Design of Intrusion Detection System using Ensemble Learning Technique in Cloud Computing Environment

Rajesh Bingu*, S. Jothilakshmi
Research Scholar, Department of Information Technology, Annamalai University, Chidambaram, Tamil Nadu 608002-India

***Abstract*—The key advantage of the cloud is that it fluidly propagates to fulfil changeable requirements and provides an environment that is repeatable and can be scaled down instantly when needed. Therefore, it is necessary to protect this cloud environment from malicious attacks such as spamming, keylogging, Denial of Service (DoS), and Distributed Denial of Service (DDoS). Among these kinds of attacks, DDoS has the capability to establish a high flood of malicious attacks on the cloud environment or Software Defined Networking (SDN) based cloud environment. Hence in this work, an ensemble based deep learning technique is proposed to detect attacks in cloud and SDN based cloud environments. Here, the ensemble model is formed by combining K-means with deep learning classifiers such as Long Short term Memory (LSTM) network, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU) and Deep Neural Network (DNN). Initially, preprocessing with data cleaning and standardization is applied to the input data. Meanwhile, a random forest is implemented for extracting the minimum significant features. After that, the proposed ensemble based approach is utilized for detecting the intrusion. This approach is used to enhance the performance of the deep learning classifiers without much computational complexity. This model is trained and evaluated using two datasets as CICIDS 2018 and SDN based DDOS attack datasets. The proposed approach provides better intrusion detection performance in terms of F1 measure, precision, accuracy, and recall. By using the proposed approach, the accuracy and precision value attained is 99.685 and 0.992, respectively.**

Keywords*—Cloud; distributed denial of service; intrusion detection; ensemble; recurrent neural network; convolutional neural network; random forest; gated recurrent unit; K-means clustering; long short term memory

I. INTRODUCTION

Cloud Computing (CC) is a different kind of Internet-based infrastructure for providing Information Technology and various resources such as storage services, hardware equipment, operating system, network infrastructure, and entire software applications to users at low cost [1]. It has the advantage of scalability, higher cost efficiency, faster development, and minimal management effort [2]. The introduction of the cloud is a watershed moment in technological advancement for quick information processing. When a new computing system is introduced, scholars and researchers are concerned about its protection. Securing information processing across any information system has become critical to a knowledge acquisition system's success. Always CC or grid computing enables rapid and location-independent information processing. Because of the location-independent processing, the trust is a major issue among Cloud users when using their resources is a major issue [3], [4].

The complex architecture of CC is vulnerable to several kinds of attacks. Compared to a single Intrusion Detection System (IDS), the detection accuracy is improved with a cooperative IDS system. It is due to limited knowledge of attack patterns or implications [5]. The only solution to this type of threat is the development of effective IDS [6]. The approaches of attack detection by the IDS are of two types, they are signature based and behavior based. Out of these two approaches, the most traditional way of discriminating the normal traffic the malicious traffic is signature based. This approach is capable of achieving higher accuracy, but it is prone to a new type of attack [7], [8].

But the behavior-based IDS achieves better results for a new type of malicious attacks. Hence, the behavior-based IDS performs well when compared to signature-based IDS interms of detection rate and seems to be the most preferable for deployment [9]. Moreover, the classification of the IDS can also be made based on the location of its deployment, and it is two types they are host based and network based [10]. The host based IDS (HIDS) is installed in the space which is nearer to the host in order to capture the intruders, whereas the network-based IDS (NIDS) tends to capture the intruders at the network level [11]. Nowadays, most cloud space is associated with software defined technology to empower its accessibility and reliability for all application services. In this regard, this hybrid environment creates more chancesof launching a high flood of malicious attacks [12].

The anomaly based IDS detects the deviation by analyzing the current system with a predefined normal profile. But it is affected by the issue of false alarms in real world implementation [13]. The hybrid IDS provides protection by combining both anomalies based and signature based detection. It is resolved with the issue of intelligent false alarm technique by involving adaptive algorithms [14]. The performance enhancement of IDS is challenged by considering the features such as access independence, elasticity, sufficient computing power, and scalability. In a distributed system, several applications require shorter response times, and it might require large quantity for heavy

load networks [15]. Due to the delay, these applications are not sufficient to support the applications of CC. In recent years, machine learning (ML) has been used in various fields to resolve issues related to high false alarms and low detection rates [16].

An extreme learning machine (ELM) is a new ML that falls into local minima, and the training is extremely fast. Better scalability and generalization in the learning process are obtained with Support Vector Machine [17]. It is used in several areas for resolving classification and regression problems. To get an optimal representation of the input data, deep learning based approaches such as RNN, Artificial Neural Networks (ANN), Deep Belief network (DBN), Deep Boltzmann Machines (DBM), and Autoencoder are commonly used [18].The performances of this algorithm are further enhanced with the hybrid combination of AlexNet, FractalNet, GoogLeNet, Visual Geometry Group (VGG), and Dense CNN [19], [20]. Based on the position and orientation of the input data, the classification process is hard, and the performance is varied for each network based on the input data. In order to select the optimal deep learning classifier, the ensemble based architecture is proposed. It contains several DNN classifiers in which the better result is taken into consideration. Hence it's right to design an intrusion detection framework for SDN based cloud platforms. To accomplish this task, in this research work, the proposed DL model has been trained and evaluated using two kinds of dataset, i.e., the first dataset is cloud based attack, and the second Dataset is SDN based cloud DDoS attack.

1) Research gap: Most of the prior researches particularly focused on machine or deep learning based approaches and the architecture based on its application. Most of them are based on systematic mapping for providing meaningful and comprehensive research. To the best of our knowledge, there are no researches based on the feasibility of utilizing ensemble learning. In addition to that, no researches include the comparison of the classifiers used in ensemble based technique through systematic mapping. None of the researches consider intrusion detection based on attack type, evaluation metric, and dataset characteristics, and strength and weakness of deep learning approaches. The proposed approach is developed with the consideration of above mentioned research gaps.

The overall contribution of the work can be described as listed below.

- Removing inconsistent or missing values to make the data easier to process. For the traffic instance of the dataset, data pre-processing techniques such as standardization and data cleaning have been applied.
- Extracting the minimal set of discriminative features from the pre-processed data using a random forest algorithm. The complexity and storage space are reduced with a minimal set of discriminative features.
- Clustering the dataset with the K-means algorithm eliminate incorrect detection.
- Classifying the traffic clusters as benign and malicious using the proposed ensemble based deep learning approach. Also, performing multi label classification with these clusters for efficient feature identification. The accuracy is improved with optimal selection of deep learning approach.
- Comparative analysis of the five deep learning classifiers has been done using various performance metrics such as accuracy, precision, recall and F1-measure.

The paper organization is given as follows. Section II describes the related work, and Section III describes the proposed methodology. The experimental results of the proposed intrusion detection are given in Section IV. Section V describes the significant aspects of the proposed methodology and conclusion.

II. Related Work

The work related to the proposed intrusion detection system is described as follows.

Loheswaran Karuppusamy et al. [21] had proposed a Chronological Salp Swarm Algorithm-based Deep Belief Network (CSSA-DBN) for detecting intrusion into a cloud environment. The optimal solution was obtained with the fitness, which accepts a minimum error value for providing better performance. The accuracy, sensitivity, and specificity obtained with the CSSA-DBN approaches are 0.9618%, 0.9702%, and 0.9307%, respectively.

Idhammad et al. [22] proposed an ensemble classifier-based intrusion detection model is ideally suited for the cloud environment. The model used in this work was trained and assessed using the CICIDS-001 dataset, and it is currently running on the Google Cloud platform.Here, Naive Bayes and the random forest method are used to build the ensemble classifier. This system took 0.23 seconds to run and had an average accuracy of 97% and a false positive rate of 0.2%. Jaber et al. [23] developed with ensemble classifiers that are made up of fuzzy c-means clustering and SVM classifiers. The hybrid algorithm FCM–SVM was evaluated with NSL-KDD Dataset for detecting anomalies with higher accuracy.

S. Krishnaveni et al. [24] recommended a univariate ensemble feature selection method to find an appropriate reduced feature set from an incursion dataset. To create robust classifiers using a voting mechanism, single classifiers were fused. With performance indicators like FAR and ROC, this technique performed admirably enough.Nguyen et al. [25] proposed a security framework forSDN enabled cloud environment by combining theintrusion detection model based on three different nodes such as edge, fog and cloud. By developing policies, a collaborative and network intelligent architecture is created for anomaly detection. Better anomaly detection performance in SDN-based cloud IoT networks helped to reduce the bottleneck issue.

Using the Ant Lion optimization strategy, T. Thilagam et al. [26] proposed an improved Recurrent Convolutional Neural Network (RCNN) for intrusion detection. With a classification accuracy and a small error rate are 94% and

0.0012, network layer threats are well categorized. Smitha Rajagopal et al. [27] had developed a Meta classification technique with binary and multi-label classification. Robustness has been improved with an optimal set of hyperparameters and discriminative features of Azure machine learning. The efficiency of the approaches was validated, and an accuracy of 99.8% was achieved for the UNSW NB-15 dataset.

