Journal of Naval Sciences and Engineering
2022, Vol. 18, No. 2, pp. 333-361
Computer Engineering/Bilgisayar Mühendisliği

RESEARCH ARTICLE

**An ethical committee approval and/or legal/special permission has not been required within the scope of this study.*

MACHINE AND DEEP LEARNING-BASED INTRUSION DETECTION AND COMPARISON IN INTERNET OF THINGS*

Siham AMAROUCHE[1]
Kerem KÜÇÜK[2]

[1]*Kocaeli University, Department of Computer Engineering, Kocaeli, Turkey,*
siham.m.amarouche@gmail.com

[2]*Kocaeli University, Department of Software Engineering, Kocaeli, Turkey,*
kkucuk@kocaeli.edu.tr

Received: 17.02.2022 **Accepted: 23.04.2022**

ABSTRACT

In today's technology world, intrusion detection is an important topic for the Internet of Things (IoT) systems. With the growth of using tiny devices connected to wireless networks in IoT, the amount of data is growing rapidly. This data may be vulnerable to attacks so IoT systems need to secure it for increasing the system's confidentiality, availability, and reliability. The progress of detecting attacks using artificial intelligence (AI) autonomously has become a more convenient method in network intrusion detection systems (NIDS). In this article, we propose a new detecting technique to improve performance and increase accuracy in NIDS. We present different machine learning (ML) and deep learning (DL) methods to detect the different types of attacks on IoT systems. We also provide the experiments to find out the best way to identify the anomaly in the IoT system environment, make comparisons between different AI models. The experiment was evaluated with the open database UNSW-NB15.

Keywords: *Deep Learning, IoT Security, Machine Learning, Intrusion Detection, Cyber Security.*

MAKİNE VE DERİN ÖĞRENME YÖNTEMLERİ İLE NESNELERİN İNTERNETİ İÇİN SALDIRI TESPİTİNİN KARŞILAŞTIRILMASI

ÖZ

Günümüz teknoloji dünyasında, Nesnelerin İnterneti (IoT) sistemleri için izinsiz giriş tespiti önemli bir konudur. IoT'de kablosuz ağlara bağlı küçük cihazların kullanımının artmasıyla birlikte veri miktarı ihtiyacı da hızla artmaktadır. Bu veriler saldırılara karşı savunmasız olabilmektedir. Bu nedenle IoT sistem çözümlerinin gizliliğini, kullanılabilirliğini ve güvenilirliğini sağlamak için bu verilerin güvenceye alınması gereklidir. Yapay zekânın (AI) otonom biçimde kullanarak saldırıların tespit edilmesi, ağ saldırı tespit sistemlerinde (NIDS) daha uygun bir yöntem haline gelmiştir. Bu çalışmada, bu tespit sistemlerinin performansını iyileştirmek ve doğruluğu artırmak için yeni tespit tekniği önerilmektedir. IoT sistemleri için farklı saldırı türlerini tespit etmek için farklı makine öğrenimi (ML) ve derin öğrenme (DL) yöntemleri birlikte sunulmaktadır. Ayrıca, IoT sistem ortamındaki anomaliyi tanımlamanın en iyi yolunu bulmak için deneyler sunulmaktadır. Bununla birlikte farklı AI modelleri arasında karşılaştırmalar yapılmaktadır. Deneyler için, UNSW-NB15 veri seti kullanılmıştır.

Anahtar Kelimeler: *Derin Öğrenme, IoT Güvenliği, Makine Öğrenmesi, Saldırı Tespiti, Siber Güvenlik.*

1. INTRODUCTION

The Internet of Things (IoT) is a technology in which a network connects anything with the Internet, based on embedded systems, specific protocols, and sensors to conduct information exchange and communications in order to obtain smart recognitions, monitoring, localization, tracking, and control systems (Patel & Patel, 2016). The sensitivity and importance of the information carried out by IoT devices and networks signifies the importance of its security. To overhead challenges and problems on the server end we used different models as a decision engine to decide about traffic data type, whether it is normal or malicious. Cyber threats have become more widespread and several new types of attacks have been generated targeting organizations, companies, and governments. Furthermore, the number of devices and objects that are connected to wireless networks increased since the IoT has emerged. The proposed research here is found on the intersection between intrusion detection and mitigation, and Artificial Intelligence (AI) technologies. To mitigate cyber-attack, cyber security analysts heavily depend on Intrusion Detection System (IDS). IDS can detect malicious activities by matching patterns of known attacks using the signature-based detection method or observing anomaly activities using anomaly-based intrusion detection systems this method is introduced to detect unknown attacks (Khraisat, Gondal, & Vamplew, 2019).

Obviously, we can see that all governments and security intelligence try to protect their information and not allow spies to eavesdrop on it and its decisions. For the importance of cyber security topics, we research in this work the effectiveness of using ML and DL models in cyber security as well as current challenges that face security analysts and we aim to use different methodologies to prevent, mitigate attacks and drop the malicious packets and threats.

In the literature, there are many studies utilizing various machine learning and deep learning techniques on different datasets to enhance the IDS performance. In this section, we will handle anomaly-based IDS techniques which are dependent on heuristics, statistics, or rules, rather than signatures or patterns. As in the study (Alsamiri & Alsubhi, 2019), new features were extracted from the Bot-IoT and compared with existing studies from the literature. Alsamiri et al. extracted features using CICFlowMeter. In the evaluation phase, seven different machine learning (KNN, QDA, ID3, RF, AdaBoost, MLP, and NB) were used. They observed that Adaboost was the best performing algorithm, followed by KNN and ID3. Also, in (Kasongo & Sun, 2020), the authors tried to analyze the performance of intrusion detection system using a feature selection method on the UNSW-NB15 dataset, then implemented the following machine learning approaches using the reduced feature space: Support Vector Machine, k-Nearest-Neighbour, Logistic Regression, Artificial Neural Network, and Decision Tree. The feature selection method that is applied was a filter-based feature reduction technique using the XGBoost algorithm. The results showed that the XGBoost-based feature selection method allows models such as the Decision tree model to enhance their test accuracy rate from 88.13% to 90.85% for the binary classification scheme. Vinayakumar, Soman, and Poornachandran (2017) inferred from their work that experiments of families of RNN architecture achieved a low false positive rate in comparison to the traditional machine learning classifiers. The reason for that is that RNN architecture is able to memorize information over time. This work applied to publically available ID datasets, KDDCup '99' and UNSW-NB15.

