Received September 19, 2020, accepted September 24, 2020, date of publication October 5, 2020, date of current version October 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028690

Hyperband Tuned Deep Neural Network With Well Posed Stacked Sparse AutoEncoder for Detection of DDoS Attacks in Cloud

AANSHI BHARDWAJ, VEENU MANGAT, (Member, IEEE), AND RENU VIG
University Institute of Engineering and Technology, Panjab University, Chandigarh 160014, India

Corresponding author: Veenu Mangat (vmangat@pu.ac.in)

This work was supported by the Technical Education Quality Improvement Project III (TEQIP III) of the Ministry of Human Resource Development (MHRD), Government of India, assisted by the World Bank (Sanctioned to UIET, Panjab University, Chandigarh, India) under Grant P154523.

ABSTRACT Cloud computing has very attractive features like elastic, on demand and fully managed computer system resources and services. However, due to its distributed and dynamic nature as well as vulnerabilities in virtualization implementation, the cloud environment is prone to various cyber-attacks and security issues related to cloud model. Some of them are inability to access data coming to and from cloud service, theft and misuse of data hosted, no control over sensitive data access, advance threats like malware injection attack, wrapping attacks, virtual machine escape, distributed denial of service attack (DDoS) etc. DDoS is one of the notorious attack. Despite a number of available potential solutions for the detection of DDoS attacks, the increasing frequency and potency of recent attacks and the constantly evolving attack vectors, necessitate the development of improved detection approaches. This article proposes a novel architecture that combines a well posed stacked sparse AutoEncoder (AE) for feature learning with a Deep Neural Network (DNN) for classification of network traffic into benign traffic and DDoS attack traffic. AE and DNN are optimized for detection of DDoS attacks by tuning the parameters using appropriately designed techniques. The improvements suggested in this article lead to low reconstruction error, prevent exploding and vanishing gradients, and lead to smaller network which avoids overfitting. A comparative analysis of the proposed approach with ten state-of-the-art approaches using performance metrics-detection accuracy, precision, recall and F1-Score, has been conducted. Experiments have been performed on CICIDS2017 and NSL-KDD standard datasets for validation. Proposed approach outperforms existing approaches over the NSL-KDD dataset and yields competitive results over the CICIDS2017 dataset.

INDEX TERMS Machine learning, intrusion detection, artificial neural network, cloud computing, distributed denial of service attack.

I. INTRODUCTION

Cloud Computing has very attractive features like elastic, on demand and fully managed computer system resources and services which makes it well suited for application in sectors such as healthcare, manufacturing, retail, entertainment, etc. Due to its distributed and dynamic nature as well as vulnerabilities in virtualization implementation, the cloud environment is prone to various cyber attacks. One of these attacks which is very perilous is Distributed Denial of Service (DDoS) attacks. DDoS is a deadly weapon which overwhelms the server or network by sending floods of packets towards it. The attack disrupts the services running on the target, thereby blocking the legitimate traffic accessing its services. The implications of DDoS attacks can be direct financial and business loss and indirect reputation loss for cloud providers as well as customers. Particularly in the cloud scenario, it can lead to Economic Denial of Service attacks [1]. Thus, there is a compelling need for timely and accurate detection of these attacks.

The motivation for undertaking this problem is twofold. Firstly, recent statistics reveal that the potency and frequency of DDoS attacks are increasing at an alarming rate. In February 2020 [2], Amazon Web Services (AWS) customers suffered from a critical breakdown when a DDoS attack targeted Amazon's Simple Service Storage (S3) and other services,

The associate editor coordinating the review of this manuscript and approving it for publication was Jonghoon Kim.

thereby taking them down for almost eight hours. It is one of the largest DDoS attacks witnessed till now with a capacity of 2.3 Tbps. The attack brought down DNS web service router which had adverse effect on other services also like Elastic Load Balancing (ELB), Relational Database Service (RDS) and Elastic Compute Cloud (EC2), which are used to lookup public DNS resolution system. Secondly, attack vectors are evolving constantly. For instance, attackers have recently introduced a new DDoS amplification protocol which has already increased the attack capacity to reach upto 350 Gbps. The attackers exploit the multicast protocol web services dynamic discovery protocol for amplifying DDoS attacks by spoofing the return IP address. Wikipedia, the world's largest online encyclopedia was rendered inaccessible in various parts of Europe, Africa, and the Middle East on 6th September 2019 [3] by a long duration high capacity DDoS attack. According to a Security Week article, researchers have found out that the average number of distinct DDoS attacks that hit the Internet per day is about 28,700 [4].