Abusitta et al. [28] proposed a cooperative intrusion detection framework for the cloud environment, and it was designed using a stacked autoencoder and multilayer perceptron. The decision making was enabled with an aggregation algorithm, in which the detection accuracy was achieved by up to 95%. Ammar Aldallal et al. [29] developed SVM with GA and fitness for evaluating accuracy. SVM was deployed with varying hyperparameters such as kernel, degree, and gamma. In cloud computing, a high level of symmetry was reached between attack detection, information security, and the discovery of bad things.

Mayuranathan et al. [30] proposed an effective intrusion detection model based on RHM-RBM. Here the author utilizedRandom Harmony Search (RHS) optimization model for feature selection and Restricted Boltzmann Machines (RBM) for classification purposes to yield better results. The security issues related to the network layer have been resolved with enhanced detection accuracy and low computational complexity.

At the end of the survey analysis, it can conclude that most of the existing solution does not rely on the ensemble-based approach using a deep learning algorithm to enhance its efficiency without much computational complexity. Hence in this work, an ensemble based deep learning technique has been deployed. To achieve this, clustering followed by a classification task has been carried out. By doing so, the unsupervised technique (clustering) collaborates with the supervised technique (classification). A convolutional neural network and the K-means clustering procedure are used to carry out this strategy. Convolutional neural networks (CNN) and the other four deep learning algorithms (DNN, RNN, GRU, and LSTM) are evaluated in terms of performance.

III. Proposed Methodology

In Fig. 1, the components involved in the proposed model have been elucidated, and its data flow can also be visualized. The modules involved in the proposed model are the data pre-processing layer, feature extraction layer, clustering process and classification. Each module has been explained in the following subsection

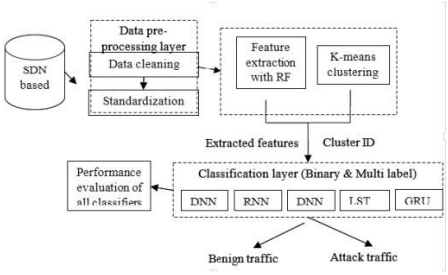


Fig. 1. Overall architecture of the proposed intrusion detection system.

A. Data Pre-Processing

In the data pre-processing layer, the instances of the two datasets are manipulated accordingly to ease the classification process. These instances have undergone two pre-processing techniques, such as data cleaning and standardization.

1) Data cleaning: In the CICIDS-2018 Dataset**,** some columns with infinity values were stripped away, including 'Dst Port', 'Timestamp', 'Bwd PSH Flags', 'Fwd PSH Flags', 'FlowByts/s', 'Bwd URG Flags', 'Fwd URG Flags' and 'Flow Pkts/s'. Two protocol columns have been used instead of feature protocol for binary classification.They are named Protocol 17 and Protocol 6 by implying the feature ina categorical type. The 'protocol' column has been used without alteration for the multiclass classification.Likewise, for the SDN DDoS dataset, the null values are removed for all the features.

2) Standardization: To get a normal distribution with a mean of zero and a standard deviation of one, standardized measures are applied to each input variable independently by subtracting the mean and dividing by the standard deviation. In standardization, the values are scaled in column wise manner using the standard scalar format. This process facilitates the classification process, which is easily performed using deep learning.

B. Feature Reduction

The random forest algorithm has been utilized for feature extraction in this work. Random forest is an effective unsupervised machine learning technique that falls within the area of embedded methods.For the predictor variable, the subset is chosen for dividing the internal node based on predetermined constraints, considered an optimization issue. The classification is based on entropy, which specifies the lower bound of the random variable. The entropy is computed as follows for each internal node of the decision tree.

$$F = -\sum_{j=1}^{d} q_j \times \log(q_j) \quad (1)$$

Where, d represents the amount of unique classes and the prior probability for the class is represented as q_j. This value is increased to obtain more information in each decision tree split.

The embedded method combines both the quality of the filter and wrapper method. Furthermore, it can be used for classification and feature extraction. So Random Forest inherently has a built-in feature selection approach. Since it is well suited for feature selection, a Random forest has been used for the feature selection task.The reduced features of the SDN based dataset can be listed as dt, bytecount, packetins, pktperflow, byteperflow, pktrate and Protocol. In CICIDS 2018 dataset, the features are reduced, and it can be elucidated as 'Fwd Seg Size Min', 'SubflowBwd Pkts', 'Tot Bwd Pkts', 'Fwd Pkt Len Std', 'Flow IAT Mean', 'Init Fwd Win Byts', 'Bwd Pkt Len Max', 'URG Flag Cnt', 'FIN Flag Cnt', 'Bwd Pkt Len Std',' Pkt Size Avg' and 'RST Flag Cnt'. Fig. 2 and Fig. 3 show the robust feature set of the SDN based dataset

and cloud dataset correspondingly. Here, random forest is evaluated with 5-fold cross validation to extract some meaningful features from the two datasets separately.

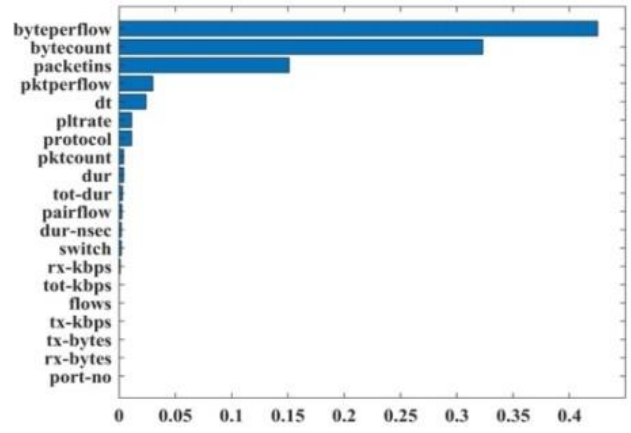


Fig. 2. Significant feature in SDN dataset by random forest.

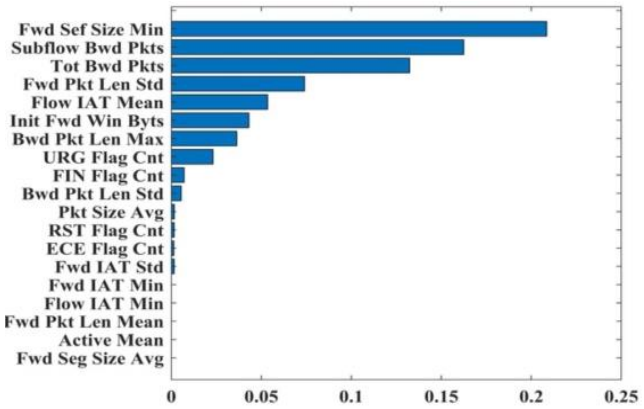


Fig. 3. Significant feature in CICIDS 2018 dataset by random forest.

C. Clustering Layer

The process of clustering is to group the instances given in the dataset into many clusters. This process assigns each instance a unique cluster ID. This was done based on a pattern extracted by the K-means clustering algorithm. Therefore, along with the reduced features yielded by the random forest, the Cluster id was also given to the Deep learning classifiers to improve its performance. The working principle of the K-means clustering process is given below:

1) K-means clustering algorithm: K-means clustering always work in an unsupervised manner by grouping the instances in the dataset without a label for the training and testing process. Based on the predefined value, the number of clusters is generated by this clustering algorithm. This algorithm forms the clusters based on their centroid value. The main goal of the clustering algorithm is to shorten the distance between the data instances and the groups to which they belong. This step is repeated continuously until the algorithm finds the better clusters. At the end of the process, 'K' number of clusters has been obtained, whereas k is a predetermined value. The procedure of the K-means algorithm is given as follows.

Algorithm 1: K-means clustering
Step 1: Specify the number of clusters k .
Step 2: Assign k centroids randomly.
Step 3: repeat, until the position is not varied.
Step 3.1: Closet centroid is assigned with each point.
Step 3.2: For each cluster, a new centroid is computed with mean value.

D. Classification Layer

At this level, the reduced features, as well as the cluster ID were given as input for the classification process. To classify the normal traffic instances from the malicious traffic instances, five deep learning techniques were implemented separately for this task. These five deep learning were analyzed comparatively for both binary and multiclass classification. To perform binary classification, SDN Dataset was used for training and testing purposes, while CICIDS 2018 was used for the same purpose for multiclass classification. A detailed explanation for these five DL classifiers is given in the following sub-sections.

1) Convolutional Neural Network (CNN): Convolutional Neural Network is also known as Convnets and is mainly used for image processing and object detection purposes. This CNN has several layers the process goes through to get the desired output. The architecture of CNN consists of many layers, namely Convolution Layer, Rectified Linear Unit, Pooling Layer, andFully Connected Layer. The input goes through all of these layers, each of which contains a variety of operators and filters to get the right output.