Our article is comprehensive with different experiences and concepts. We considered two schemes binary and multiclass classification configurations. We made a comparison between different Machine Learning (ML) and Deep Learning (DL) methods. We evaluated the applied models with different performance parameters. We applied Random Forest algorithm over the selected dataset to calculate the feature importance measure for each feature, select more important features and generate reduced optimal feature vectors, this process may increase the accuracy of intrusion detection and increase the speed of models to get performance results. We selected this algorithm because it belongs to the category of embedded methods where these methods combine the qualities of filter and wrapper methods, are more accurate, and generalize better. We applied the feature selection method only for machine learning models because in neural networks the important features are chosen automatically. We tried to make a comparison between performance results for ML models with a full feature space and

ML models with a reduced optimal vector that was generated using the feature selection method. Also for deep learning methods to exceed the overfitting problem, we used different techniques like cross-validation, and early stopping techniques. In some DL models, we designed different architectures where one architecture is more complicated than the other. In addition, we implemented different hyperparameters such as epoch numbers and batch size values over the best-performing multiclass classification based deep learning model, to find the best hyperparameters that could be applied to improve our model's performance. We noticed the effect of parameters on the model's accuracy.

2. MATERIALS AND METHODS

In the literature, there are a lot of relevant intrusion detection datasets. An example, KDD99, NSL-KDD, DS2OS, UNSW-NB15, CIC-IDS 2017, MQTT-IOT-IDS2020, and Bot-IoT Datasets. However, in this section, we describe the dataset we used in our work.

2.1. UNSW-NB15 Dataset

In our study, we used the UNSW-NB15 dataset which is created by the Cyber Range Lab of the Australian Centre for Cyber Security (Moustafa & Slay, 2015). We selected this dataset because it is a public dataset not private, diversity of attack types included in this dataset, the difficulty of evaluating and analyzing the UNSWNB15 on existing classification systems demonstrated that this data set contains complex patterns, the training and testing sets have a similar probability distribution, and regular updates that can be applied to this dataset. The Bro-IDS, Argus tools are employed and twelve algorithms are developed to extract 49 features with the class label (Moustafa & Slay, 2015). The number of instances in the training set is 175,341 (68.05%) records and the testing set is 82,332 (31.95%) records from the different types of attack and normal. The UNSW-NB15 dataset includes nine types of attack classifications to describe malicious behaviors. Attack types included in the UNSW-NB15 dataset are Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

2.2. Technologies Overview

In our work, we used Python programming language, Python is interpreted, high-level, and object-oriented. Today, machine learning has become more popular and attracted the attention of students, scientists, and researchers. Python with its useful packages and libraries enables to make complex computational tasks easily. Python has a lot of libraries such as Keras, Scikit-learn, Tensorflow, pandas, Numpy, and Matplotlib, etc. We used these libraries in our study to help us implementing feature engineering process over the dataset, classical machine learning, and deep learning methods. The proposed models in our work are implemented in Jupyter Notebook by using the Python language and its libraries.

2.3. Pre-processing Data

2.3.1. Feature Transformation and Standardization

Before feeding the data to classical machine learning and deep learning models, feature transformation has been applied and this step is important because models accept only numerical data as input for that, all non-numerical values of the dataset are converted into numerical data. In this work, we used one hot encoding (ohe) technique to encode categorical features as a one-hot numeric array. The encoder derives the unique values for each feature and represents it as a one-hot array. On the other side, for the standardization process, we used the StandardScaler technique over the numerical data because the values of the dataset are in different ranges and we tried to standardize data in the same range. Standardization is a scaling technique where input values are centered on the mean with a unit standard deviation.

2.3.2. Feature Selection and Dimensionality Reduction

In this step, we select the most important features and delete unnecessary features or reduce the dimensions of features in the dataset. For the task of feature extraction and dimension reduction processes, we can use different models like Principal component analysis (PCA), Linear discriminant analysis (LDA), Autoencoder, and t-SNE models. On the other hand, feature selection is used to reduce irrelevant and redundant variables and it measures the relevance of each feature with the output labels/classes based on feature importance metric. Feature selection technologies are divided into embedded, wrapper, and filter methods (Chandrashekar & Sahin, 2014). Our goal in applying the feature selection method and reducing the number of input variables is to improve the performance of the model in some cases and reduce the computing cost and time of the training model.

In this study, we applied only feature selection methods using Random Forest that depends on tree-based strategies and belongs to the category of embedded methods. Embedded methods combine the qualities of filter and wrapper methods, are more accurate, and generalize better. Features importance and its scores that calculated using Random Forest method as shown in Figure 1:

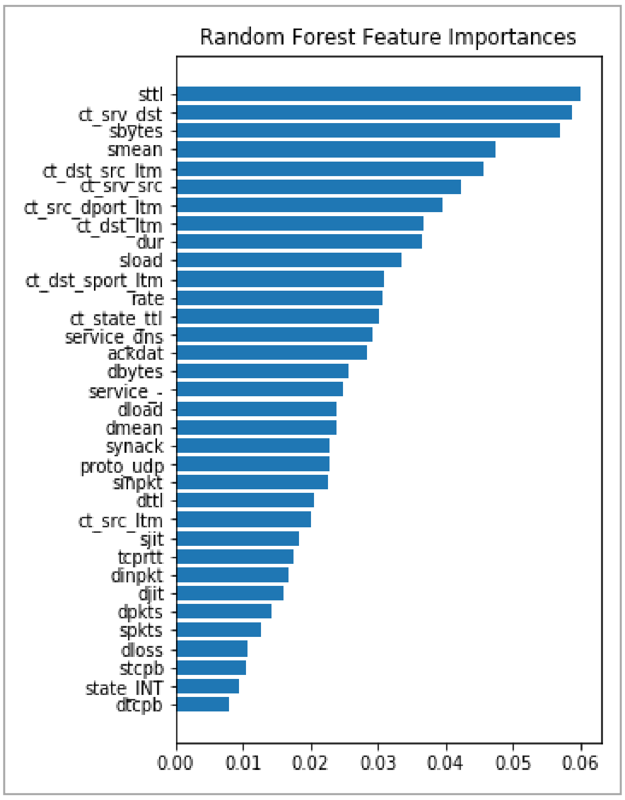


Figure 1. Feature selection using random forest.

