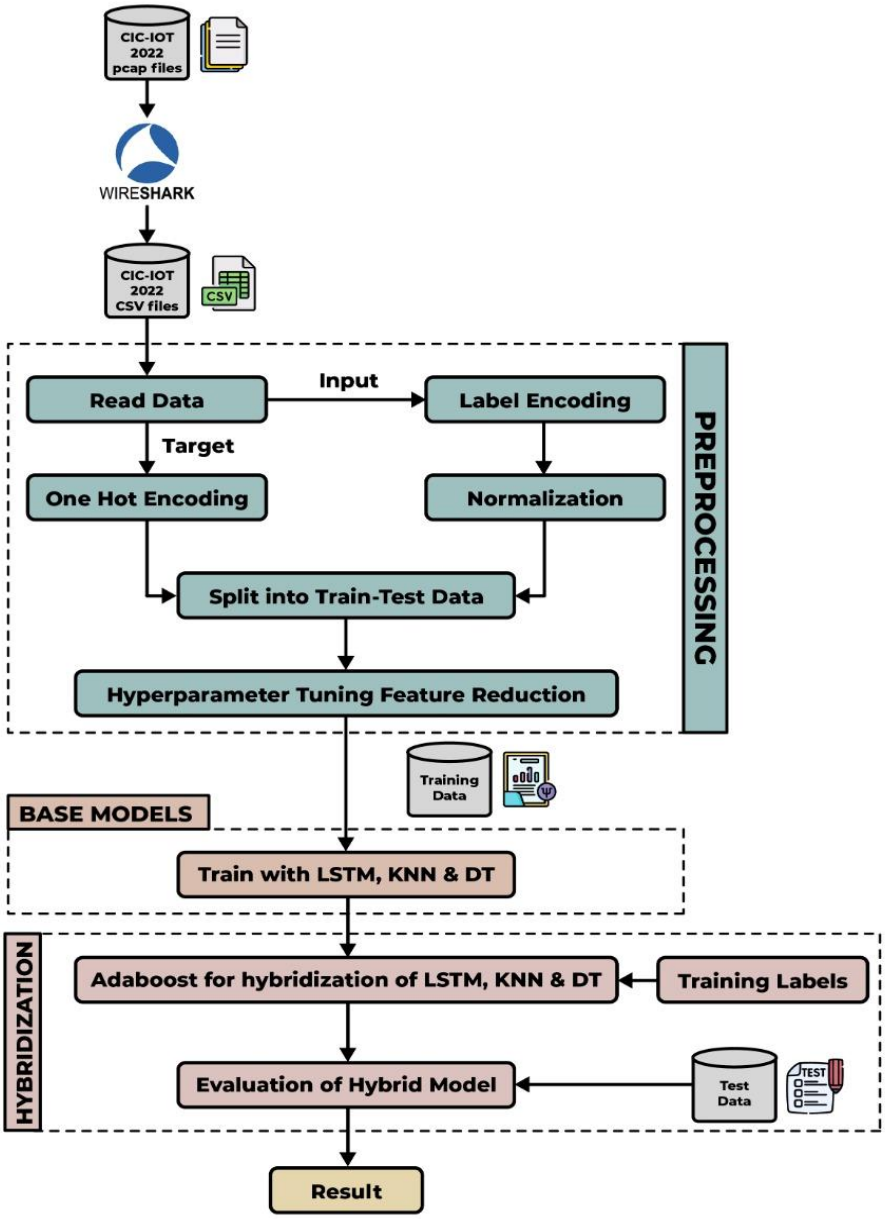

Figure 4. Experimental setup for hybrid KNN, LSTM, and DT.

Hyperparameter tuning with epochs is also explained in Figure 5.

Adaboost, also called Adaptive Boosting, is an ML technique used as an ensemble method to build a strong learner (predictive model). The predictive model or prediction is carried out by using the weighted average of weak classifiers. The predictive values are +1.0 or −1.0 for every new input instance and weak learner computation. For the sum of the weighted predictions, the ensemble model is used. For the positive sum, the first class

is used, else a second class will be used, such that the five classes' prediction values are 1.0, 1.0, −1.0, 1.0, and −1.0. According to the vote calculation, 1.0 is selected as the predictive value, whereas the 5 weak classifier values are 0.2, 0.5, 0.8, 0.2, and 0.9, respectively. These prediction results indicate that the output of −0.8 is an aggregate prediction of −1.0 or the second class. The pseudocode for Algorithm 1 is given as:

Algorithm 1 pseudocode caption

Initialize weights
for Each base learner, do:
Train base learners with a weighted sample.
Test base learner on all data.
Set learner weight with a weighted error.
$[\alpha = \frac{1}{2} \ln \frac{(1-\text{total error})}{\text{Total error}}]$
Update weights based on ensemble predictions.
end for