The convolution layers are interpreted with the sub-sampling layers to minimize the computation time.In the convolution layer, the feature map from the previous layer is mixed with kernels that can be provided, and the resulting feature map is sent to the activation function. The convolution is combined with multiple feature maps in each output map. In general, it is represented as,

$$y_k^u = g\left(\sum_{j \in N_k} y_k^{m-1} * l_{j,k}^m + c_k^m\right) \quad (2)$$

Where, N_k is the input map selection which includes the pair of all triplets. For each output map additive bias of c is added. The input is convolved with various kernels for each output map. If the output map k and l are integrated with the output map j . Then the kernel is applied to various output maps k and l .

2) Deep Neural Network (DNN): An artificial neural network (ANN) with numerous hidden layers between the input and output layers is called a deep neural network (DNN). DNNs may simulate complex non-linear interactions just like shallow ANNs. This particular kind of neural network consists of an input, an output, and a deep network of sequential data flow. In order to solve practical problems like

categorization, neural networks take in data, run it through complex calculations, and then output the results.

DNN has several fully connected layers in which nodes of each layer are connected with each node of the previous layer. The DNN is a linear combination of independent variables with corresponding weights and bias terms. The DNN output computation is represented as,

$$A = c + x_1 y_1 + x_2 y_2 \ldots + x_p y_p \quad (3)$$

Where, x represents the weights or beta coefficients and y represents the input or independent variable. The loss or error term is computed to find the deviation from actual and predicted values. It has the objective of minimizing the loss function in order to achieve an optimal error term. Based on the previous layer computation, the output is estimated as follows:

$$A = x_{i0}^T * i_2 + c_{i0} \quad (4)$$

Where, x_{i0} is the weight matrix between two layers, c is the bias, and T represents the transpose. After estimating the output, it passes through the activation function for computing the node value. The output error is estimated, and the error is minimized with the optimal weight value.

3) Recurrent Neural Network (RNN): Recurrent neural networks (RNNs) are artificial neural networks where nodes' connections can cycle, allowing output from one node to control how input to that node is processed. The term "recurrent neural network" refers to a group of networks that have an infinite impulse response. Common uses of RNNs include natural language processing, time series analysis, handwriting recognition, machine translation, and photo captioning. RNNs are capable of handling inputs of any length. In contrast to infinite impulse recurrent networks, finite impulse recurrent networks can be unrolled and substituted with tight feedforward neural networks.

In a RNN, the output of a certain layer is fed into the input of the layer before it to predict the output layer. A single RNN layer is created by combining the nodes of several NN layers. The input layer is represented with y, the output layer is represented with z, and i represents the hidden layer. The output of the model is increased with the network parameters. For the given duration u, the input is estimated with the integration of $y(u)$ and $y(u-1)$. The output of each state is represented as,

$$i(u) = g_d(i(u-1), y(u)) \quad (5)$$

Where, $i(u)$ represents the new state, $i(u-1)$ represents the old state, g_d represents the function with parameter d, and $y(u)$ is the input vector with time u.

4) Long Short-Term Memory (LSTM): It is a kind of RNN that can recall and learn long-term dependencies. Due to their ability to remember past inputs, they are also utilized in time series prediction. They communicate in an original fashion thanks to their chain-like arrangements with four interacting levels. These are employed for purposes other than only time series prediction. Additionally, they are employed in medicinal research, music composition, and voice recognition. It is connected in such a way that directed cycles are formed, and it permits the LSTM output to be used as the input of the current layer. It can also recall previous inputs due to its internal memory.

LSTM is modelled to avoid long term dependencies, and it has three parts. The first eliminates irrelevant information, the second updates or adds new information, and the third pass the updated information. In LSTM, it initially decides to keep the information obtained from the previous step or not. The forget gate equation is represented as follows.

$$f_u = \sigma(y_u * v_g + i_{u-1} * x_g) \quad (6)$$

Where, y_t represent the current timestamp of the input, i_{u-1} represent the hidden state of the previous timestamp, v_g denotes the weight of the input, and x_g is the weight matrix of the hidden state. The sigmoid function is applied to make the forget gate between 0 and 1. Then it is multiplied by the timestamp of the previous layer.

$$d_{u-1} * g_u = 0 \ \ if \ \ f_u = 0 \quad (7)$$

$$d_{u-1} * g_u = d_{u-1} \ \ if \ \ g_u = 1 \quad (8)$$

The significance of new information is quantified with the input gate, and the equation is denoted as,

$$i_u = \sigma(y_u * v_u + i_{u-1} * x_j) \quad (9)$$

Where, y_t represents the input of the current timestamp, v_u represents the weight, and x_j represents the hidden state. The output of the current timestamp is estimated using the activation function of softmax.

$$z = soft\max(i_u) \quad (10)$$

Where, i_u represents the hidden state.

5) Gated Recurrent Unit (GRU): In order to overcome the vanishing exploding gradient problem faced by the Recurrent neural network, many variants of RNN have started to occur. Out of those findings, the GRU(gated recurrent unit), a variant of RNN architecture, has seemed to perform well. The architecture of GRU comprised three gates without any internal cell state. Instead of an internal cell state present in the LSTM architecture has been replaced by a hidden cell state

in the architecture of GRU. The gates used in this architecture can be listed as Update Gate, Reset Gate and Current Memory Gate.

The input of GRU is represented as y_t, a previous timestamp $u-1$, and the hidden state is represented as i_{u-1}. The new hidden state is the output of the next timestamp; and it contains two gates, namely the reset gate and the update gate. The reset gate is considered a hidden state i_u. The equation for the reset gate is represented as follows.

$$s = \sigma(y * v + i * x) \tag{11}$$

Where, v, x represents the weight, y represents the input. By using the sigmoid function, the values of s is converted within the range between 0 and 1.The update gate is similar to the reset gate, only the weight matrix is varied. To estimate the hidden state i_u, the two state processes are used. Initially, the candidate's hidden state is estimated with the following equations.

$$\hat{i}_u = \tanh(y_u * v_h + (s_u * i_{u-1}) * x_h) \tag{12}$$

The input is taken from the hidden and previous timestamp and multiplied by the output of the reset gate. The overall information is passed through the activation function tanh; the resultant value is the candidate's hidden state. The GRU network is accurate in a longer sequence dataset. The information in GRU is transferred through the cell state and hidden state.

IV. Experimental Results and Discussion

The Python 3.6 software environment was utilized to create the deep learning model and the intrusion detection system based on clustering method for this experiment. Python was installed over the Windows 10 operating system, which is on the laptop and has 2GB ram and 1TB hard disk memory with a 2GHz i3 processor. Initially, the input data is pre-processed with data cleaning and standardization. In data cleaning, the corrupted, incorrectly formatted, incomplete, incorrect, or duplicate data are eliminated within the dataset. Then the data is converted into a simplified format to simplify the intrusion detection process. The features are extracted with 5-fold cross validation of the random forest algorithm. On the other hand, data is grouped into clusters, and each instance is assigned a unique cluster ID. The extracted features and the cluster ID is given to the input of the ensemble classifier. In the proposed algorithms, the parameters are included based on the existing results. In the existing papers, these parameters provide better performance than assigning other parameter. Hence, the parameters are selected for enhancing the intrusion detection performance. By using this parameter optimal level of intrusion detection was attained with the proposed ensemble based approach. Table I provides a description of the implementation parameters used in the suggested technique.

TABLE I. Parameters of Deep Learning Techniques

Parameters	Value
Number of clusters	4
Batch Size	Binary:2500, Multilabel:5000
Loss Function	Binary: Binary_crossentropy Multi-label: Categorical_crossentropy
Activation	ReLU
Epoch	150
Verbose	0
Metric	Accuracy
Optimizer	Adam
Epoch	Binary:15 Multilabel:10

A. Dataset Description

1) SDN based dataset: This dataset was designed to generate a DDoS attack by the mininet simulator in order to replicate the Software defined networking environment [15]. Here the network traffic was collected at the switch setup in the environment. The instances given in the dataset were broadly classified into two categories: benign and malicious. Hence the benign instances were labelled as 0, whereas the malicious instances were labelled as 1. SDN based datasetsare used to train and evaluate the proposed model for binary classification.

2) CIC IDS 2018: This is one of the datasets used to train and evaluate the proposed model, which is extracted from the official website [16], and it is simulated in the real time cloud environment to capture cloud based attacks. This dataset is utilized here to perform a multiclass classification of cloud attacks using an ensemble based approach. It has 2830540 instances and 83 attributes. Among these 83 attributes, 80 attributes are utilized for feature extraction procedure.

The benign instances in the dataset were labelled as 0, and bot attack instances were labelled as 1. Likewise, the instances of DoS attacks-SlowHTTPTest, and DoS attacks-Hulkwere labelled as 2 and 3, respectively. Table II elucidates the sample distribution of benign and attack instances used for training and testing purposes.