3. METHODS FOR INTRUSION DETECTION SYSTEM

3.1. Classical Machine Learning Methods

3.1.1. Naïve Bayes

Naive Bayes (NB) is a subset of Bayesian decision theory and it is a simple probabilistic machine learning model based on the Bayes theorem where assumptions between features are considered independent. This method is used for classification tasks. There are different types of Naive Bayes as Multi-nominal, Bernoulli, and Gaussian Naive Bayes algorithms (Kaviani & Dhotre, 2017). In our work, we used Bernoulli Naive Bayes model. Bayes theorem mathematically can be described as follows:

$$\mathrm{P}\ (\mathrm{A}|\mathrm{B}) = \frac{\mathrm{P}(\mathrm{B}|\mathrm{A})\mathrm{P}(\mathrm{A})}{\mathrm{P}(\mathrm{B})} \tag{1}$$

3.1.2. K-nearest Neighbors

K-nearest neighbors (KNN) is an ML algorithm that is capable of both supervised and unsupervised approaches and it is used for both classification and regression problems. In this research, we used it for supervised binary and multiclass classification tasks. KNN algorithm assumes that similar instances exist in the same area and proximity. Different distance metrics are used in this model. Distance metrics find the distance between two instances between a new data point and an existing point in the training dataset (Chomboon, Chujai, Teerarassamee, Kerdprasop, & Kerdprasop, 2015). One of the commonly used distance metrics is Euclidean distance, the formula of it as follows:

$$\mathrm{d}\ (\mathrm{x},\mathrm{y}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x}_\mathrm{i} - \mathrm{y}_\mathrm{i})^2} \tag{2}$$

This formula is based on Pythagorean Theorem; and it can be used to calculate the distance between two data points x and y in Euclidean space.

3.1.3. Logistic Regression

Logistic regression (LR) is a statistical method used for binary classification tasks. Although its name regression, it is a classification algorithm. Logistic regression tries to make a logarithmic line that distinguishes between classes and its estimation is done through maximum likelihood. LR model depends on the Sigmoid function where its logistic curve is limited between 0 and 1 values (Boateng & Abaye, 2019). The expression of Sigmoid function is as follows:

$$\sigma(x) = \frac{1}{1 + e^{(-x)}} \tag{3}$$

3.1.4. Decision Tree

Decision Tree (DT) is a decision support tool that uses a tree-like method. DT is a supervised model consisting of internal nodes that represent attributes, branches that represent the outcome of the tests, and leaf nodes that represent classes/labels and decisions after the computing process. The paths between root and leaf represent decision rules for classification tasks (Safavian & Landgrebe, 1991).

3.1.5. Random Forest Classification

Random Forest (RF) is an ensemble method for classification and regression. RF model is a combination of different decision trees. The ensemble method is a machine learning technique that combines several base models or decision trees to produce one optimal model and predict with better performance than utilizing a single model (Breiman, 2001).

3.2. Deep Learning Methods

Deep learning (DL) is a subfield of machine learning inspired by the structure of the human brain and biological neural networks. DL is known with its high performance and efficiency across many types of data.

3.2.1. Deep Neural Network

Deep neural network (DNN) is an artificial neural network (ANN) model with high complexity, usually with at least two hidden layers. This model has become a popular model for classification, regression, clustering, controlling models, and prediction in many applications (Abiodun et al., 2018). Deep net process data by employing sophisticated math methods (Sze, Chen, Yang, & Emer, 2017). The feed-forward neural network was the first of neural networks (NN) that found and simplest type. In this network, the data move forwardly from the input layer through any hidden layer to the output without loops. The model can give different performance results depends on the number of hidden layers, the number of nodes in each layer, and the type of activation layer (Abiodun et al., 2018).

This model is applied in a supervised manner with the class labels and the input attributes. In this model, we have forwarding propagation which aims to predict results as an attack or normal by using a perceptron classifier. The main Equation of the perceptron in the artificial neural network is mentioned in Equation (4):

$$y = \sum_{i=1}^{n} X_i W_i + b \tag{4}$$

Where n denotes the number of nodes in the layer, X denotes the values of these nodes which are the samples values, W refers to weights (connection strength), and b to the biases of these nodes. These results will be inserted into different activation functions which return the probabilities for each class and then choose the largest value from the vector of probability values to give a more accurate value. Sigmoid, ReLU, Softmax, and Tanh are among the most frequently used activation functions; this model employed ReLU activation functions in the hidden layers stated in Equation (5).

$$R(z) = \begin{cases} z, & z > 0 \\ 0, & z \leq 0 \end{cases} \tag{5}$$

Also, we used Softmax activation function in the output layer stated in Equation (6).

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{6}$$

Here Softmax function transforms a vector of numbers into a vector of probabilities between 0 and 1. After the forwarding propagation stage, the backpropagation step comes and is a technique to train deep neural networks by modifying the weights and biases. It includes loss function and optimizer (Manaswi, 2018). In this step, the loss between predicted and true values will be calculated, then adjust the weights in the neural network according to the loss. Categorical cross entropy loss function has been applied in this work. The loss function needs to reach the optimal values of parameters (weight and bias) for that we used Adam optimizer to get the best parameter values. The optimizer is a way of tuning parameters (Manaswi, 2018). In our work for DNN models, we used two different architectures: DNN-2 and DNN-1. DNN-2 structure is more complicated than DNN-1 where we added one more hidden layer.

3.2.2. Convolutional Neural Network

Convolutional neural network (CNN) is one of the most powerful models in deep learning. CNN has excellent performance with different applications like image classification, video recognition, action recognition, and natural language processing (NLP). It handles input data as matrices for that we reshaped our input data before feeding it to be more convenient for the CNN module. CNN models have multiple layers, including convolutional layer, pooling layer, non-linearity layer, and fully connected layer (Albawi, Mohammed, & Al-Zawi, 2017).

In our work, we used 1D-CNN architecture for intrusion detection tasks while 2D-CNN architecture is mainly used for image processing tasks. Similarly, to DNN model, ReLU activation function was used for hidden layers, Softmax activation function was applied in the output layer, categorical cross entropy was used as a loss function, and Adam was used as an optimizer. We designed two different CNN models, where the structure of CNN-2 is more complicated than CNN-1 and we added one more hidden layer in the CNN-2 model.

3.2.3. Recurrent Neural Network

A recurrent neural network is a type of artificial neural network in which nodes' connections form a directed graph through a temporal sequence.

RNNs can process variable-length sequences of inputs by utilizing their internal state (memory) (Medsker & Jain, 2001). RNN model is used to memorize and remember previous computations. This model allows the use of previous outputs as inputs while maintaining hidden states (Sherstinsky, 2020) where for each timestep t, the activation function and the output are stated as follows;

$$a^{<t>} = g_1\left(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a\right) \quad (7.\mathrm{a})$$
$$\text{and} \quad y^{<t>} = g_2\left(W_{ya}a^{<t>} + b_y\right) \quad (7.\mathrm{b}) \qquad \textbf{(7)}$$

Where W_{ax}, W_{aa}, W_{ya}, b_a, b_y are temporally shared coefficients and g_1, g_2 activation functions.

RNN model in some cases can face the long-term dependency problem, the vanishing gradient, and exploding gradient problems. In order to solve these problems, LSTM and gated recurrent unit (GRU) networks have been proposed.