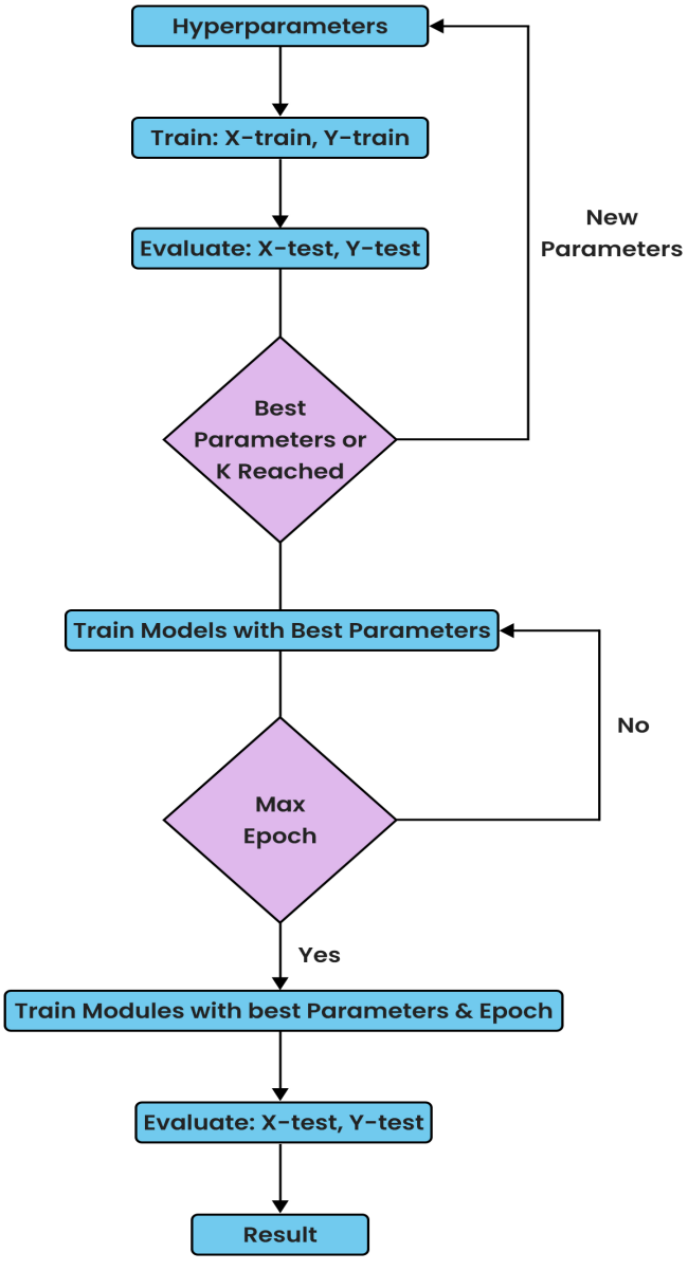

Figure 5. Hyperparameter Tuning.

In the proposed model, the outputs of LSTM, KNN, and DT are combined and given to Adaboost as a single input to classify intrusion, benign, and attack classes. The pseudo-code is given as in Figure 6:

```
Convert CIC-IOT2022 pcapfiles into csv by wireshark
Read CIC-IOT2022
One hot encoding
Data Normalization
Split into test data and train data
Load x_train, y_train, x_test, y_test
Hyperparameter Tuning with Keras Tuner
Feature Reduction with PCA
Chech PCA generated feature performance with NB

#---Train LSTM
LSTM <- LSTMClassifier with best parameters
LSTM training on x_train, y_train
LSTM evaluate on x_test, y_test

#---Train KNN
KNN <- KNNClassifier with best parameters
KNN training on x_train, y_train
KNN evaluation on x_test, y_test

#---Train DecisionTree
RF <- DTClassifier with best parameters
DT training on x_train, y_train
DT evaluation on x_test, y_test



#---Predict x_train to get output from KNN, RF, LSTM Classifiers
y_pred_rf <- LSTM.predicts on x_train
y_pred_rf <- KNN predictions on x_train
y_pred_rf <- DT.predicts on x_train


#---combine output of KNN, DT and LSTM Classifiers
y_pred <- combine y_pred_knn, y_pred_dt, y_pred_lstm

#----Now Hybradization
AdaBoost <- AdaboostClassifier with best params
Adaboost training on y_pred, y_train
Adaboost evaluation on x_test, y_test
```

Figure 6. Pseudo-code.

4. Results and Discussion

In the first experiment, the proposed hybrid ML/DL scheme was evaluated in terms of accuracy, precision, recall, and F1 score. A comparative analysis of the proposed scheme with other ML and DL schemes was also performed on two datasets, named CICIDS2022 and UNSW-NB15. The results show that the proposed scheme outperformed other benchmark ML/DL schemes. PCA was used to generate features concerning improved accuracy, and the hybrid KNN, DT, and LSTM model improved other performance metrics when

used as a hybrid combination by overcoming the single techniques' faults. The hybrid model was implemented utilizing the Keras Tuner to select the best hyperparameters so that the accuracy could be improved while decreasing the loss. Figure 7 shows the epoch vs. accuracy and loss.

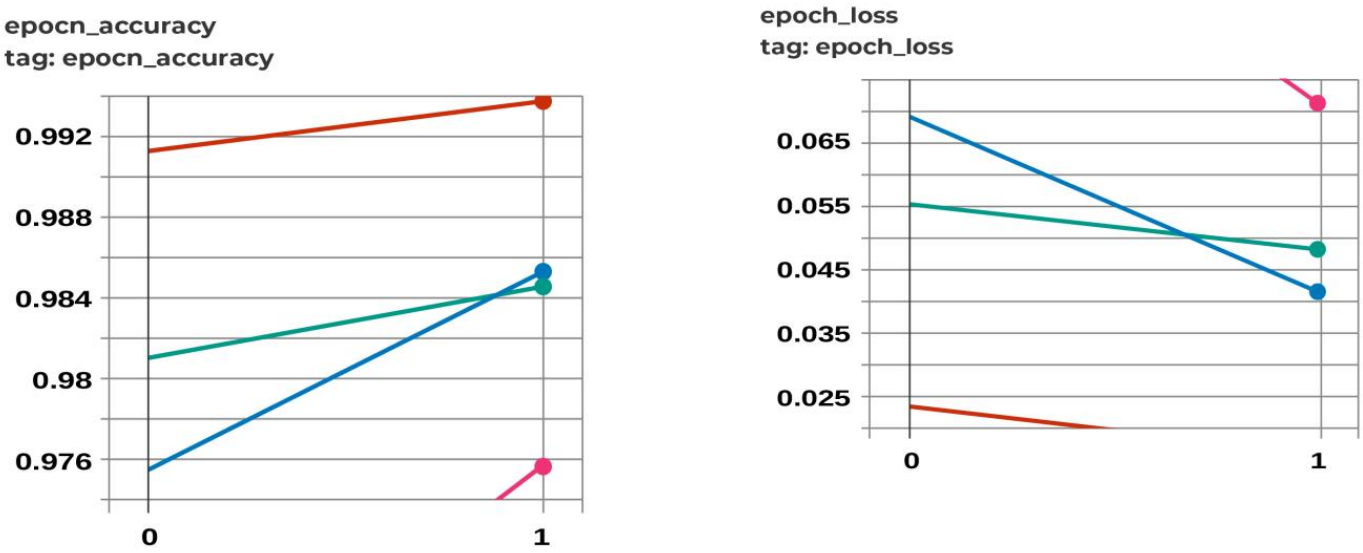

Figure 7. Epoch vs. Accuracy and Epoch vs. Loss.