TABLE II. Samples Distribution for DL Models

	SDN based DDoS attack dataset (Binary classification)		CIC IDS 2018 dataset (Multi-label classification)	
Type	Number of samples			
	Training	Testing	Training	Testing
Benign	25496	10902	646603	161601
Attack	20404	8735	86811	21709
DoS attacks-SlowHTTPTest			34	7
DoS attacks-Hulk.			87117	21811

B. Performance Metrics

In this sub-section, the performance of both binary as well as multi-label classification done by the five classifiers was evaluated and discussed using the below given metrics:

Accuracy: It estimates the classifier performance in overall, which is computed as follows.

$$A_c = \frac{TN + TP}{TN + FN + TP + FP} \quad (13)$$

Where, TP represents the true positive, FP represents the false positive, TN represents the true negative, and FN represents the false negative.

3) Precision: It represents the capacity of the classification models to classify the significant models of the data set. It is calculated using the ratio of expected positives from all samples. Precision is denoted as follows.

$$P_r = \frac{TP}{FP + TP} \quad (14)$$

4) Recall: It represents the capacity of the classification technique for categorizing essential data points in the dataset. It is measured as the ratio of positives from the whole set of positive samples. Recall R_c can be computed as

$$R_c = \frac{TP}{FN + TP} \quad (15)$$

5) F-measure: It uses the mean value to combine the result of precision and recall. F-measure F_m is measured as,

$$F_m = \frac{2}{1/P_r + 1/R_c} \quad (16)$$

6) Receiver operator characteristic curve: ROC curves are a useful visual tool for comparing different classifiers. It describes the trade-offs that could be made between a false positive rate (FPR) and a true positive rate (TPR). The model's ROC curve accuracy is evaluated using the Area Under the Curve (AUC).

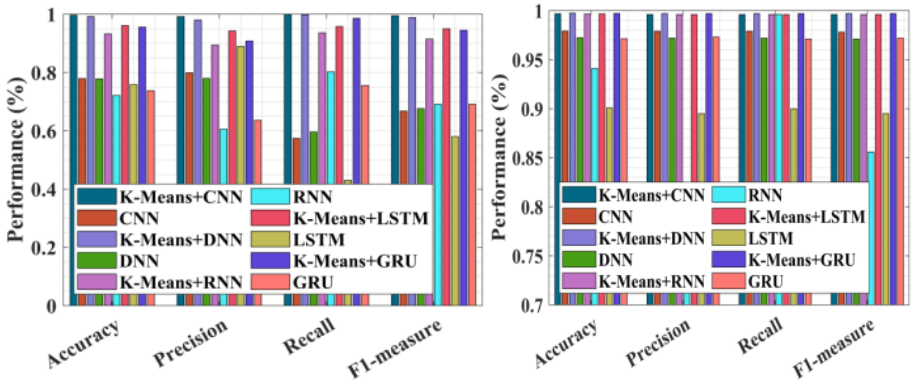
Where the performance monitors used in the above-mentioned equations can be defined as

- TP (True Positive): It is defined as the count of the attack instancessuccessfully predicted as an attack by the classifier.
- TN (True Negative): It is defined as the count of the benign instances successfully predicted as benign by the classifier.
- FP (False Positive): It is defined as the count of the benign instances wrongly predicted as an attack by the classifier.
- FN (False negative): It is defined as the count of the attack instances wrongly predicted as benign by the classifier.

In Table III, the values of the performance metrics for the binary classification have been enumerated. In this table, each classifier is implemented with and without clustering separately. The clustering process is carried out using the K-means algorithm. By observing the values for the five classifiers, it shows that the accuracy value without K-means ranges from ~72% to ~77%. But with K-means implementation for the five classifiers achieves better results for accuracy; it ranges from ~93% to ~99%. Hence it is clearly shown that the binary classification, the implementation of the ensemble approach, performs better than the standalone DL architecture. Out of those five DL classifiers, CNN performs well than the others. In the same way, Table III explores the performance analysis of multi-label classification. Unlike binary classification, the results of the standalone DL algorithm and the ensemble approach show only a minimal gap. Both models yield good results for multi-label classification. In this classification, DNN performs better than all four DL classifiers.

TABLE III. PERFORMANCE ANALYSIS OF BINARY AND MULTI-LABEL CLASSIFICATION

	Binary classification				**Multi-label classification**			
Algorithm Used	**Accuracy**	**Precision**	**Recall**	**F1-measure**	**Accuracy**	**Precision**	**Recall**	**F1-measure**
K-Means+CNN	**99.685**	**0.992**	**0.999**	**0.995**	99.685	0.996	0.996	0.996
CNN	77.911	0.799	0.574	0.668	97.906	0.979	0.979	0.978
K-Means+DNN	99.178	0.980	0.998	0.989	**99.735**	**0.997**	**0.997**	**0.997**
DNN	77.821	0.780	0.596	0.676	97.244	0.972	0.972	0.971
K-Means+RNN	93.262	0.894	0.936	0.915	99.649	0.996	0.996	0.996
RNN	72.162	0.606	0.802	0.691	94.111	0.790	0.996	0.856
K-Means+LSTM	96.128	0.943	0.958	0.950	99.671	0.996	0.996	0.996
LSTM	75.853	0.890	0.430	0.580	90.076	0.895	0.900	0.895
K-Means+GRU	95.602	0.908	0.986	0.945	99.712	0.997	0.997	0.997
GRU	73.767	0.636	0.756	0.691	97.135	0.973	0.971	0.972



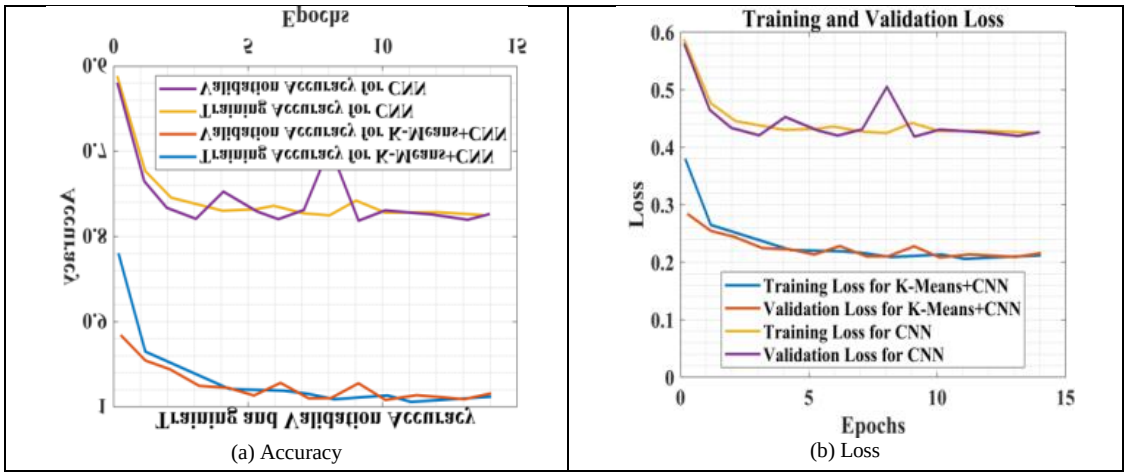
(a) Binary classification (b) Multi-label classification

Fig. 4. Comparison of accuracy, precision, recall, and F1-measure with different voting techniques.

The suggested ensemble-based strategy for binary and multi-label classification is compared in Fig. 4 for both cases. It is evaluated in terms of accuracy, F-measure, recall, and precision. For binary classification, better performance is obtained using the K-means+CNN approach. The accuracy, precision, recall, and F1-measure obtained with K-means+CNN approach are 99.685, 0.992, 0.999, and 0.995, respectively. Compared to other deep learning-based approaches, K-means+DNN performs better on multilabel-based classification. K-means with deep learning approaches provide better performance than only deep learning based approaches. If the clustering is not performed for the proposed approach, the performance is lower than 0.8.

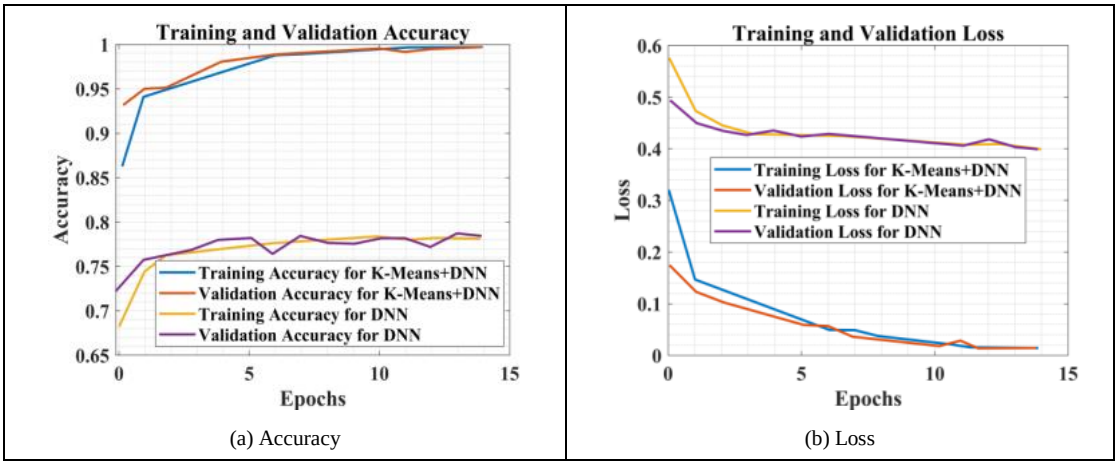
Fig. 5 compares the accuracy and loss for the CNN and K-means+CNN techniques. The accuracy improves and the loss decreases as the number of epochs rises. The accuracy of validation is greater than that of training. The optimal value of accuracy is reached with the epoch between 10 and 15. The lower loss value is reached with a higher epoch that is nearer to 15. The accuracy and loss comparison with DNN and K-means+DNN for binary classification is shown in Fig. 6. Increased accuracy and decreased loss result from more approaches. The optimal accuracy is obtained with the number of epochs 14. When the number of epochs reaches 3, the accuracy rate crosses the value of 0.9. The accuracy below 0.75 is reached, and the loss value is higher up to the number of epochs is 3. When it goes beyond 3, there is a gradual decrease in loss, and the smooth curve is obtained up to 14 epochs.