3.2.4. Long Short-Term Memory

Long short-term memory is an artificial recurrent neural network architecture. In contrast to standard feed forward neural networks, LSTMs include feedback connections and process the data as sequences (Hochreiter & Schmidhuber, 1997). LSTM can be used to perform tasks such as connected handwriting recognition, speech recognition, and anomaly detection in network systems or intrusion detection systems (IDS) (Graves, 2012). The main difference from a simple RNN is that memory blocks are used in place of nonlinear units in the hidden layers and this model offers three units called gates which are input gate, forget gate, and output gate. The following Equations (8) represent the gates in LSTM (Van Houdt, Mosquera, & Nápoles, 2020);

$$i_t = \sigma\left(w_i[h_{t-1}, x_t] + b_i\right) \quad (8.\mathrm{a})$$
$$f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \quad (8.\mathrm{b}) \qquad \textbf{(8)}$$
$$o_t = \sigma\left(w_o[h_{t-1}, x_t] + b_o\right) \quad (8.\mathrm{c})$$

Where “i” for the input gate, “f” for the forget gate, “o” for the output gate, “σ” is the Sigmoid function, “b_i” is the biases for the gate(x), “h_{t-1}” is the output of the previous LSTM block, and “x_t” is the current input.

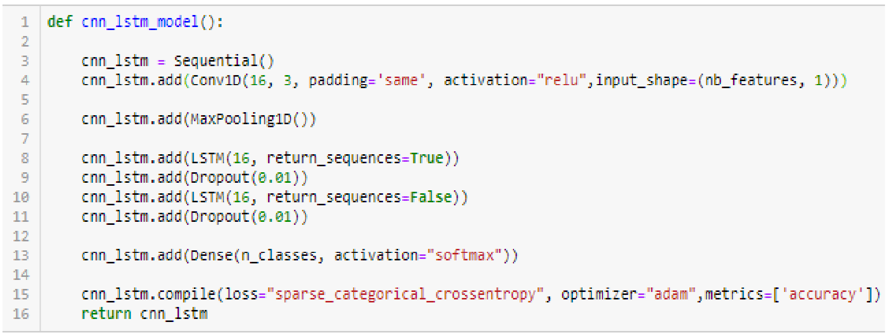
3.2.5. Gated Recurrent Unit

The GRU is similar to a long short-term memory (LSTM) with a forget gate, but requires fewer parameters due to the absence of an output gate (Hochreiter & Schmidhuber, 1997). GRU is a simplified version of LSTM and it merges the forget and the input gates into a single "update gate", as well as merges cell and hidden state (Jozefowicz, Zaremba, & Sutskever, 2015).

In RNN, LSTM, and GRU models, we employed Tanh activation function in the hidden layers; also we used Softmax activation function in the output layer. Sparse categorical cross entropy loss function and Adam optimizer have been applied for these models.

3.2.6. CNN-LSTM Model

CNN-LSTM model is a hybrid model that combines convolutional neural networks (CNN) and LSTM networks. This architecture involves using CNN layers for feature extraction from input dataset, followed by an LSTM model to detect intrusions sequentially. In our work, we constructed our own CNN-LSTM model using Python's Keras library, the activation function that is used is ReLU for CNN model, Tanh for LSTM model and Softmax for the output layer, loss function is sparse categorical cross entropy and the optimizer is Adam; shown as below in Figure 2:



```python
def cnn_lstm_model():

    cnn_lstm = Sequential()
    cnn_lstm.add(Conv1D(16, 3, padding='same', activation="relu",input_shape=(nb_features, 1)))

    cnn_lstm.add(MaxPooling1D())

    cnn_lstm.add(LSTM(16, return_sequences=True))
    cnn_lstm.add(Dropout(0.01))
    cnn_lstm.add(LSTM(16, return_sequences=False))
    cnn_lstm.add(Dropout(0.01))

    cnn_lstm.add(Dense(n_classes, activation="softmax"))

    cnn_lstm.compile(loss="sparse_categorical_crossentropy", optimizer="adam",metrics=['accuracy'])
    return cnn_lstm
```

Figure 2. Keras library usage for CNN-LSTM model.

3.3. Evaluation Metrics

The applied models are evaluated by defining five performance parameters: accuracy, false alarm rate, precision, recall, and f1 score.

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}} \quad \textbf{(9)}$$

$$\text{FAR} = \frac{\text{FP} + \text{FN}}{\text{FP} + \text{FN} + \text{TP} + \text{TN}} \quad \textbf{(10)}$$

$$\text{Precion} \ = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad \textbf{(11)}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad \textbf{(12)}$$

$$\text{F1}_{\text{score}} \quad = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad \textbf{(13)}$$

In addition to the accuracy, we used different evaluation metrics such as precision, recall, and F1 since our dataset is imbalanced. In this case, the accuracy may cause to mislead evaluation of performance in some situations. As we can see from Equation (13) F1 score is the harmonic mean value of recall and precision.

3.4. Flow Diagram and Architecture

Flow diagram of this study shown as in flow chart Figure 3, we applied different Artificial Intelligence (AI) models: Deep Learning (DL) and Machine Learning (ML) models. Besides that, we handled our data with two different perspectives: binary and multiclass classification. We implemented Random Forest feature selection method only for machine learning models. In deep learning models we excluded the need to the feature selection method because these models are inherently black boxes and generate the non-linear combinations where the features have less effect get lesser weights automatically.

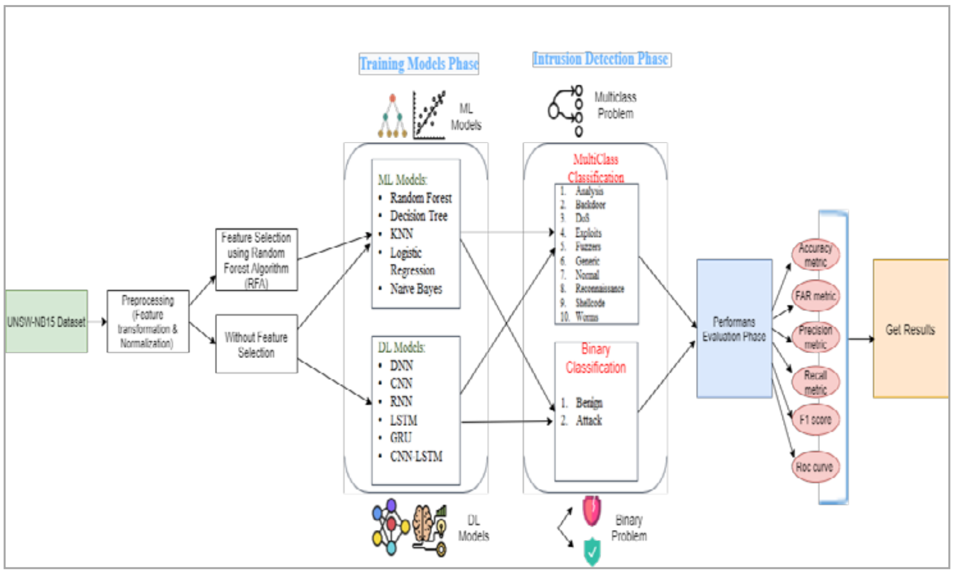


Figure 3. Architecture and flow diagram of proposed work.