Authors in [5] discussed National Cyber Security Strategies (NCSS) of various countries like India, New Zealand, Australia etc. They also discussed challenges in standardization of cyber security. The challenges being faced are in the areas of standardization, lack of agility, economic considerations, competing set of standards, lack of awareness. Various solutions based on packet-level analyses, flow-level analyses, behavioral analyses, traffic mining and deep packet inspection of network traffic, have been proposed by researchers for combating DDoS attacks [6]–[11]. Recent advances in machine learning and deep learning techniques have also been employed for detection of DDoS attacks [12]–[14].

A detailed study of all these methods reveals that their effectiveness is limited due to some challenges. Firstly, there is a lack of training data for DDoS attacks. Organizations are often unwilling to publicly acknowledge having been attacked and do not share network attack data, for fear of loss of reputation. Moreover, the datasets that are available, are imbalanced wherein distribution of different classes i.e. attacks traffic and legitimate traffic is skewed. This leads to overfitting of the data and results in undesirable and misleading accuracy. The second major challenge is the selection of optimal features of the network traffic. Though deep learning methods are capable of handling a large number of features in the input, feeding too many features to the algorithm confounds the classification problem and can even lead to overfitting. The presence of too many irrelevant input features also slows down the learning. Since attack vectors are constantly evolving, when a new attack vector is launched, the system may not be able to detect it as attack traffic. Therefore, some efficient way of feature representation is required to learn the complex often non-linear representations from data. Additionally, it has been estimated that 1.7MB of data every second will be generated by every person in 2020. According to the report by Domo [15], everyday more than 2.5 quintillion bytes of data are generated. In order to deal with this huge volume of data, the detection or classification model should be compact. Apart from high dimensionality, the other major challenges are presence of noise in the data and computational cost for training over large datasets.

This article proposes an architecture employing a well posed AutoEncoder (AE) with Deep Neural Network (DNN) to deal with the challenges of effective feature learning, handling noisy data and preventing overfitting. The main contribution of this work is as follows:

1. It proposes a novel DNN architecture that uses a well posed stacked sparse AE tailored for learning informative feature representation from network traffic.
2. AE and DNN are optimized for detection of DDoS attacks by tuning the parameters using appropriately designed techniques.
3. Comparison of proposed architecture with ten other state-of-the-art machine learning approaches has been presented over NSL-KDD and CICIDS2017 standard datasets.
4. Proposed architecture is found to either outperform or yield competitive results in terms of performance metrics-accuracy, precision, recall and F1-Score.

The rest of the paper is organized as follows. Section 2 discusses the recent work in literature related to approaches for detection of DDoS attacks. Section 3 describes the proposed approach based on optimized AE and DNN model. Experimentation and validation details are described in Section 4. Results and discussion are presented in Section 5. Finally, Section 6 concludes the paper and presents future research directions.

II. RELATED WORK

This section discusses work related to the detection of DDoS attacks from network traffic. Specifically, those works have been mentioned which are aligned with the focus of current study and have employed machine learning or deep learning either in feature reduction or hyperparameter optimization or classification stage, over the same standard datasets viz. CICIDS2017 [16] and NSL-KDD [17]. Since a considerable amount of literature exists in the broad area of network intrusion detection, this inclusion criteria has been adopted for selection of most recent and highly relevant papers for this study. Interested readers can refer to surveys on broad area of detection of DDoS attacks available in [12]–[14].