The accuracy and loss comparison for GRU and K-means+GRU is shown in Fig. 7. There is a gradual increase in accuracy value from 1 to 5 epochs. After 5, the accuracy deviation is low, and it isn't very important. For the GRU approach, this deviation is higher up to 14 epochs. The training accuracy is lower than the validation accuracy in all aspects. When the numbers of epochs are 3, the loss is higher than 0.5 and 0.15 for the validation set and training set of the K-means+GRU approach.



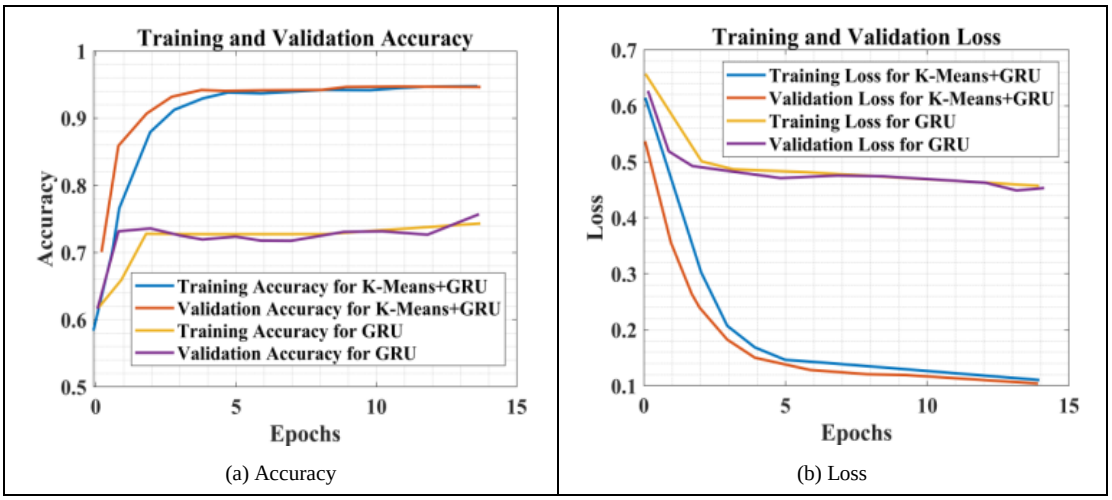
(a) Accuracy (b) Loss

Fig. 5. Accuracy and loss comparison for training and validation set of CNN, and K-means+CNN (Binary Classification).



(a) Accuracy (b) Loss

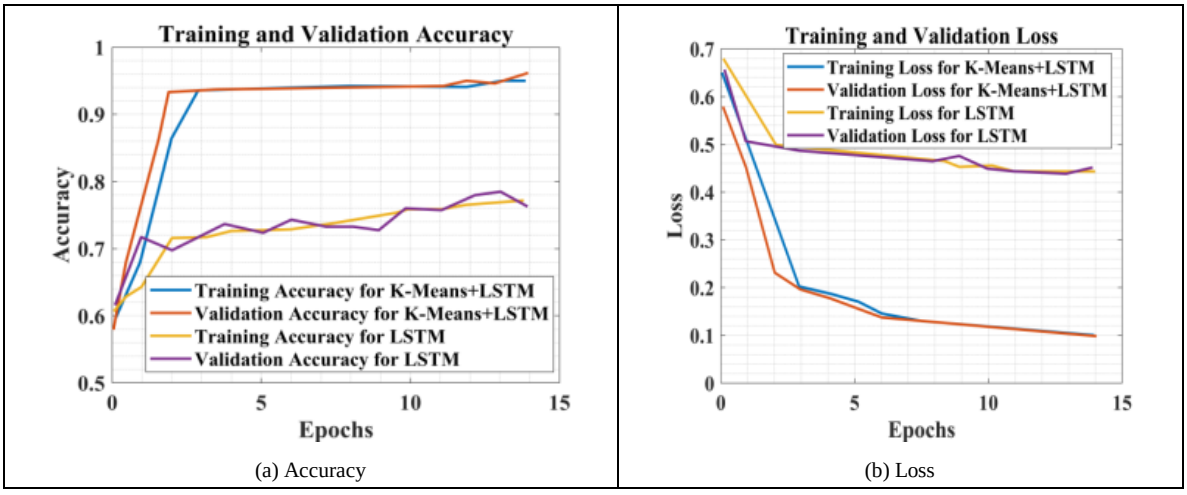
Fig. 6. Accuracy and loss comparison for training and validation set of DNN and K-means+DNN (binary classification).



(a) Accuracy (b) Loss

Fig. 7. Accuracy and loss comparison for training and validation set of GRU and K-means+GRU (binary classification)

The accuracy and loss comparison for K-means+LSTM and LSTM approaches are shown in Fig. 8. There is a fluctuation in the accuracy value as the number of epochs is increased in the LSTM approach. The accuracy is above 0.9 for most epochs in the training and validation set of K-means+LSTM approaches. The accuracy loss is higher than 0.5 for LSTM and K-means+LSTM with fewer epochs. The lower loss is reached with the number of epochs 14.



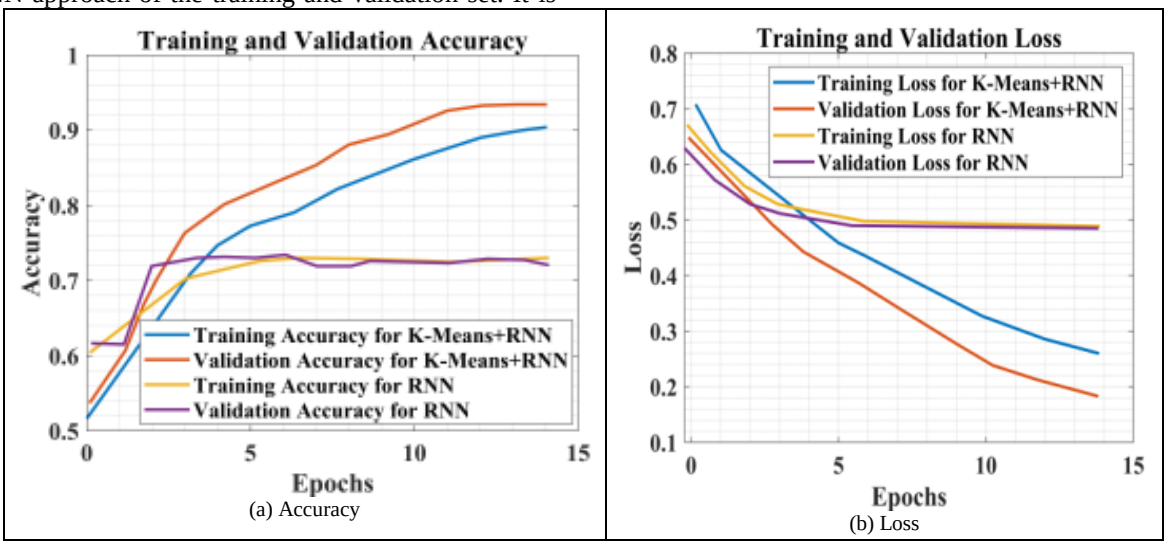
(a) Accuracy (b) Loss

Fig. 8. Accuracy and loss comparison for training and validation set of LSTM, and K-means+LSTM (binary classification).

The performance evaluation of RNN and K-means+RNN for binary classification is shown in Fig. 9. The K-means+RNN and the validation set provide better performance than only RNN and the training set. When the number of the epoch is 3, k-means+ RNN and RNN approaches interfere with each other. When the number of epochs is decreased, the variation in loss for the training and validation sets is also decreased. When the number of epochs is decreased, the loss deviation for training and validation is also decreased. The training and validation loss is less than 0.5 for the K-means+RNN approach. For only the RNN approach, the training and validation loss is less than 0.3.