4. ANALYSIS AND RESULTS

As we can see in Table 1 and 2, we employed all feature space of UNSW-NB15 dataset for multiclass classification and binary configuration. In Table 2 and 4 we used reduced feature vectors for multiclass classification and binary setting. In the experimental process, we applied different AI models that contain ML models and DL models. Here in the analysis section, we have 4 stages. In the first step, we tried to compare performance results for ML (binary and multiclass approaches) with full feature space that contains 42 attributes and with a reduced optimal vector (34 features) that generated using the Random Forest method. In the second stage, we tried to compare multiclass classification with binary configurations. In the third phase, we compare performance results between different DL models. In the fourth phase, we tried to compare DL performance results with ML results. In each table, Test Accuracy is the accuracy that obtained on testing data.

From Table 1 and 2, the performance results demonstrated that the feature selection methodology allows improving accuracies for such models as Random Forest accuracy increased from 75.38% to 75.90% and Decision

Tree accuracy increased from 73.43% to 73.86%. These results represented for multiclass classification scheme.

For Table 3 machine learning (ML) models for binary classification were applied without feature selection by using all our 42 features and for Table 4 our ML models were applied with a random forest feature selection method and we selected the most important 34 features. As we can see from Table 3 and 4 there is no performance improvement for binary classification tasks with the feature selection technique.

If we tried to compare multiclass classification results with binary classification results either with using the feature selection technique or without it, obviously we can see that accuracies improved for models in binary classification approach. As we can see from Table 1 and 3 with all 42 features, Random Forest accuracy increased from 75.38% to 87.09%, Decision Tree accuracy increased from 73.43% to 86.36%, KNN from 70.93% to 84.48%, Logistic Regression from 68.51% to 80.93%, and Naive Bayes from 53.45% to 74.78%. Also, from Table 2 and 4 with applying the feature selection method, Random Forest accuracy increased from 75.90% to 86.97%, Decision Tree accuracy increased from 73.86% to 86.18%, KNN from 70.59% to 82.10%, Logistic Regression from 67.92% to 80.33%, and Naive Bayes from 54.44% to 73.06%. This is a normal case because when classes number decreased, possible probabilities decrease, and the success rate will be increases and improves.

Table 1. Results using ML models with 42 features – multiclass classification.

ML Model	**Cross Validation Results**				**Evaluation Results on Testing Data**				
	CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
Random Forest	82.43	82.30	82.43	81.54	75.38	24.62	83.80	75.38	77.51
Decision Tree	80.94	80.98	80.94	80.55	73.43	26.57	80.75	73.43	76.26
KNN	76.52	76.86	76.52	76.51	70.93	29.07	79.04	70.93	73.85
Logistic Regression	77.07	76.88	77.07	75.25	68.51	31.49	76.94	68.51	69.96
Naive Bayes	63.90	72.83	63.90	64.84	53.45	46.55	74.91	53.45	58.35

Table 2. Results using ML models with 34 features (features selection) – multiclass classification.

ML Model	Cross Validation Results				Evaluation Results on Testing Data				
	CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
Random Forest	82.90	82.75	82.90	81.93	75.90	24.10	83.54	75.90	77.87
Decision Tree	81.08	80.93	81.08	80.53	73.86	26.14	80.52	73.86	76.35
KNN	76.31	76.80	76.31	76.37	70.59	29.41	78.72	70.59	73.53
Logistic Regression	76.34	75.88	76.34	74.39	67.92	32.08	76.04	67.92	69.52
Naive Bayes	65.13	73.69	65.13	65.61	54.44	45.56	72.95	54.44	58.22

Table 3. Results using ML models with 42 features – binary classification.

ML Model	Cross Validation Results				Evaluation Results on Testing Data				
	CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
Random Forest	95.95	95.93	95.95	95.93	87.09	12.91	88.84	87.09	86.77
Decision Tree	94.87	94.88	94.87	94.87	86.36	13.64	87.29	86.36	86.13
KNN	93.79	93.76	93.79	93.76	84.48	15.52	86.17	84.48	84.07
Logistic Regression	93.52	93.75	93.52	93.37	80.93	19.07	84.03	80.93	80.07
Naive Bayes	74.63	82.26	74.63	75.47	74.78	25.22	76.68	74.78	74.74

Table 4. Results using ML models with 34 features (features selection) – binary classification.

ML Model	**Cross Validation Results**				**Evaluation Results on Testing Data**				
	CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
Random Forest	95.87	95.86	95.87	95.85	86.97	13.03	88.82	86.97	86.64
Decision Tree	94.79	94.79	94.79	94.79	86.18	13.82	87.20	86.18	85.94
KNN	93.74	93.71	93.74	93.70	82.10	17.90	83.13	82.10	81.72
Logistic Regression	93.37	93.74	93.37	93.18	80.33	19.67	83.71	80.33	79.38
Naive Bayes	77.65	82.16	77.65	78.38	73.06	26.94	73.32	73.06	73.12

Table 5 and 6 obtained DL performance results by using full feature space with 42 features for multiclass classification and binary classification, respectively. In deep learning methods, we eliminate the need for features selection since these methods act as black boxes and generate non-linear combinations while the features that have less effect get lesser weight automatically. As introduced in Table 5, the best performing models for multiclass classification tasks are the more complicated models and more convenient models for the used dataset, such as DNN-2 with 77.36% accuracy, RNN with 76.69% accuracy, and CNN-LSTM with 76.50% accuracy. Besides that, in Table 6 for binary classification tasks, CNN-LSTM with 87.34% accuracy, LSTM with 86.64% accuracy, and RNN with 86.24% achieved better results for intrusion detection in our dataset. Also, as we explained, we know that DNN-2 is more complicated than DNN-1 and the structure of CNN-2 is more convoluted than CNN-1. In Table 5, DNN-2 (77.36%) model achieved better accuracy than DNN-1 (74.61%). The same for CNN models in Table 6 the results demonstrated that the accuracy increased from 86.98% for CNN-1 to 85.80% for CNN-2.

For the multiclass classification problem, Table 5 shows that DNN-1 has 74.61% accuracy and GRU has 72.96% accuracy these models get lower performance results. While for the binary classification task, CNN-2 with 84.91% accuracy and CNN-1 with 84.50% accuracy perform lower accuracies than other DL models and we think this happened with convolutional neural networks because our dataset in the binary classification problem has smaller dimensions, which leads to overfitting problem. In Table 5, we can observe that the best performing models run a lower number of epochs with early stopping based on 5-fold cross-validation and this method prevents overfitting problems in the dataset.