NIDS using Sparse AE (SAE) and SoftMax Regression (SMR) has been proposed by Javaid *et al.* for classification of attacks [18]. The data is first preprocessed using 1-n encoding and also normalized in range [0,1] using min-max normalization. The preprocessed data is then fed to SAE for self taught learning. The learned features are then given to SMR for classification of attacks. The method reported 88.39% accuracy on NSL-KDD dataset. An ensemble deep learning model is presented which comprises AE, Deep Belief Network (DBN), DNN, and Extreme learning methods (ELMs) and validated over NSL-KDD dataset with detection rate of 97.95%, and false alarm rate of 14.72% [19].

The ensemble method performed better for detection rate and false rate but achieved lower detection accuracy compared to other methods.

Yousefi *et al.* proposed unsupervised feature learning method using AE for extracting latent feature set from whole dataset [20]. The dataset used is Microsoft Malware Classification Challenge which has been hosted by Kaggle particularly for AE based representations. The proposed method has two training stages i.e. pre-training and fine tuning stage. In pre-training stage, the optimum weights and AE parameters are searched from parameter space and then fed to AE for feature reduction. The reduced feature set is given to Gaussian Naive Bayes (NB) for classification. The method shows better performance compared to K-Nearest Neighbour (KNN), Support Vector Machine (SVM) and Gradient Boosting. A Recurrent Neural Network (RNN) based Intrusion Detection System (IDS) is proposed for binary and multi-class classification by authors in [21]. First in preprocessing, numericalization i.e. conversion of non-numeric features to numeric features, and then normalization i.e. converting the data into range [0,1] for efficient classification are done. The pre-processed data is then fed to RNN. Experimentation on NSL-KDD has shown improved accuracy and lower false positive rate.

Yusof *et al.* provided adaptive feature selection method for detection of DDoS attacks [22]. DDoS Characteristic based Features (DCF) and Consistency-based Subset Evaluation (CSE) are used for selecting the features. Then simple majority voting technique is used for selecting the most suitable methods. Then these selected features are given to Multi-layer perceptron (MLP) for classification of attacks. NSL-KDD dataset has been used and 91.7% accuracy is reported by the proposed method. Authors proposed self-taught learning (STL) framework based on SAE and SVM for binary and multi-class classification [23]. SAE provides reduced significant feature set using unsupervised method. This reduced feature set is given to SVM for efficient classification. The results on NSL-KDD have shown improved detection accuracy and reduced SVM training and testing time as compared to shallow techniques like NB, Random Forest (RF), J48 and SVM. The model reported 84.96% accuracy.

Distributed machine learning based IDS has been proposed by Idhammad *et al.* for cloud environment [24]. The pre-processed data is used by anomaly detection module which uses NB classifier for separating the network traffic data into normal or abnormal traffic. Ensemble learning classifier based on RF performs a multi-class classification. Experimentation was performed on Google platform using CIDDS-001 dataset and showed better performance than standard RF. Authors in [25] proposed an IDS which combines Cuckoo Optimization Feature Selection (COFS) and Naïve Bayes Algorithm (NBA). COFS removes the redundant and irrelevant features, the processed features are passed to NBA which performs the classification part. The proposed method gives better accuracy than the existing feature selection methods like information gain-based feature selection (IGFS), chi-square feature selection (CSFS), information gain ratio-based feature selection (GRFS) and One R feature selection (ORFS). Authors [26] used Nonsymmetric Deep AE (NDAE) (deep approach) and RF (shallow approach) for classification of intrusions for obtaining high quality results. Two NDAEs were stacked using three hidden layers and the encoded representation output was then fed to RF for classification. RF properties like low bias, overfitting correction and robustness to outliers are very useful for this method. The results have been evaluated against previous work on benchmark datasets and shown 5% increase in accuracy and training time scaled down by 98.81%.