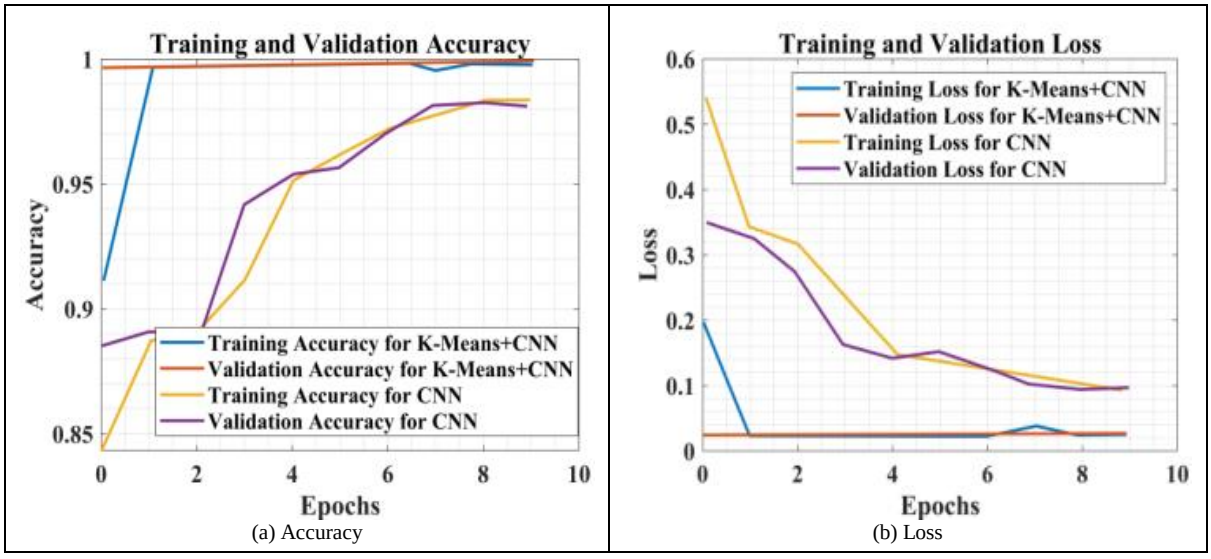
The accuracy and loss comparison for the multi-label classification of K-means+CNN is shown in Fig. 10. For all epochs, the validation accuracy is higher and nearer to 1. But, for the training set, the accuracy is lower with a reduced number of epochs. The accuracy value is between 0.85 and 0.9 for the CNN approach of the training and validation set. It is increased to the level between 0.95 and 0.99 for the numbers of epochs are 8. The loss value is lower for the validation set of the K-means+CNN approach. For the training set, it is higher with lower epochs. For lower epochs, the loss value is between 0.5 and 0.6. The loss is reduced to 0.1 for the number of epochs between 8 and 10.

The accuracy and loss comparison for the DNN and K-means+DNN approaches is shown in Fig. 11. The accuracy value is higher with the validation set of K-means+DNN approaches. The training accuracy is lower than the validation set. The higher value is reached with the number of the epoch is 4, and the value remains the same up to 10 epochs. K-means+DNN has a lower training and validation loss than the DNN-based method. The loss is decreased when the number of epochs is increased. The loss value of GRU and K-means+GRU is compared for the training and validation set. The loss is decreased with an increasing number of epochs.



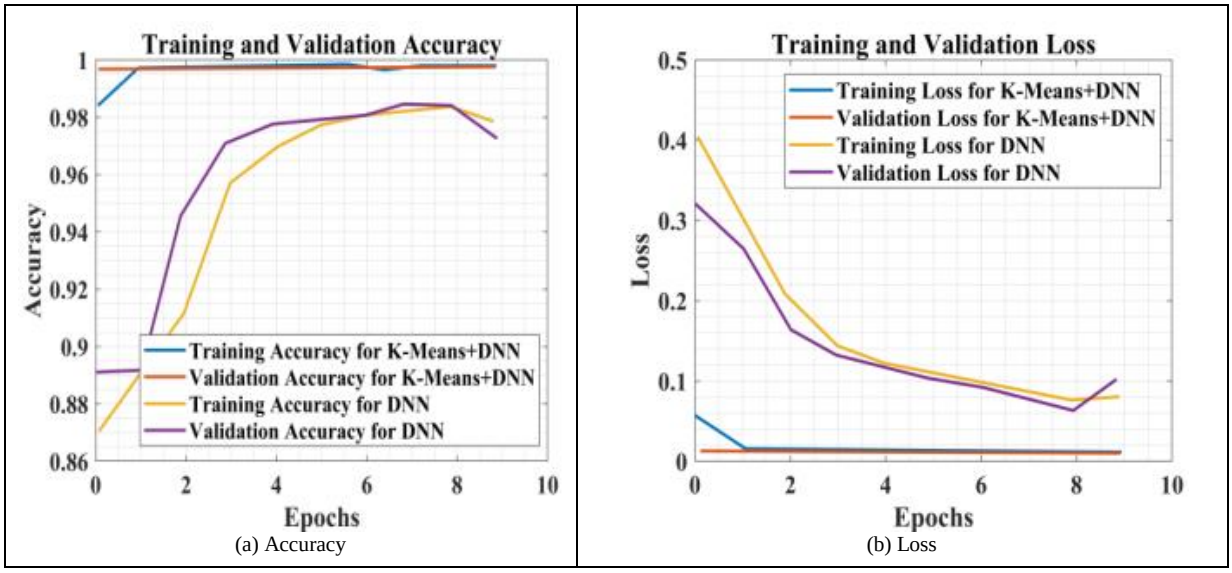
(a) Accuracy (b) Loss

Fig. 9. Accuracy and loss comparison for training and validation set of RNN and K-means+RNN (binary classification).



(a) Accuracy (b) Loss

Fig. 10. Accuracy and loss comparison for training and validation set of CNN, and K-means+CNN (multi-label classification).

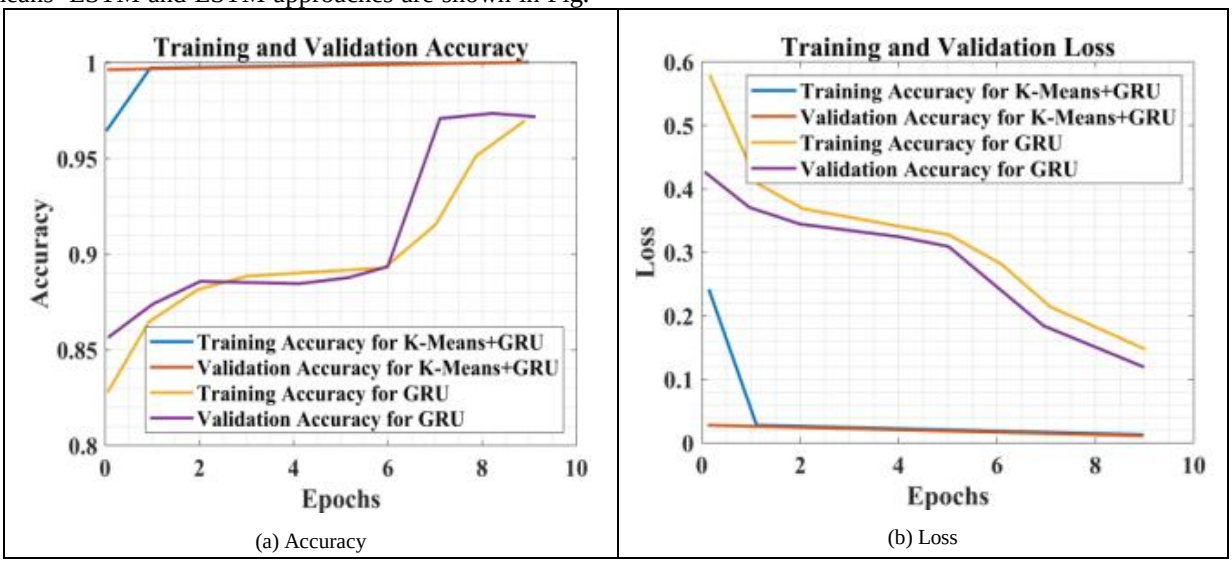


(a) Accuracy (b) Loss

Fig. 11. Accuracy and loss comparison for training and validation set of DNN and K-means+DNN (multi-label classification).