Important features with scores are presented in Figure 8.

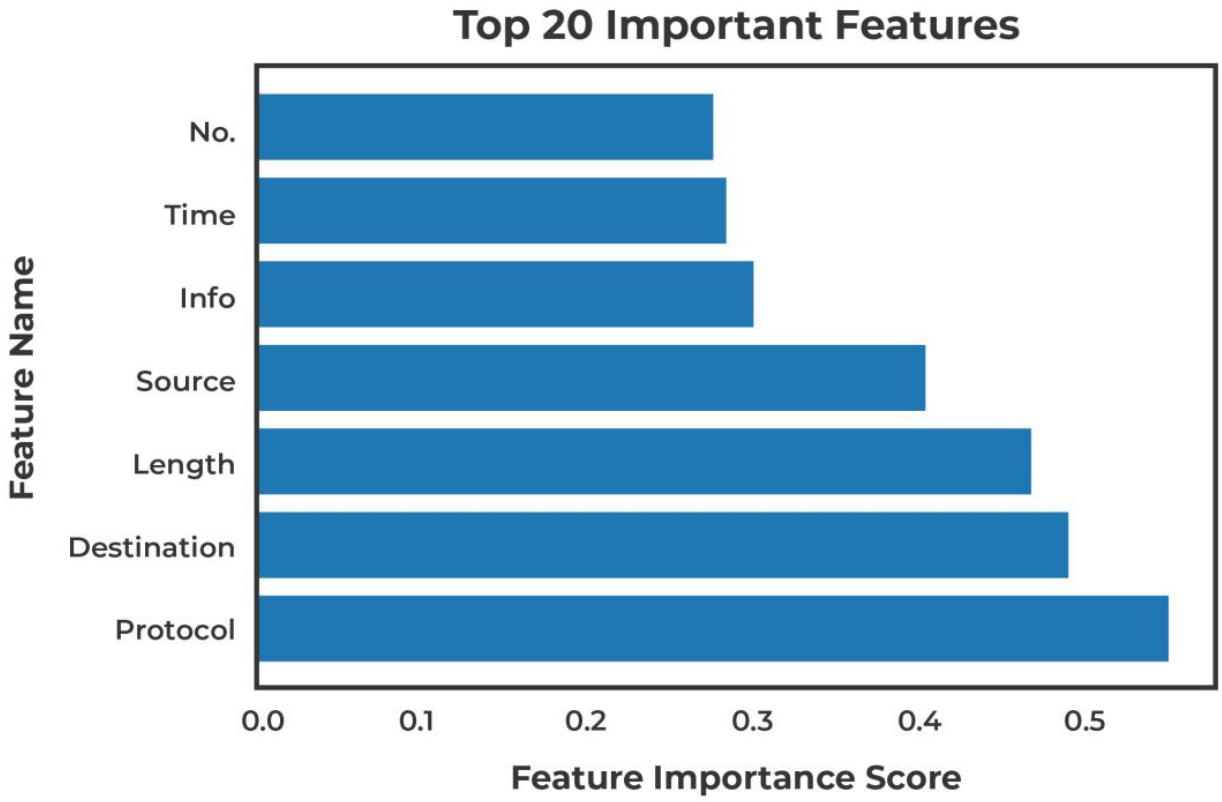

Figure 8. Top 20 Features.

The correlation matrix between ten important features is also presented in Figure 9.