In study [27], authors gave an intelligent scheme based on DNN using hybrid optimization system. The hybrid optimization consists of Improved Genetic Algorithm (IGA) and Simulated Annealing Algorithm (SAA). The GA is improved by optimizing the fitness function using parallel processing and fitness value hashing strategies. Experimentation was performed using CloudSim 4.0 simulator and showed high detection rate of 99.95% and 0.05% false positive rate. Authors in [28] combined Binary Bat algorithm with RF for classification of intrusions in the network. Two new fitness functions - Similarity-based Fitness Function (FSFF) and Classifier Accuracy based Fitness Function (CAFF) are used in Bat algorithm for selecting the optimal features. Experiments performed on UNSWNB15 and CICIDS-2017 datasets showed highest accuracy of 97.09% and false positive rate of 2.03%.

Authors used Particle Swarm Optimization (PSO) for optimal selection of hyperparameters in the pretraining phase [29]. Four deep neural models i.e. DNNs, Long Short-Term Memory RNN (LSTM-RNN), gated recurrent unit RNNs (GRU-RNNs), and DBNs have been used on the datasets KDD CUP 99, NSLKDD, CIDDS, and CICIDS2017. The approach has shown significant improvement over shallow learning models. A powerful IDS which is a combination of three models- MLP network, Artificial Bee Colony (ABC) and Fuzzy C-means (FCM) clustering algorithm been proposed by Hajimirzaei and Navimipour [30]. FCM clustering prepares homogeneous sub-sets of training data. MLP with back propagation classifies the intrusions and normal traffic. ABC helps MLP to determine optimal weights and bias values for efficient classification. Root mean square error, kappa statistic and mean absolute error are used for comparing the proposed method with other state-of-the-art methods.

Authors proposed packet level classification using word embedding and LSTM model [31]. The approach uses word embedding scheme to extract word semantics and syntax of header fields and then with the support of LSTM it extracts the temporal relations among the packets and classifies the attack and non-attack packets. The primary advantage of using packet level classification is that the detection process speed boosts up as significant time is reduced in flow processing. Experimentation shows that the proposed approach can attain nearly 100% accuracy in detecting attack and non-attack packets. Authors in [32] proposed LSTM based scheme for detection of attacks. The LSTM approach does not require feature engineering process instead it automatically learns complex spatial and temporal relationships from the dataset. It can learn relationships from a small subset of data. The proposed architecture of LSTM consists of four layers which are 2 LSTM layers, 1 drop out layer and 1 connected layer. Experiments have shown that the LSTM based scheme can detect unknown attacks and outperforms other machine learning methods like DT, SVM and Artificial Neural Network (ANN).

Authors proposed Snort intrusion detection and deep learning model for detection of DDoS attacks in Software Defined Network (SDN) [33]. Sampling techniques-sFlow and adaptive polling sampling have been applied to reduce the processing and network head of switches. DBN network has been used in the classification of intrusions. Experimentation performed on NSL-KDD has shown better accuracy than SVM and back propagation neural network. The speed of detection has been increased and detection accuracy was reported as 95.25%. The authors in [34] proposed double PSO based method for the detection of attacks. The proposed method consists of four stages: data pre-processing, pre-training, training and testing. In data pre-processing, the non-numerical values are transformed to numerical values using hot-encoding method, also the values are normalized using min-max normalization. The next stage of pre-training involves double PSO optimization algorithm which is a hierarchical multi-purpose algorithm which has two levels. First level uses filterbased feature selection algorithm using entropy (FPSBPSO-E) which finds the best feature subset for optimal accuracy and the second level uses PSO for hyperparameter selection. DNN, LSTM-RNN, and DBN are trained and tested on the generated results of the above two stages. NSL-KDD and CICIDS have been used for validating the results and showed high accuracy, precision, recall, and F1-Score.

SVM and AE based combined method is proposed by authors in [35]. AE was trained with optimal hyperparameters which helps in dimensionality reduction and feature extraction. The best feature set is further fed to SVM for classification of DDoS attacks. Experiments were performed on CICIDS2017, NSL-KDD and virtually generated dataset. Results have shown that the proposed approach is better results than pure Bayesian, RF, SVM and J48. The proposed approach showed low false positive detection rate due to feature extraction and trained hyperparameters.