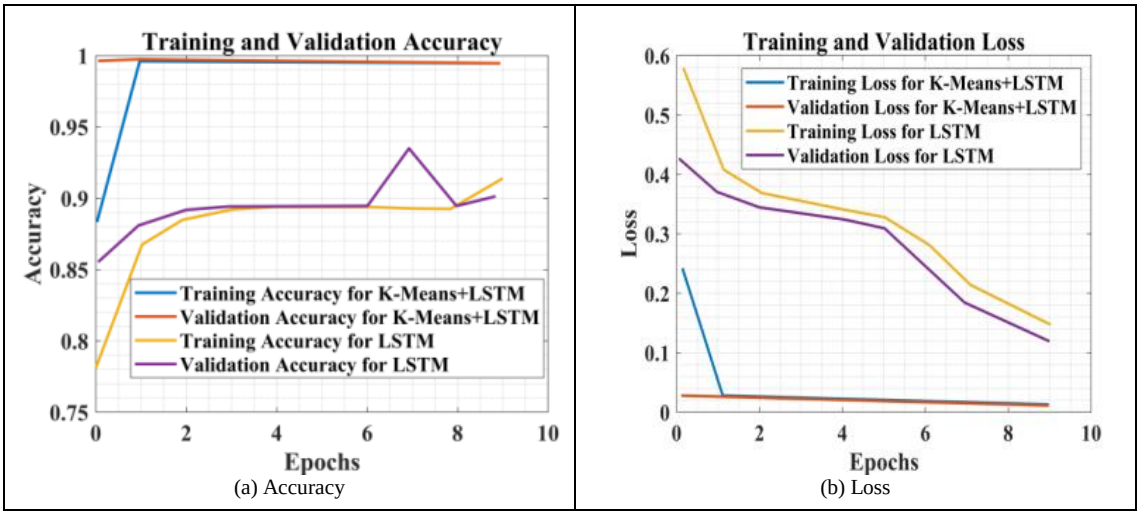
The multi-label classification of the training and validation set for K-means+GRU and GRU based approaches are given in Fig. 12. The accuracy is higher with the K-means+GRU approach, and it is lower with GRU based approach. If the number of the epoch is 1, the training accuracy of K-means+ GRU is lower than the validation accuracy.

The accuracy comparisons with multi-label classification for K-means+LSTM and LSTM approaches are shown in Fig. 13. When the number of epochs is 7, the validation accuracy of LSTM highly deviates from the training accuracy of LSTM. The validation accuracy of K-means+LSTM achieves a constant value nearer to 1. With a lower amount of epochs, the accuracy value is lower than 0.9. The value of the training and validation set is lower for K-means+LSTM and higher for LSTM.



(a) Accuracy (b) Loss

Fig. 12. Accuracy and loss comparison for training and validation set of GRU and K-means+GRU (multi-label classification).

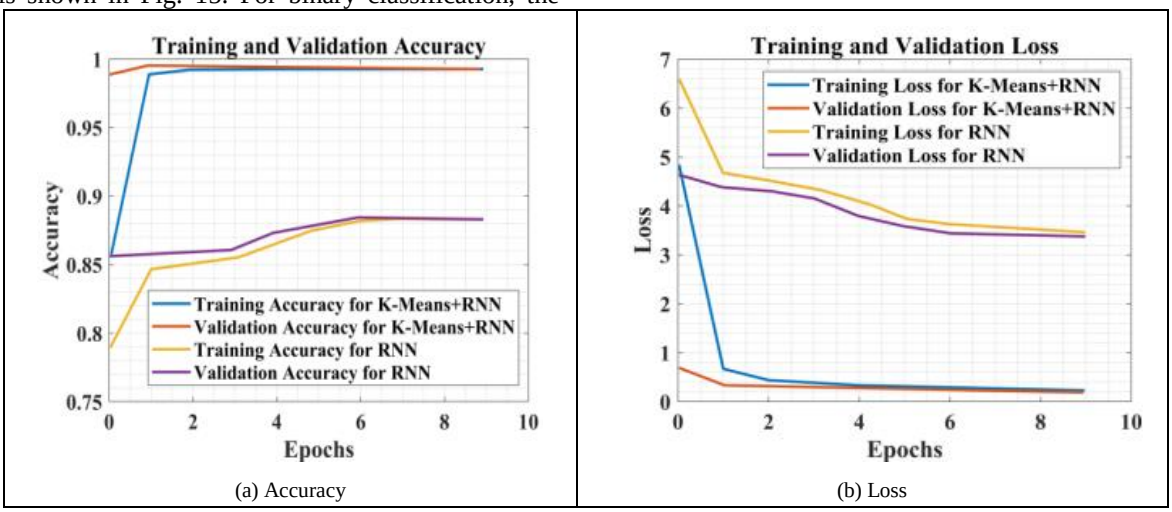


(a) Accuracy (b) Loss

Fig. 13. Accuracy and loss comparison for training and validation set of LSTM and K-means+LSTM (multi-label classification).

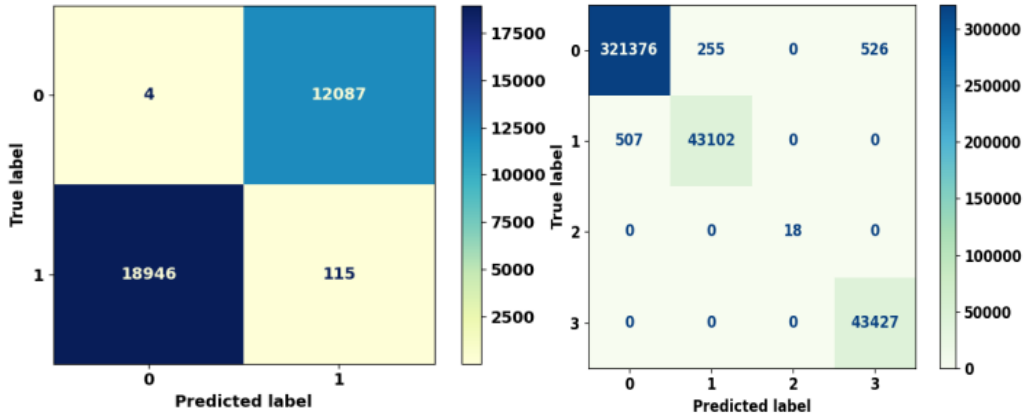
The accuracy comparisons of multi-label classification for K-means+RNN and RNN approaches are shown in Fig. 14. The accuracy is higher for the K-means +RNN approach than the RNN approach. The training accuracy is lower than the validation accuracy. By increasing the number of epochs, the accuracy is increased for K-means+RNN and RNN approaches. The loss value is lower for the K-means+RNN approach, whereas it is higher for the RNN approach. When the number of the epoch is 9, the training and validation accuracy for K-means+RNN and RNN is the same.

The confusion matrix for the proposed ensemble-based approach is shown in Fig. 15. For binary classification, the CNN-based approach provides better detection performance; for multi-label classification, DNN provides better intrusion detection results. The precision recall curve for the ensemble approach is shown in Fig. 16. If the recall value is closer to 1, then the precision also gets closer to 1. The best results from the multi-label categorization are displayed in Fig. 16. Fig. 17 depicts the receiver operating characteristic curve for the ensemble method. It's a graph made up of true positive and false positive numbers. The DNN method of multi-label classification has achieved a macro-average ROC of 1. The ROC curve value reached for class 0 is 1.00, class 1is 0.99, class 2 is 1.00, and class 3 is 1.00.



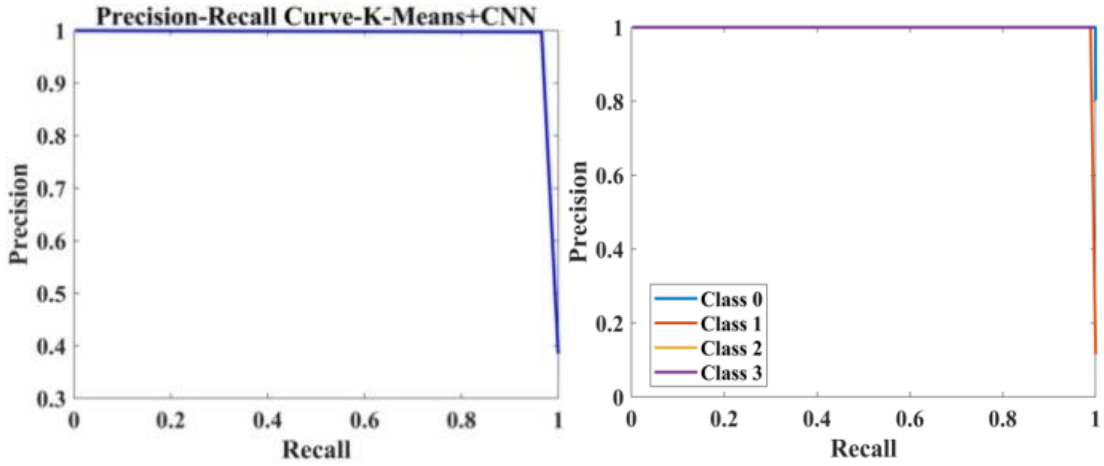
(a) Accuracy (b) Loss

Fig. 14. Accuracy and loss comparison for training and validation set of RNN and K-means+RNN (multi-label classification).