To comparison between ML and DL, we can say that DL models achieved higher accuracies than ML models. From Table 1 and 5, we observed that while accuracies in DL models for multiclass classification between 77.36% and 72.96%; the accuracy results in ML models decreased until 53.45%, and in general accuracies values are between 75.38% and 53.45%.

From Table 3 and 6, we can see that while accuracies in DL models for binary classification between 87.34% and 84.50%; the accuracy results in ML models decreased until 74.78%, and in general accuracies values are between 87.09% and 74.78%. We observed that random forest and decision tree perform well this is because it is ensemble methods. Ensemble methods are techniques that use multiple classifiers, and then combine them to provide enhanced performance. The predictions of ensemble methods are collected to determine the most often occurring outcome and usually produce more accurate results than a single model. Also, we observed that families of RNN architecture achieved a high accuracy rate in comparison to the traditional machine learning classifiers and the reason for that is the RNN architectures are capable of retraining information over time and maintaining connection sequence data. As a result, from the experiments, we notice that are more complex models get higher accuracies in some cases. We applied 5-fold cross-validation for both the DL and ML models to get more accurate results about the performance and compared intrusion detection accuracies between different models.

Table 5. Results using DL models with 42 features – multiclass classification.

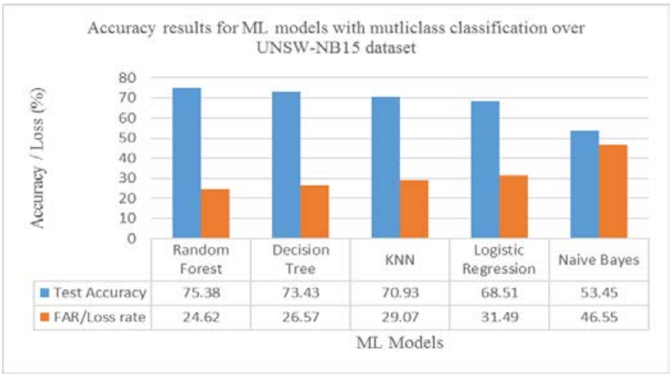
DL Model	**Cross Validation Results**					**Evaluation Results on Testing Data**				
	Epochs for early stopping based on 5-fold CV	***CV Accuracy mean***	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
DNN-2	23, 2, 2, 7, 1	81.55	81.53	81.55	79.36	77.36	22.64	81.26	77.36	77.07
RNN	24, 22, 1, 5, 1	80.89	80.84	80.89	78.11	76.69	23.31	79.64	76.69	76.22
CNN-LSTM	31, 6, 1, 7, 9	81.41	81.26	81.41	78.66	76.50	23.50	81.25	76.50	76.76
CNN-1	20, 9, 1, 11, 3	79.84	80.36	79.84	76.78	75.93	24.07	79.21	75.93	75.44
LSTM	44, 5, 1, 3, 1	81.76	81.49	81.76	79.64	75.83	24.17	79.94	75.83	76.19
CNN-2	24, 2, 3, 3, 5	80.54	79.94	80.54	77.52	74.80	25.20	82.29	74.80	74.92
DNN-1	24, 5, 1, 2, 1	81.10	80.92	81.10	79.37	74.61	25.39	81.41	74.61	75.08
GRU	45, 2, 1, 2, 3	81.65	81.66	81.65	79.61	72.96	27.04	80.57	72.96	74.51

Table 6. Results using DL models with 42 features – binary classification.

DL Model	**Cross Validation Results**					**Evaluation Results on Testing Data**				
	Epochs for early stopping based on 5-fold CV	***CV Accuracy mean***	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
CNN-LSTM	41, 3, 6, 2, 4	94.96	94.95	94.96	94.93	87.34	12.66	89.01	87.34	87.03
LSTM	18, 12, 2, 6, 4	94.89	94.87	94.89	94.85	86.64	13.36	88.14	86.64	86.33
RNN	28, 5, 5, 12, 2	94.82	94.81	94.82	94.78	86.24	13.76	88.04	86.24	85.88
DNN-1	21, 1, 2, 3, 7	94.71	94.71	94.71	94.66	86.13	13.87	88.13	86.13	85.74
GRU	24, 21, 1, 4, 2	94.87	94.85	94.87	94.84	86.05	13.95	87.90	86.05	85.67

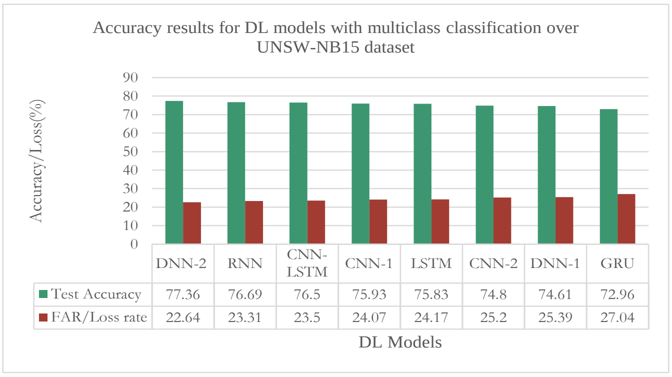
Table 6. (Cont.) Results using DL models with 42 features – binary classification.

DL Model	Cross Validation Results					Evaluation Results on Testing Data				
	Epochs for early stopping based on 5-fold CV	***CV Accuracy mean***	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
DNN-2	25, 1, 2, 1, 3	94.92	94.91	94.92	94.88	86.03	13.97	87.89	86.03	85.66
CNN-2	17, 5, 1, 5, 1	94.70	94.73	94.70	94.63	84.91	15.09	87.73	84.91	84.35
CNN-1	23, 1, 6, 2, 1	94.49	94.56	94.49	94.41	84.50	15.50	87.42	84.50	83.91



	Random Forest	Decision Tree	KNN	Logistic Regression	Naive Bayes
Test Accuracy	75.38	73.43	70.93	68.51	53.45
FAR/Loss rate	24.62	26.57	29.07	31.49	46.55

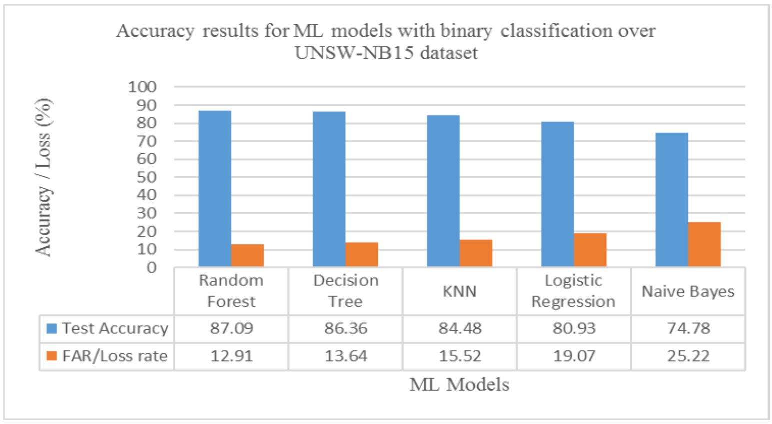
Figure 4. ML models performance over multiclass classification with 42 features.