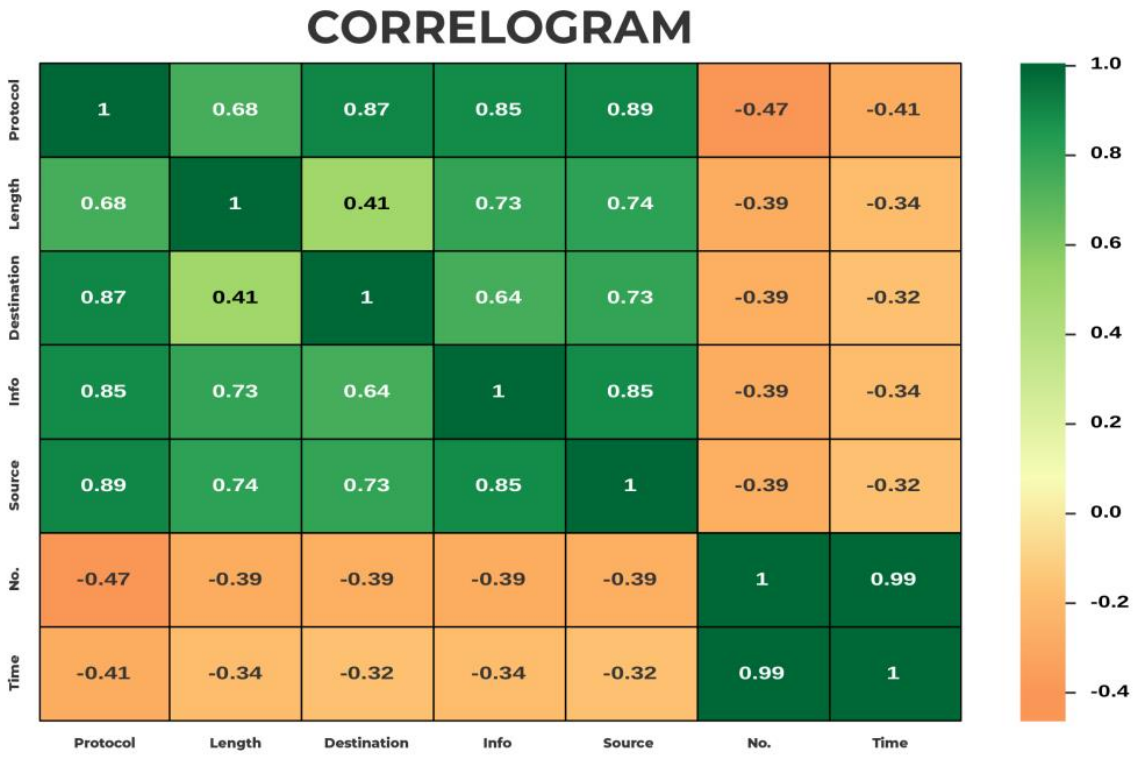

Figure 9. Correlogram.

PCA was used to generate auto features to improve performance, which was then tested using the Naive Bayes algorithm. The performance of these features in terms of accuracy is shown in Figure 10.

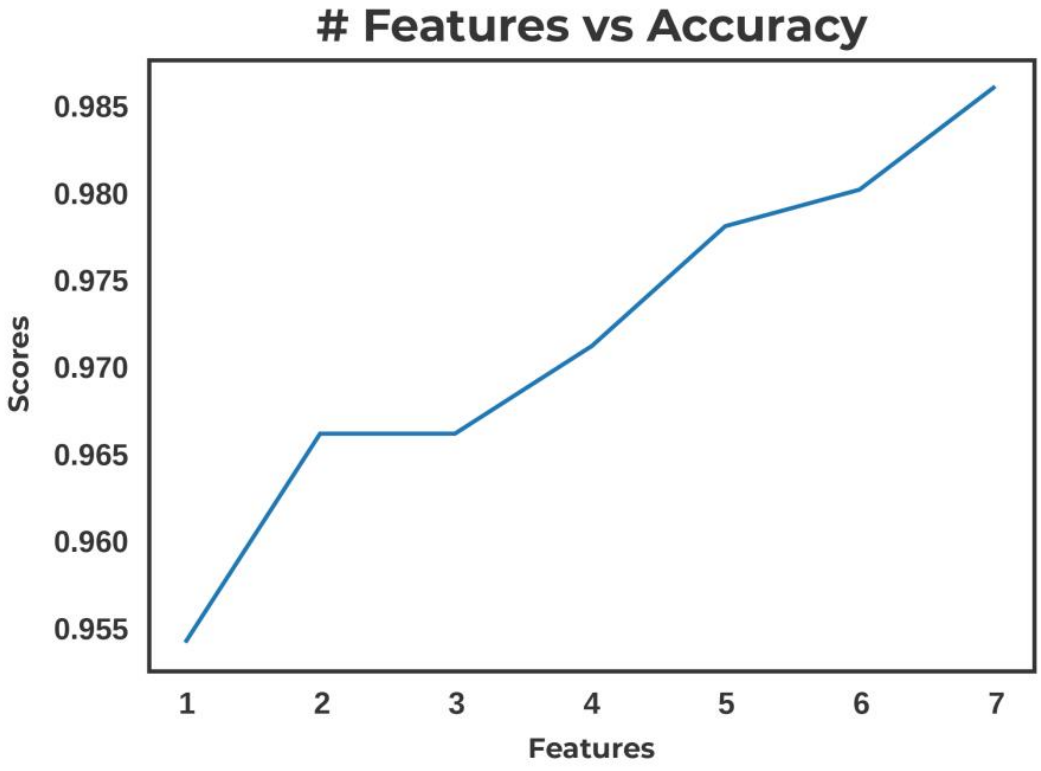

Figure 10. Features vs. Accuracy.

The confusion matrix shown in Figure 11 shows the performance of the ML/DL classification model.

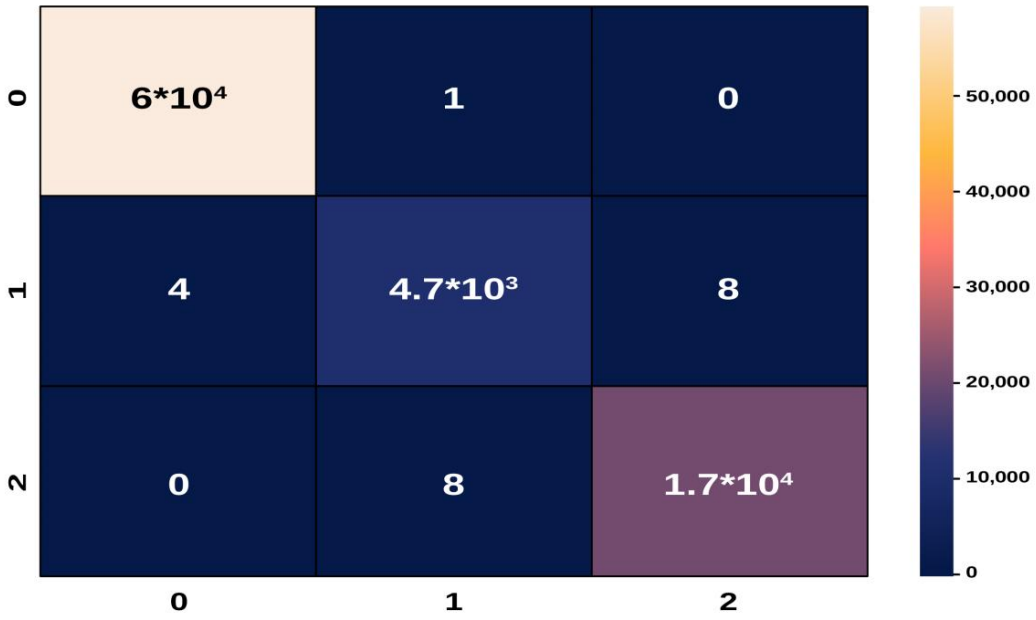

Figure 11. Confusion Matrix.