Wang *et al.* [36] proposed an MLP based model for feature selection and Sequential Backward Selection (SBS) which is a wrapper feature selection method. The method was tested on the NSL-KDD dataset and showed that using feedback mechanism the proposed method can effectively perceive the detection errors. The accuracy achieved is 97.66% with false alarm rate of 0.62%. Authors employed unsupervised deep learning methods AE and Variational AE (VAE) along with One Class SVM (OCSVM) for detection of both known and unknown attacks [37]. The proposed AE and VAE have 2 encoding and 2 decoding layers with bottleneck layer having 64 neurons. The hyperparameter tuning was done using the recommendation given by [38]. Experimentation performed on CICIDS2017 dataset shows better performance in terms of AUC and ROC curves.

Authors in [39] provided an approach called SAVAER that uses supervised VAE and the merits of Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) rather than plain GAN. VAE helps to obtain the latent representation of attack set. The output from VAE is fed to DNN and the trained VAE weights determines the weights of hidden layer of DNN. The combined SAVER and DNN approach can detect known and unknown low rate frequent attacks also. Experimentation on UNSW-NB dataset has shown that the proposed method has outperformed the state-of-the-art models in terms of accuracy, detection rate and F1-score.

III. PROPOSED METHODOLOGY

A. DATASETS

1) NSL-KDD

NSL-KDD dataset is updated, cleaned and revised version of KDD'99 dataset of university of New Brunswick [17]. The KDD cup was an International Knowledge Discovery and Data Mining Tools Competition in 1999. The challenge was to build an IDS to distinguish between good and bad connections. In order to achieve this, large amount of network traffic was collected and gathered together to form KDD dataset. The NSL-KDD dataset has been generated from the KDD dataset. NSL-KDD dataset consists of 43 features. Among these, 41 features correspond to input traffic features and last two features are the class label (i.e. whether an attack or normal) and score (defines severity of attack) respectively. These are shown in Table 1.

There are 37 different attacks which fall under four categories viz. DoS, probe, U2R and R2L. These attack categories are shown in Table 2.

2) CICIDS2017

The second dataset that has been chosen for our experiments is the CICIDS dataset. The dataset is appropriate for our research problem as it includes up to date attacks and the features are complete. This is in comparison with other network intrusion datasets like UNSW-NB15 [40], CAIDA [41], AWID [42], DARPA [43], CIDDS-001 [44], which are either incomplete or noisy. The dataset contains benign and malicious traces of network traffic. It is a labelled dataset with a total of 84 features. The last feature is the class label which identifies the sample as attack or benign traffic. The features have been extracted by CICFlowMeter-V3. The output of CICFlowMeter-V3 is a CSV file that includes: Flow ID (1), Source IP (2) and Destination IP (4), Source Port (3) and Destination Port (5), Protocol (6), Timestamp (7) and Label (84). The above mentioned features are the basic features and

TABLE 1. Features in NSL-KDD Dataset.

SNo.	Feature	SNo.	Feature
1	Duration	23	Count
2	protocol_type	24	srv_count
3	service	25	serror$_{r}ate$
4	flag	26	srv_serror_rate
5	src_bytes	27	rerror_rate
6	dst_bytes	28	srv_rerror_rate
7	land	29	same_srv_rate
8	wrong$_{f}ragment$	30	diff_srv_rate
9	urgent	31	srv_diff_host_rate
10	hot	32	dst_host_count
11	num_failed_logins	33	dst_host_srv_count
12	logged_in	34	dst_host_same_srv_rate
13	num_compromised	35	dst_host_diff_srv_rate
14	root_shell	36	dst_host_same_src_port_rate
15	su_attempted	37	dst_host_srv_diff_host_rate
16	num_root	38	dst_host_serror_rate
17	num_file_creations	39	dst_host_srv_serror_rate
18	num_shells	40	dst_host_rerror_rate
19	num_access_files	41	dst_host_srv_rerror_rate
20	num_outbound_cmds	42	Label
21	is_host_login	43	Score
22	is_guest_login		

TABLE 2. Attack categories in NSL-KDD Dataset.

DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_password, Ftp_write, Imap, Phf,Multi hop, Warezmaster, Xlock, Xsnoop, Snmpgue ss, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

TABLE 3. CICIDS Features.