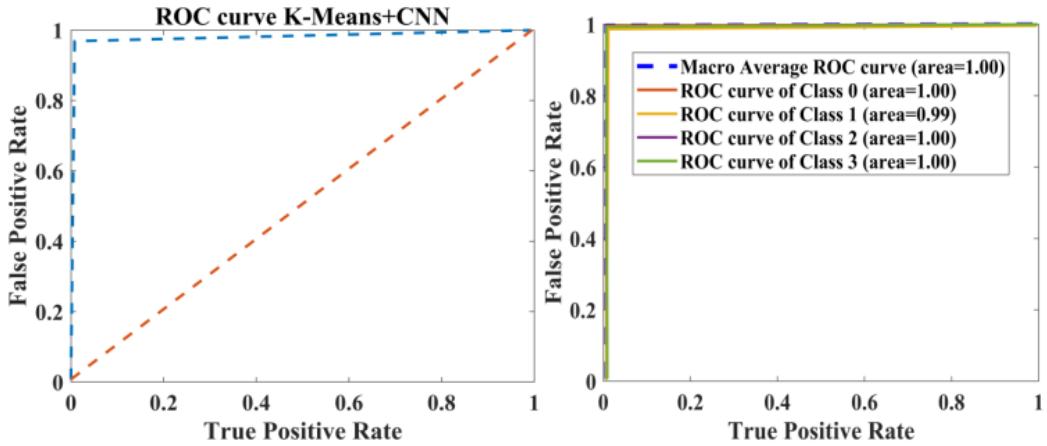
(a) CNN (Binary) (b) DNN (Multi-label)

Fig. 15. Confusion matrix for an ensemble approach.



(a) CNN (Binary) (b) DNN (Multi label)

Fig. 16. Precision-Recall Curve for ensemble approach.



(a) CNN (binary) (b) DNN (multi-label)

Fig. 17. ROC curve for an ensemble approach.

V. Conclusion

In this work, an intrusion detection framework has been designed using the ensemble based deep learning algorithm for the SDN based Cloud environment. This proposed model performs both binary as well as multi-label classification. The feature extraction with a decision tree provides accurate feature extraction. It reduces overfitting and is a flexible approach to feature reduction. The clustering process makes the interpretation easier than other approaches. By implementing the clustering process, the computational complexity of the DL algorithm got reduced by neglecting the hybrid DL algorithm. With the proposed ensemble approach, higher prediction accuracy is obtained by handling different

models. By using the deep learning approaches, the identical features are correlated and combined for an efficient learning process. The features are automatically learned from the data, and thus, the processing efficiency is increased. The performance of the ensemble based method achieves a higher detection rate of around 99.8% approximately. By deploying the clustering process, the training process also can be reuced to some extent. In future work, the performance is enhanced with the Quality-of-Service requirement.

REFERENCES

[1] B. Hajimirzaei, N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm." *Ict Express* pp. 56-595, no. 1, 2019.

[2] J. Fontaine, C. Kappler, A. Shahid, E. D. Poorter, "Log-based intrusion detection for cloud web applications using machine learning." In Advances on P2P, Parallel, Grid, Cloud and Internet Computing: Proceedings of the 14th International Conference on P2P, Parallel, Grid, *Cloud and Internet Computing (3PGCIC-2019)*, Springer International Publishing, pp. 197-210, vol.14, 2020.

[3] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions." Computer Communications pp. 2022.

[4] Z. Zhang, J. Wen, J. Zhang, X. Cai, L. Xie, "A many objective-based feature selection model for anomaly detection in cloud environment." *IEEE Access* pp. 60218-60231, vol. 8, 2020.

[5] A. Aldribi, I. Traoré, B. Moa, O. Nwamuo, "Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking." *Computers & Security* pp. 101646, vol. 88, 2020.

[6] S. Dey, Q. Ye, S. Sampalli, "A machine learning based intrusion detection scheme for data fusion in mobile clouds involving heterogeneous client networks." *Information Fusion* pp. 205-215, vol. 49, 2019.

[7] I. H. Abdulqadder, S. Zhou, D. Zou, I.T. Aziz, S. M. A. Akber, "Multi-layered intrusion detection and prevention in the SDN/NFV enabled cloud of 5G networks using AI-based defense mechanisms." *Computer Networks* pp. 107364, vol. 179, 2020.

[8] G. Jakka, I. M. Alsmadi, "Ensemble Models for Intrusion Detection SystemClassification." *International Journal of Smart Sensor and Adhoc Network* pp. 8,vol. 3, no. 2, 2022.

[9] V. Kanimozhi, T. Prem Jacob, "Artificial Intelligence outflanks all other machine learning classifiers in Network Intrusion Detection System on the realistic cyber dataset CSE-CIC-IDS2018 using cloud computing." *ICT Express* pp. 366-370, vol. 7, no. 3, 2021.

[10] [10] S. Ranjithkumar, S. C. Pandian, "Fuzzy Based Latent Dirichlet Allocation for Intrusion Detection in Cloud Using ML." *CMC-Computers Materials & Continua* pp. 4261-4277, vol. 70, no. 3, 2022.

[11] [11] K. Venkatachalam, P. Prabu, B. S. Balaji, B.G. Kang, Y. Nam, M. Abouhawwash, "Cross-layer hidden Markov analysis for intrusion detection." *CMC-Computers, Materials & Continua, Researchgate.net*pp. 3685-3700, vol. 70, 2021.

[12] J. Wei, C. Long, J. Li, J. Zhao, "An intrusion detection algorithm based on bag representation with ensemble support vector machine in cloud computing." *Concurrency and Computation: Practice and Experience* pp. e5922, vol. 32, no. 24, 2020.

[13] A. Meryem, Bouabid E.L. Ouahidi, "Hybrid intrusion detection system using machine learning." *Network Security*pp. 8-19, vol. 2020, no. 5, 2020.

[14] V. Prabhakaran, A. Kulandasamy, "Integration of recurrent convolutional neural network and optimal encryption scheme for intrusion detection with secure data storage in the cloud." *Computational Intelligence* pp. 344-370, vol. 37, no. 1, 2021.

[15] Y. Wang, W. Meng, W. Li, Z. Liu, Y. Liu, H. Xue, "Adaptive machine learning-based alarm reduction via edge computing for distributed intrusion detection systems." *Concurrency and Computation: Practice and Experience* pp. e5101, vol. 31, no. 19, 2019.

[16] G. S. Kushwah, V. Ranga, "Voting extreme learning machine based distributed denial of service attack detection in cloud computing." *Journal of Information Security and Applications*pp. 102532, vol. 53, 2020.

[17] R. M. Yadav, "Effective analysis of malware detection in cloud computing." *Computers & Security* pp. 14-21, vol. 83, 2019.

[18] E. Mugabo, Q.Y. Zhang, "Intrusion Detection Method Based on Support Vector Machine and Information Gain for Mobile Cloud Computing." *Int. J. Netw. Secur. Academia.edu*pp. 231-241, vol. 22, no. 2, 2020.

[19] Z. Chkirbene, A. Erbad, R. Hamila, A. Mohamed, M. Guizani, M. Hamdi, "TIDCS: A dynamic intrusion detection and classification system based feature selection." IEEE Access pp. 95864-95877, vol. 8, 2020.

[20] J. Brugman, M. Khan, S. Kasera, M. Parvania, "Cloud based intrusion detection and prevention system for industrial control systems using software defined networking." *In 2019 Resilience Week (RWS)* pp. 98-104, vol. 1, 2019.

[21] L. Karuppusamy, J. Ravi, M. Dabbu, S. Lakshmanan, "Chronological salp swarm algorithm based deep belief network for intrusion detection in cloud using fuzzy entropy." *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, Wiley online library*pp. e2948, vol. 35, no. 1, 2022.

[22] M. Idhammad, K. Afdel, M. Belouch, Distributed intrusion detection system for cloud environments based on data mining techniques. *Procedia Computer Science*pp. 35-41, vol. 127, 2018.

[23] A. N. Jaber, S. U. Rehman, FCM–SVM based intrusion detection system for cloud computing environment. *Cluster Computing*pp. 3221-3231, vol. 23, no. 4, 2020.

[24] S. Krishnaveni, S. Sivamohan, S. S. Sridhar, S. Prabakaran, "Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing." *Cluster Computing* pp. 1761-1779, vol. 24, no. 3, 2021.

[25] [25] T.G. Nguyen, T. V. Phan, B. T. Nguyen, C.So-In, Z. A. Baig, S. Sanguanpong, Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks. *IEEE access*pp. 107678-107694, vol. 7, 2019.

[26] Thilagam, T., and R. Aruna. "Intrusion detection for network based cloud computing by custom RC-NN and optimization." *ICT Express*pp. 512-520, 7, no. 4, 2021.

[27] S. Rajagopal, P. P. Kundapur, K. S. Hareesha, "Towards effective network intrusion detection: from concept to creation on Azure cloud." *IEEE Access* pp. 19723-19742, vol. 9, 2021.

[28] A. Abusitta, "A deep learning approach for proactive multi-cloud cooperative intrusion detection system." *Future Generation Computer Systems*pp. 308-318, vol. 98, 2019.

[29] Aldallal, Ammar, and Faisal Alisa. "Effective intrusion detection system to secure data in cloud using machine learning." *Symmetry* pp. 2306, vol. 13, no. 12, 2021.

[30] M. Mayuranathan, M. Murugan, V. Dhanakoti, "Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment." *Journal of Ambient Intelligence and Humanized Computing*, pp. 3609-3619, vol. 12, no. 3, 2021.