The proposed hybrid KNN, DT, and LSTM model was compared with other ML/DL schemes, namely, GRU, BiRNN, Bernoulli NB, Multinomial NB, RNN, Categorical NB, and Complement NB. Accuracy is the measure of the correct classification by ML or DL algorithms. Multiclass classification means the correct classification of an instance for each class. In the proposed solution, accuracy means the correct classification of an instance for Benign and two attack classes as well. Figure 12 shows the comparison of the proposed scheme with other schemes in terms of accuracy.

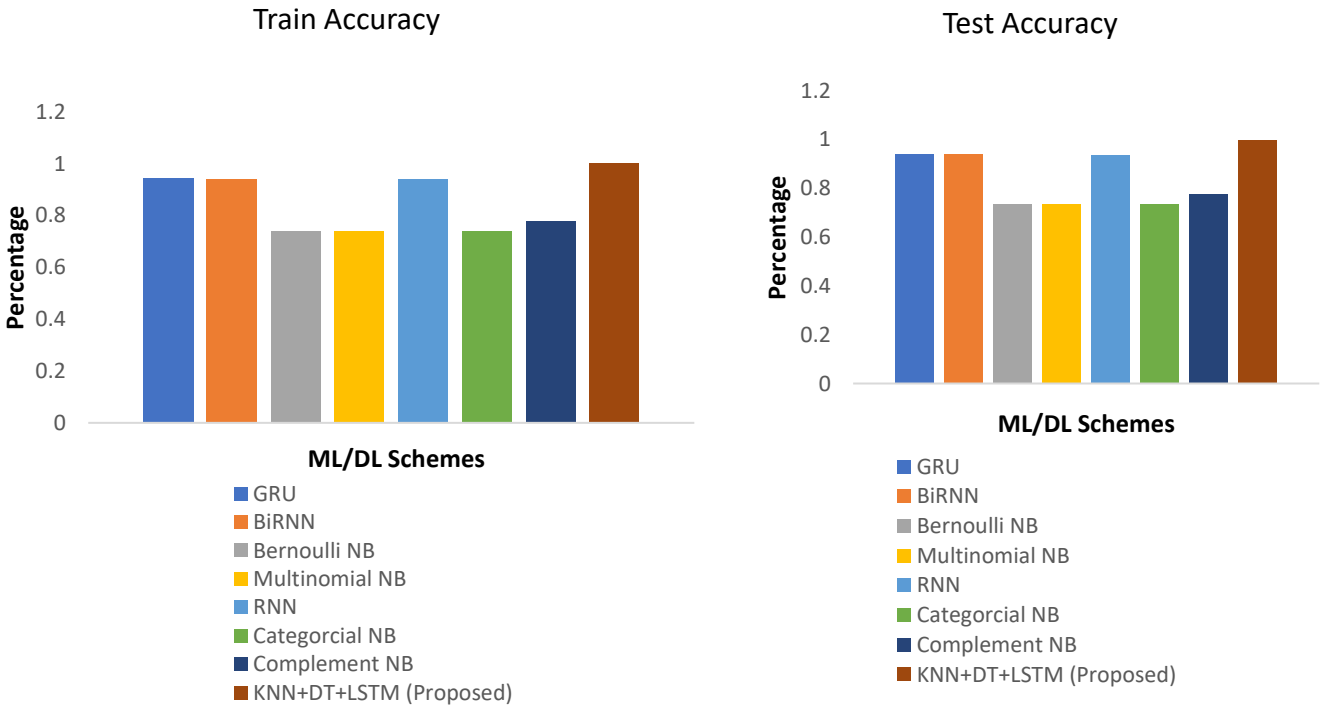

Figure 12. Training and Test Accuracy on Different Models.

The results reveal that the proposed KNN+DT+LSTM outperforms the other schemes. The gates involved in the LSTM architecture make it better for long-term dependencies and result in improved accuracy when hybridized with DT and KNN for attack detection in the proposed smart home scenario.

Precision is a measure of the reliability of the machine learning or deep learning model. It measures the model's accuracy in classifying an instance as positive. For the proposed hybrid model, if it can classify benign traffic correctly and does not misclassify attacks as benign, then the precision will be high. KNN is slow for the real-time detection of attacks, and therefore, DT and LSTM were hybridized to improve the precision and performance of IDS-IoT. A graphical representation of precision when compared with other schemes is given below. It shows that the proposed hybrid KNN, DT, and LSTM performed well. Figure 13 shows the training and test precision results of different models.