	DNN-2	RNN	CNN-LSTM	CNN-1	LSTM	CNN-2	DNN-1	GRU
Test Accuracy	77.36	76.69	76.5	75.93	75.83	74.8	74.61	72.96
FAR/Loss rate	22.64	23.31	23.5	24.07	24.17	25.2	25.39	27.04

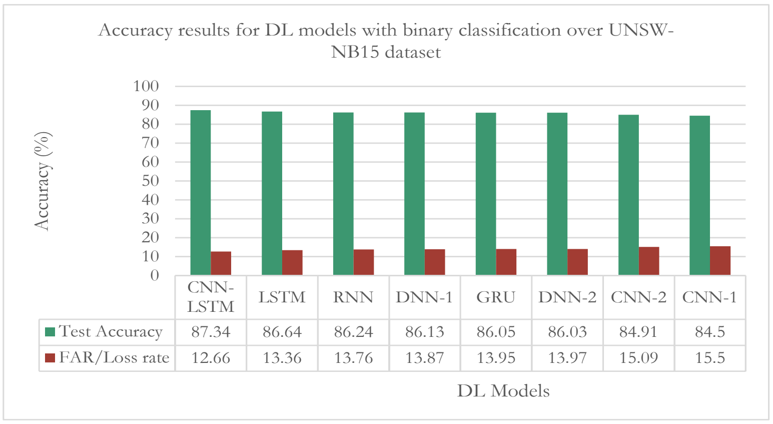
Figure 5. DL models performance over multiclass classification.

As we can observe from above Figure 4 and 5, the traditional machine learning model’s performance decreased to 53.45% against the case in the deep learning model the accuracy results decreased to 72.96%. We notice that decision tree and random forest ensemble methods produce improved results similar to deep learning methods and it is a normal case because these techniques aggregate multiple models and predictors to provide results that are more accurate. In addition, we inferred from the experiments that families of RNN architecture achieved a high accuracy rate in comparison to the traditional machine learning classifiers. The reason for that is that RNN architectures are able to memorize information over time and have connection sequences of information that store previous blocks values. On the other hand, for binary classification, as illustrated in Figure 6 and 7, the performance of classical machine learning models decreased to 74.78% while the accuracy of deep learning models decreased to 84.50%, which is a higher value than the machine learning model’s value.



	Random Forest	Decision Tree	KNN	Logistic Regression	Naive Bayes
🟦 Test Accuracy	87.09	86.36	84.48	80.93	74.78
🟧 FAR/Loss rate	12.91	13.64	15.52	19.07	25.22

Figure 6. ML models performance over binary classification with 42 features



	CNN-LSTM	LSTM	RNN	DNN-1	GRU	DNN-2	CNN-2	CNN-1
🟩 Test Accuracy	87.34	86.64	86.24	86.13	86.05	86.03	84.91	84.5
🟥 FAR/Loss rate	12.66	13.36	13.76	13.87	13.95	13.97	15.09	15.5

Figure 7. DL models performance over binary classification

The receiver operating characteristic (ROC) is a graphical representation of a binary classifier system's diagnostic ability as its discrimination threshold is varied. This curve plots two parameters: False Positive Rate (FPR) and True Positive Rate (TPR). As shown in Figure 8, we can see the variance in performances between ML and DL models for binary classification problems, the ROC curve of DL models is better than the ROC curve of ML models.

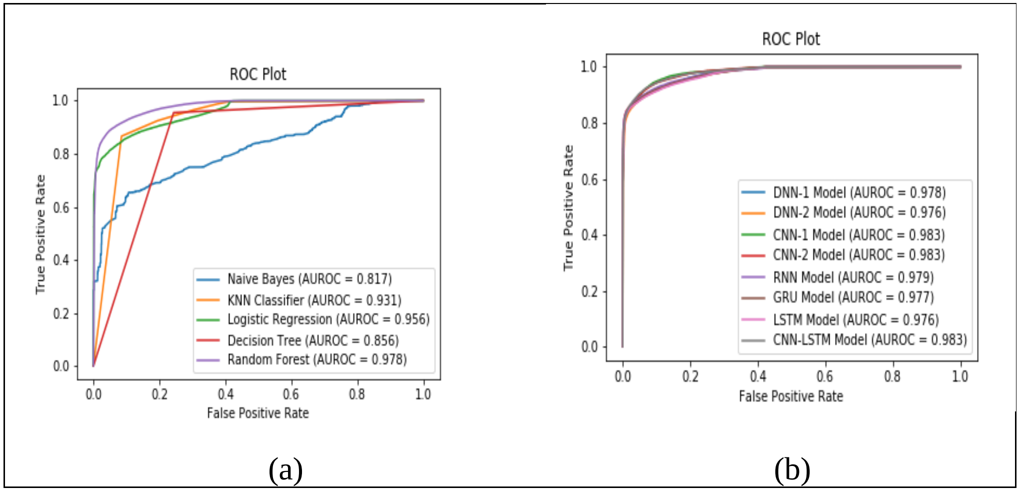


Figure 8. Comparison between ROC curves for classical ML models and DL models: (a) ROC curve for ML models in binary classification, (b) ROC curve for DL models in the binary classification problem.

Implementation of deep learning models depends on different hyperparameters, which are a critical component of any deep network since they enable us to optimize the network's quality.

Table 7. Intrusion detection results using DNN-2 DL model over multiclass classification with different number of epochs.

DNN-2 Model	**Epochs**	**Cross Validation Results**				**Evaluation Results on Testing Data**				
		CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
* Activation function in hidden layers: ReLU & in output layer: Softmax * Optimizer: adam * Batch size: 32	40	82.09	82.16	82.09	80.09	<u>77.17</u>	22.83	81.57	77.17	<u>78.14</u>
	50	82.06	81.68	82.06	80.32	75.58	24.42	80.29	75.58	76.59
	100	82.25	81.90	82.25	80.47	75.39	24.61	81.81	75.39	76.45
	25	81.95	81.99	81.95	79.98	75.15	24.85	80.82	75.15	76.29
	10	81.86	81.50	81.86	79.98	75.02	24.98	80.97	75.02	76.25

Table 8. Intrusion detection results using DNN-2 DL model over multiclass classification with different batch size.