SNo.	Feature	SNo.	Feature	SNo.	Feature	SNo.	Feature
1	Flow ID	22	Flow Packets/s	43	Fwd Packets/s	64	Fwd Avg Bulk Rate
2	Source IP	23	Flow IAT Mean	44	Bwd Packets/s	65	Bwd Avg Bytes/Bulk
3	Source Port	24	Flow IAT Std	45	Min Packet Length	66	Bwd Avg Packets
4	Destination IP	25	Flow IAT Max	46	Max Packet Length	67	Bwd Avg Bulk Rate
5	Destination Port	26	Flow IAT Min	47	Packet Length Mean	68	Subflow Fwd Packets
6	Protocol	27	Fwd IAT Total	48	Packet Length Std	69	Subflow Fwd Bytes
7	Time stamp	28	Fwd IAT Mean	49	Packet Len. Variance	70	Subflow Bwd Packets
8	Flow Duration	29	Fwd IAT Std	50	FIN Flag Count	71	Subflow Bwd Bytes
9	Total Fwd Packets	30	Fwd IAT Max	51	SYN Flag Count	72	Init_Win_bytes_fwd
10	Total Backward Packets	31	Fwd IAT Min	52	RST Flag Count	73	Act_data_pkt_fwd
11	Total Length of Fwd Pck	32	Bwd IAT Total	53	PSH Flag Count	74	Min_seg_size_fwd
12	Total Length of Bwd Pck	33	Bwd IAT Mean	54	ACK Flag Count	75	Active Mean
13	Fwd Packet Length Max	34	Bwd IAT Std	55	URG Flag Count	76	Active Std
14	Fwd Packet Length Min	35	Bwd IAT Max	56	CWE Flag Count	77	Active Max
15	Fwd Pck Length Mean	36	Bwd IAT Min	57	ECE Flag Count	78	Active Min
16	Fwd Packet Length Std	37	Fwd PSH Flags	58	Down/up ratio	79	Idle Mean
17	Bwd Packet Length Max	38	Bwd PSH Flags	59	Average Packet Size	80	Idle Packet
18	Bwd Packet Length Min	39	Fwd URG Flags	60	Avg Fwd Segment Size	81	Idle Std
19	Bwd Packet Length Mean	40	Bwd URG Flags	61	Avg Bwd Segment Size	82	Idle Max
20	Bwd Packet Length Std	41	Fwd Header Length	62	Fwd Avg Bytes/Bulk	83	Idle Min
21	Flow Bytes/s	42	Bwd Header Length	63	Fwd Avg Packets/Bulk	84	Label

the features from number 8 to 83 are high level features which are statistically computed from the low level features. The features are as shown in Table 3.

This dataset has 11 characteristic features which makes it an authentic IDS dataset. These crucial characteristics are as given below:

Anonymity: Many datasets remove the payload information due to privacy concerns. This removal hampers the detection methods. But in CICIDS dataset the payload remains intact.

Attack Diversity: DDoS attack vectors are changing rapidly. This dataset includes all the recent attacks which are brute force, DoS, browser-based, DNS based attacks, port scan or enumeration, backdoors and other attacks like Heartbleed, Apple SSL library bug, Shellshock.

Available Protocols: It provides all necessary protocols like HTTP, HTTPS, FTP, SSH and email protocols.

Complete Capture: Traffic consists of packets which have been captured in the dataset from source to router, switch, host, multicast group or broadcast domain.

Complete Interaction: The dataset includes all network interactions between the two networks i.e. victim network and attack network. This also includes interactions between the internal LAN which makes it a valuable dataset.

Complete Network Configuration: It includes numerous modem, firewall, switches, routers, and number of PCs with variety of operating systems such as Windows, Ubuntu and Macintosh to have realistic configuration for capturing the real attack traces.

Complete Traffic: It includes all the necessary traffic by using user profiling agent and 12 different machines in Victim-Network and real attacks from the Attack-Network.

Feature Set: By using feature extraction applications it extracts more than 80 network flow features from the generated network traffic and delivers the network flow dataset as a CSV file.