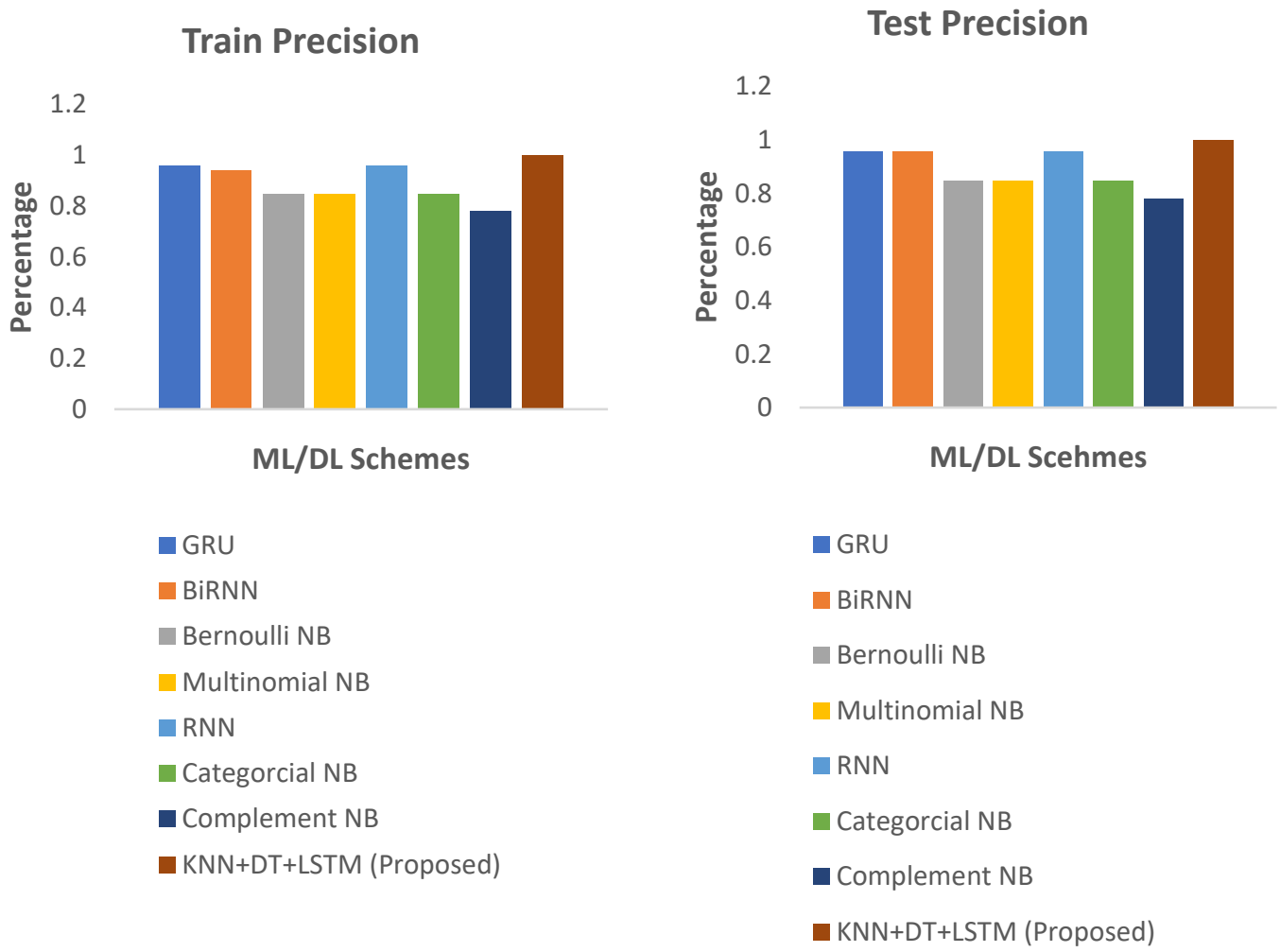

Figure 13. Training and Test Precision of Different Models.

Recall is the capability of classifying positive samples relative to the total number of positive samples. In the current scenario, it is the ability to detect benign samples. Figure 14 shows the training and test recall of different models.

The F1 score is the harmonic mean of precision and recall. It is used to combine classifiers with different precision and recall. Figure 15 shows the training and test F1 scores of different models. Tables 2 and 3 show the performance metrics of different ML/DL models with training and testing.

Figure 14. Training and Test Recall of Different Models.

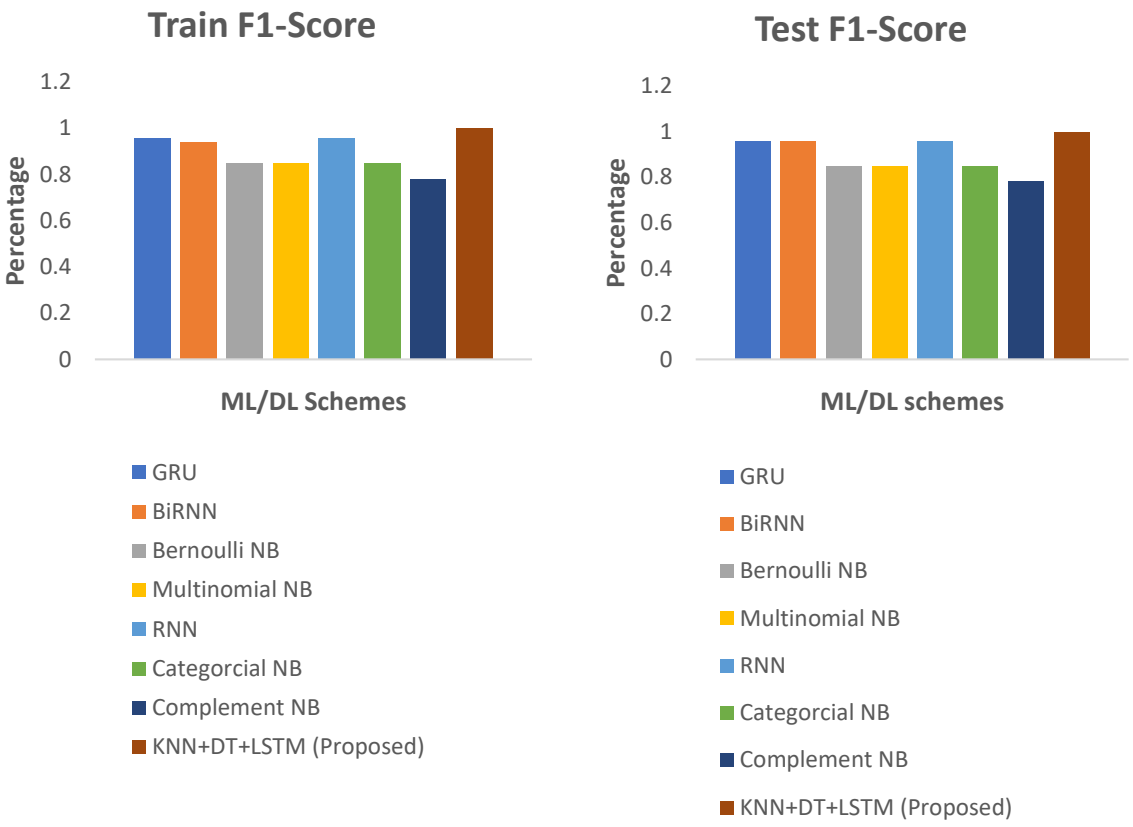

Figure 15. Training and Test F1 Scores of Different Models.