DNN-2 Model	**Batch Size**	**Cross Validation Results**				**Evaluation Results on Testing Data**				
		CV Accuracy mean	***CV Precision mean***	***CV Recall mean***	***CV F1 mean***	***Test Accuracy***	***Test FAR rate***	***Test Precision***	***Test Recall***	***Test F1***
* Activation function in hidden layers: ReLU & in output layer: Softmax * Optimizer: adam * Epochs: 40	16	81.78	81.63	81.78	79.84	<u>76.80</u>	23.20	81.42	76.80	<u>77.36</u>
	32	82.05	81.67	82.05	80.06	76.49	23.51	80.77	76.49	77
	64	81.86	81.59	81.86	80.02	75.82	24.18	80.97	75.82	76.92
	48	81.83	81.58	81.83	79.89	75.74	24.26	80.62	75.74	76.31
	128	81.96	81.73	81.96	79.83	75.62	24.38	81.15	75.62	76.73

From Table 7, we can observe that epochs number 40 will be a better choice for the DNN-2 model with 77.17% test accuracy and 78.14% test F1. Because of that, we chose it and then implemented it with different batch size values (16, 32, 48, 64, 128). As shown in Table 8, it indicates that the batch size value of 16 achieves better results with 76.80% test accuracy and 77.36% test F1. Besides comparison between accuracy and different epoch numbers, and batch size values over deep learning methods, we could also use different activation functions in hidden layers, optimization algorithms, learning rate, batch number, number of neurons, and number of hidden layers.

5. CONCLUSION AND RECOMMENDATIONS

In this research, we applied the following supervised Machine Learning (ML) methods for IDS: Naïve Bayes (NB), k-Nearest-Neighbor (kNN), Logistic Regression (LR), Decision Tree (DT) and Random Forest (RF). Also, Deep Learning (DL) methodologies such as: Deep Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Long Short-Term Memory (LSTM), and CNN-LSTM model. Traditional machine learning methods depend heavily on feature engineering, which is often time-consuming and complex and with the complexity of the IoT structure, it is critical to develop an IDS that achieves low computational costs and reduces the amount of energy consumed. As a result, it is impractical to detect anomalies in real-time using classical machine learning models. For that in our work, we also applied deep learning methods that are used to generate non-linear combinations, where the features that have a lesser effect are automatically given a lower weight. Since our data are labeled, we employed supervised deep learning methods. We applied the Random Forest algorithm over UNSW-NB15 dataset to calculate the feature importance measure for each feature, generate reduced optimal feature vectors. We considered two schemes binary and multiclass classification configurations. We compared different models using their performance results and accuracies. We can conclude that deep learning methods exceed traditional methods with their performance in the attack detection tasks; also feature selection methods can enhance performance in some classical machine learning models.

In the future work, we can compare the experimental results of UNSW-NB15 dataset with other datasets like Bot-IoT, CIC-IDS2018, and N-BaIoT datasets or we can create our own dataset using simulation tools. We aim to be more creative in intrusion detection methods and increase accuracies using hybrid models by combining blockchain with deep learning algorithms. We think to add an intrusion prevention system (IPS) to IDS, this technology used to prevent and mitigate attacks and drop the malicious packets and threats.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

REFERENCES

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). “State-of-the-art in artificial neural network applications: A survey”. *Heliyon,* 4(11), pp. 1-41.

Albawi, S., Mohammed, T.A., & Al-Zawi, S. (2017). “Understanding of a convolutional neural network”. In *2017 International Conference on Engineering and Technology (ICET)* (pp. 1-6).

Alsamiri, J., & Alsubhi, K. (2019). “Internet of things cyber attacks detection using machine learning”. *International Journal of Advanced Computer Science and Applications,* 10(12), pp. 627-634.

Boateng, E. Y., & Abaye, D. A. (2019). “A review of the logistic regression model with emphasis on medical research”. *Journal of Data Analysis and Information Processing*, 7(4), pp. 190-207.

Breiman, L. (2001). “Random Forests”. *Machine Learning,* 45(1), pp. 5–32.

Chandrashekar, G., & Sahin, F. (2014). “A survey on feature selection methods”. *Computers & Electrical Engineering,* 40(1), pp. 16-28.

Chomboon, K., Chujai, P., Teerarassamee, P., Kerdprasop, K., & Kerdprasop, N. (2015). “An empirical study of distance metrics for k-nearest neighbor algorithm”. In *Proceedings of the 3rd International Conference on Industrial Application Engineering* (pp. 280-285).

Graves, A. (2012). “Long Short-term Memory”. *Supervised Sequence Labelling with Recurrent Neural Networks* (pp. 37-45). Springer, Berlin, Heidelberg.

Hochreiter, S., & Schmidhuber, J. (1997). “Long Short-term Memory”. *Neural Computation,* 9(8), pp. 1735-1780.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). “An empirical exploration of recurrent network architectures”. F. R. Bach & D. M. Blei (Eds.), *International Conference on Machine Learning* (pp. 2342-2350). PMLR.

Kasongo, S. M., & Sun, Y. (2020). “Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset”. *Journal of Big Data*, 7(1), pp. 1-20.

Kaviani, P., & Dhotre, S. (2017). “Short survey on naive bayes algorithm”. *International Journal of Advance Engineering and Research Development*, 4(11), pp. 607-611.

Khraisat, A., Gondal, I., & Vamplew, P. (2019). “Survey of intrusion detection systems: techniques, datasets and challenges”. *Cybersecurity,* 2, pp. 1-20.

Manaswi, N. K. (2018). *Deep Learning with Applications Using Python: Chatbots and Face, Object, and Speech Recognition with Tensorflow and Keras.* Apress, Berkeley, CA.

Medsker, L. R., & Jain, L. C. (1999). *Recurrent Neural Networks: Design and Applications*. CRC Press.

Moustafa, N., & Slay, J. (2015). “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”. In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1-6). Australia.

Patel, K. K., & Patel, S. M. (2016). “Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges”. *International Journal of Engineering Science and Computing,* 6(5), pp. 6122-6131.

Safavian, S. R., & Landgrebe, D. (1991). “A survey of decision tree classifier methodology”. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3), pp. 660-674.

Sherstinsky, A. (2020). “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network”. *Physica D: Nonlinear Phenomena*, 404, Article 132306. doi:10.1016/j.physd.2019.132306.

Sze, V., Chen, Y. H., Yang, T. J., & Emer, J. S. (2017). “Efficient processing of deep neural networks: A tutorial and survey”. *Proceedings of the IEEE*, 105(12), pp. 2295-2329. doi:10.1109/JPROC.2017.2761740.

Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). “A review on the long short-term memory model”. *Artificial Intelligence Review*, 53(8), pp. 5929-5955. doi:10.1007/s10462-020-09838-1.

Vinayakumar, R., Soman, K. P., & Poornachandran, P. (2017). “Evaluation of recurrent neural network and its variants for intrusion detection system (IDS)”. *International Journal of Information System Modeling and Design (IJISMD)*, 8(3), pp. 43-63.