Metadata: It includes proper documentation about the network configuration, operating systems for attacker and victim machines, attack scenarios and useful information about the dataset.

Heterogeneity: It captures the heterogeneous network traffic from network equipment, memory dump and system call from all victim machines during the attacks execution. This helps in development of robust detection mechanism.

Labelling: It is labelled and informative for reliable and accurate analysis.

B. DATA PREPROCESSING

The first step is to preprocess the input data in order to improve its quality and subsequently the quality of the output and efficiency of the mining algorithm. The three preprocessing operations that have been applied are: label encoding, removal of irrelevant features and normalization.

1. **Label Encoding:** One hot encoding is used to convert categorical features into numerals in NSL-KDD dataset [45]. In NSL-KDD there are three categorical features: protocol type, service and flag which are converted to numeric with one hot encoding. In CICIDS2017, FlowID has been converted using hash encoding due to large number of unique values in FlowID column. Hash encoding transforms the column categorical values using hash function. It converts the value using new dimension which is fixed using n_component argument. The number of components used is 18 which can represent 2^{18} i.e. 262144 unique values. The number of components can be increased to accommodate the increase in the unique values.
2. **Removal of irrelevant features:** The values of attributes which are non informative or invalid such as have NaN and infinity are removed for efficient running of algorithms.
3. **Normalization:** Each feature of the dataset has different maximum and minimum values. The efficiency of the classifier is increased if all the values are normalized in the range [0,1] [46]. Min-max normalization is used for converting the attribute values to fall in the range [0.1]. Min-max scaling is done using the formula given below in Equation (1):

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

C. NAIVE AE AND DNN ARCHITECTURE

1) OVERVIEW OF AE

AE is a semi-supervised machine learning technique to effectively learn data representations by training the network [47]. It is useful for feature leaning and dimensionality reduction for achieving non-linear generalizations. It has one input layer, one or more hidden layers for encoding and one output layer for decoding. Suppose there is a sample of unlabelled training data X_1, X_2, X_3... where $X_i \in R_n$.

AE model uses some back propagation algorithm to always set output values to input values i.e. $Y_i = X_i$. AE architecture has two important operations which are encoding and decoding. The encoder function h = f(X) maps the input data X into latent space or reduced representation h through some transformation with bottleneck or restriction applied that defines possible representations the network can compute. The decoder function is reconstruction function R = g(h) which reconstructs the input with reduced features or improved generalizations among the features. The input high dimensional data vector X is encoded into latent or low dimensional representation h with standard neural network function using weight W, bias b and activation function σ as given below in Equation (2).

$$h = \sigma(WX + b) \tag{2}$$

The decoder function reconstructs the output from latent representation h with different weight W', bias b' and activation function λ as given below in Equation (3).

$$R = \lambda(W'h + b') \tag{3}$$

The basic AE architecture is shown in Figure 1.

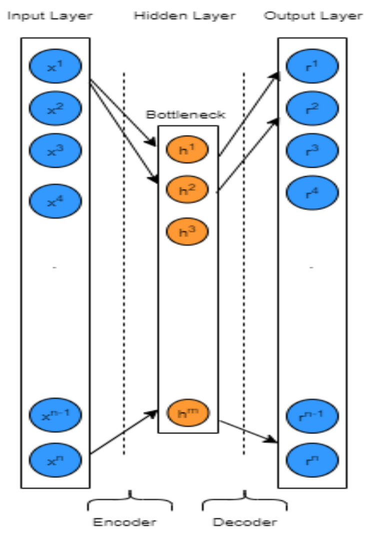


FIGURE 1. Basic AE architecture.

The loss function in AE is used to minimize the difference between input and output values. The loss function mainly used is either mean square error (MSE) or cross entropy. The MSE for input vector X and output R of length n is defined in Equation (4):

$$d(X, R) = \frac{1}{n}\sum_{i=1}^{n}(X - R)^2 \tag{4}$$

Similarly, the binary cross entropy is defined in Equation (5):

$$d(X, R) = -(X.log(R) + (1 - X).log(1 - R)) \tag{5}$$

where '·' represents element wise product and all other operations are computed element wise only. The hyperparameters which are to be set before AE training are number of layers, nodes per layer, code size and loss function. These are shown in Table 4.