Table 2. Performance Metrics of Different ML/DL Models (Training).

Techniques	Accuracy	Precision	Recall	F1 Score
	Training			
GRU (Gated Recurrent Unit)	0.94165	0.98387	0.94165	0.9591
BiRNN (Bidirectional Recurrent Neural Network)	0.9401	0.98334	0.9401	0.94006
Bernoulli NB (Naïve Bayes)	0.73807	0.99985	0.73807	0.84919
Multinomial NB (Naïve Bayes)	0.73805	0.99995	0.73805	0.84924
RNN (Recurrent Neural Network)	0.93833	0.98586	0.93833	0.95891
Categorical NB (Naïve Bayes)	0.73801	1	0.73801	0.84926
Complement NB (Naïve Bayes)	0.7768	0.84577	0.7768	0.7801
KNN+DT+LSTM (Proposed)	1	1	1	1

Table 3. Performance Metrics of Different ML/DL Schemes (Test).

Techniques	Accuracy	Precision	Recall	F1 Score
	Testing			
GRU (Gated Recurrent Unit)	0.94161	0.98292	0.94161	0.95853
BiRNN (Bidirectional Recurrent Neural Network)	0.94006	0.98246	0.94006	0.95766
Bernoulli NB (Naïve Bayes)	0.7359	0.99977	0.7359	0.84772
Multinomial NB (Naïve Bayes)	0.73588	0.99998	0.73588	0.84783
RNN (Recurrent Neural Network)	0.9379	0.98547	0.9379	0.9585
Categorical NB (Naïve Bayes)	0.73586	1	0.73586	0.84783
Complement NB (Naïve Bayes)	0.77767	0.84608	0.77767	0.78095
KNN+DT (Decision Tree) +LSTM (Proposed)	0.99974	0.99974	0.99974	0.99974

The last experiment is with UNSW-NB15 and hybrid ML/DL, which shows good performance and reveals that the proposed algorithm could also be used for NIDS and intrusion detection in other IoT environments with slight modifications if needed. Figure 16 shows the performance metrics on the UNSW-NB15 dataset.

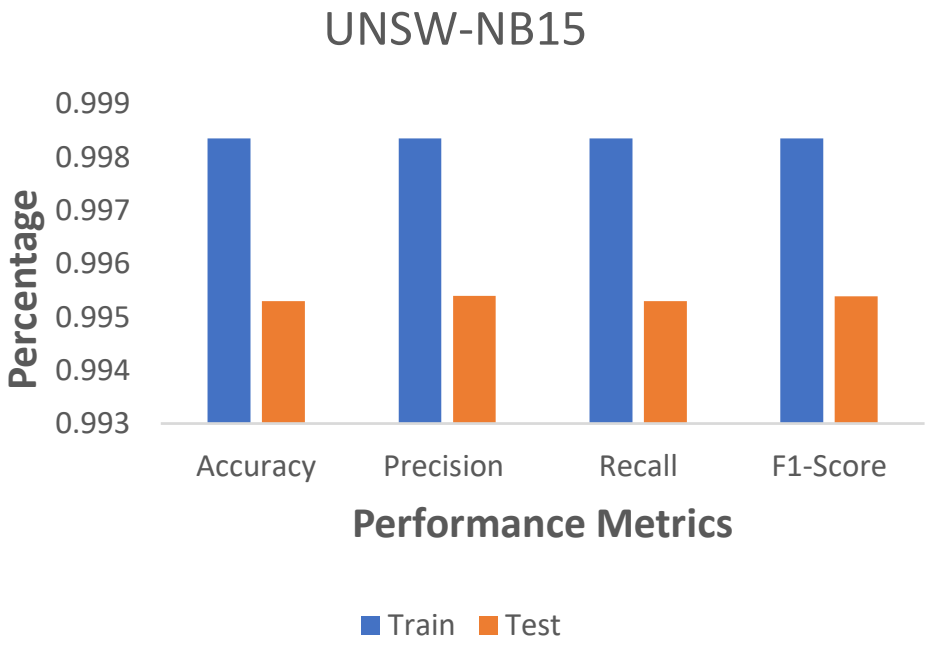

Figure 16. Performance Metrics on UNSW-NB15 Dataset.

The comparative analysis of the proposed scheme and other ML/DL schemes indicated that the proposed solution improves the performance in detecting attacks and benign classes specialized for IoT smart home networks. It also overcomes underfitting/overfitting issues and is generalizable, which makes it more promising. The proposed IDS-IoT is implemented in the Google Colab environment using Python. The performance parameters (accuracy, precision, recall, and F1 score) of the proposed mechanism were compared with other benchmark ML/DL schemes. The results show a significant improvement in the performance of the proposed solution in comparison with the other schemes.

5. Conclusions

IoT networks are ubiquitous and have changed traditional communication systems into more advanced and feasible networks. Smart homes are a field of various services, where home appliances are connected to the internet for data communication. However, these networks suffer from security attacks and vulnerabilities. Hence, an intrusion detection system is used to protect these networks by using ML/DL techniques. Therefore, ML- and DL-based techniques for attack detection in the network were investigated in the literature, which highlighted benefits, issues, and gaps. To overcome existing issues, a hybrid ML/DL-based scheme is proposed. The hybridization of KNN, DT, and LSTM was implemented on the latest CIC-IDS2022 dataset with appropriate methods for dimensionality reduction and classification. Tensor Flow in Google Colab was used to evaluate the proposed solution in terms of accuracy, precision, recall, and F1 score as compared to existing techniques. The proposed solution outperforms the others in detecting security attacks in smart home networks. In the future, the proposed solution will integrate smart grids as well.

Author Contributions: Conceptualization, N.B., A.S. and K.N.Q.; Methodology, N.B., A.S. and K.N.Q.; Formal analysis, S.H.; Writing—original draft, N.B. and A.S.; Writing—review & editing, A.O.I. and F.B.; Visualization, S.H., F.B. and N.A.; Supervision, K.N.Q. and N.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Qureshi, K.N.; Jeon, G.; Piccialli, F. Anomaly Detection and Trust Authority in Artificial Intelligence and Cloud Computing. *Comput. Netw.* **2020**, *184*, 107647. [CrossRef]
2. Elrawy, M.F.; Awad, A.I.; Hamed, H.F. Intrusion detection systems for IoT-based smart environments: A survey. *J. Cloud Comput.* **2018**, *7*, 1–20. [CrossRef]
3. Bhuvaneswari Amma, N.G.; Selvakumar, S. Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment. *Future Gener. Comput. Syst.* **2020**, *113*, 255–265.
4. Qureshi, M.A.; Qureshi, K.N.; Jeon, G.; Piccialli, F. Deep learning-based ambient assisted living for self-management of cardiovascular conditions. *Neural Comput. Appl.* **2021**, *34*, 10449–10467. [CrossRef]
5. IvanCvitic, D.; Gupta, B.B.; Choo, K.-K.R. Boosting-Based DDoS Detection in Internet-of-Things Systems. *IEEE Internet Things J.* **2021**, *9*, 2109–2123.
6. Cvitić, I.; Peraković, D.; Periša, M.; Gupta, B. Ensemble machine learning approach for classification of IoT devices in smart home. *Int. J. Mach. Learn. Cybern.* **2021**, *12*, 3179–3202.
7. Zaib, M.H.; Bashir, F.; Qureshi, K.N.; Kausar, S.; Rizwan, M.; Jeon, G. Deep learning based cyber bullying early detection using distributed denial of service flow. *Multimed. Syst.* **2021**, *28*, 1905–1924. [CrossRef]
8. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1. [CrossRef]

9. Sharmila, B.; Nagapadma, R. Intrusion detection system using Naive Bayes algorithm. In Proceedings of the 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), Bangalore, India, 15–16 November 2019; IEEE: New York, NY, USA, 2019; pp. 1–4.
10. Verma, A.; Ranga, V. ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things. In *4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 18–19 April 2019*; IEEE: New York, NY, USA, 2019; pp. 1–6.
11. Taghavinejad, S.M.; Taghavinejad, M.; Shahmiri, L.; Zavvar, M.; Zavvar, M.H. Intrusion detection in IoT-based smart grid using hybrid decision tree. In *2020 6th International Conference on Web Research (ICWR)*; IEEE: New York, NY, USA, 2020; pp. 152–156.
12. Sugi, S.S.S.; Ratna, S.R. Investigation of machine learning techniques in intrusion detection system for IoT network. In *3rd International Conference on Intelligent Sustainable Systems (ICISS), Coimbatore, India, 3–5 December 2020*; IEEE: New York, NY, USA, 2020; pp. 1164–1167.
13. Van Huong, P.; Minh, N.H. Improving the feature set in IoT intrusion detection problem based on FP-Growth Algorithm. In *2020 International Conference on Advanced Technologies for Communications (ATC)*; IEEE: New York, NY, USA, 2020; pp. 18–23.
14. Keserwani, P.K.; Govil, M.C.; Pilli, E.S.; Govil, P. A smart anomaly-based intrusion detection system for the Internet of Things (IoT) network using GWO–PSO–RF model. *J. Reliab. Intell. Environ.* **2021**, *7*, 3–21.
15. Kalnoor, G. Markov Decision Process based Model for Performance Analysis an Intrusion Detection System in IoT Networks. *J. Telecommun. Inf. Technol.* **2021**, *3*, 42–49. [CrossRef]
16. Zeeshan, M.; Riaz, Q.; Bilal, M.A.; Shahzad, M.K.; Jabeen, H.; Haider, S.A.; Rahim, A. Protocol-Based Deep Intrusion Detection for DoS and DDoS Attacks Using UNSW-NB15 and Bot-IoT Data-Sets. *IEEE Access* **2021**, *10*, 2269–2283.
17. Abdalgawad, N.; Sajun, A.; Kaddoura, Y.; Zualkernan, I.A.; Aloul, F. Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset. *IEEE Access* **2021**, *10*, 6430–6441. [CrossRef]
18. García, S.; Ramírez-Gallego, S.; Luengo, J.; Benítez, J.M.; Herrera, F. Big data preprocessing: Methods and prospects. *Big Data Anal.* **2016**, *1*, 9.
19. Seger, C. An Investigation of Categorical Variable Encoding Techniques in Machine Learning: Binary Versus One-Hot and Feature Hashing. 2018. Available online: https://www.diva-portal.org/smash/get/diva2:1259073/FULLTEXT01.pdf (accessed on 2 December 2022).
20. OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06). Open Networking Foundation. 2015. Available online: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (accessed on 2 December 2022).
21. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl. -Based Syst.* **1998**, *6*, 107–116. [CrossRef]
22. Rengasamy, D.; Jafari, M.; Rothwell, B.; Chen, X.; Figueredo, G.P. Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors* **2020**, *20*, 723. [CrossRef] [PubMed]
23. Sundermeyer, M.; Schlüter, R.; Ney, H. LSTM neural networks for language modeling. In Proceedings of the 13th Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012.
24. Elwakil, S.; El-Labany, S.; Zahran, M.; Sabry, R. Modified extended tanh-function method and its applications to nonlinear equations. *Appl. Math. Comput.* **2005**, *161*, 403–412.
25. Navada, A.; Ansari, A.N.; Patil, S.; Sonkamble, B.A. Overview of use of decision tree algorithms in machine learning. In *IEEE Control and System Graduate Research Colloquium, Shah Alam, Malaysia, 27–28 June 2011*; IEEE: New York, NY, USA, 2011; pp. 37–42.