TABLE 4. AE Hyperparameters.

Hyperparameter	Meaning
Number of layers	Can be as deep as we require.
Nodes per layer	Number of nodes decrease per layer with every successive encoder layer and increases with every decoder layer. Decoder layer is symmetric to encoder layer.
Code size	Count of nodes in middle layer. Smaller the code size, more is the compression.
Loss function	Binary cross entropy is used when values are in range 0,1 otherwise use MSE.

2) OVERVIEW OF DNN

DNN is simply a representation of the ANN except it has numerous deep hidden layers [48]. Principally, DNN consists of an input layer, one or more hidden layers, and an output layer. Every layer in DNN consists of one or more artificial neurons or nodes in such a way that theses neurons are fully-connected from layer to layer. The information is processed and propagated through DNN in feed-forward manner, i.e., from the input layer to the output layer via the hidden layers. The basic architecture of DNN is shown in Figure 2.

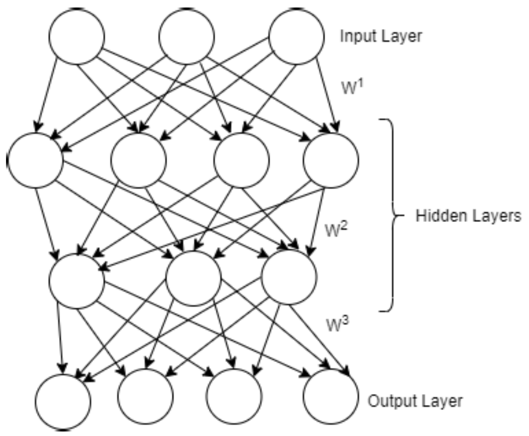


FIGURE 2. Basic DNN architecture.

In our proposed method, we have first constructed a naive baseline model by using AE for feature extraction and DNN for classification of DDoS attacks. We have used two standard network intrusion detection datasets viz. NSL-KDD and CICIDS2017. NSL-KDD dataset has four different types of attacks which are DoS, Probe, R2L and U2R. There are further 37 different attack types in each category. So, for detection of DDoS attacks we have extracted data from the dataset corresponding to DDoS attacks-Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2 and Worm. After extraction, KDDTrain and KDDTest consists of 148517 record data and 43 feature columns. The three columns-protocol type, service and flag, which have categorical values are converted to numeric values with help of one-hot encoding. In this process, 43 feature columns are mapped to 124 feature columns.

Min-max scaler is applied to normalize the values in the range [0,1] and all irrelevant values are also removed. Then AE is trained with a sample taken from input training dataset instead of the entire dataset to prevent overfitting. Both random and systematic sampling were tried during experimentation. Systematic sampling gave better results suggesting a periodicity in data. But to reduce generalisation error, input data used for training the AE is a random sample of 25000 records. The output of AE is the encoded data having 40 features. This encoded data are fed to DNN for classification of attack and non-attack traffic. The parameters used to execute AE are listed in Table 5.

TABLE 5. Naive AE Model Hyperparameters for NSL-KDD.

Model parameters	Values
Input Neurons	123
Output Neurons	123
Number of encode layers	8
Number of decode layers	8
Number of Neurons in code layer	40
Optimizer	Adadelta
Loss function	Binary cross entropy
Epochs	10
Batch size	2500

TABLE 6. Naive DNN Model Hyperparameters for NSL-KDD.

Model parameters	Values
Hidden Layers	9
Number of neurons to each subsequent layer	38, 39, 34, 42. 40, 35, 38, 28,32
Activation functions	ReLU, Sigmoid
Learning rate	0.01
Number of Neurons in code layer	40
Decay rate	0.9
Decay step	0.9
Batch size	20000
Optimizer	ADAM, lr-schedule
Loss function	Binary cross entropy

After the encoded features have been obtained, they are fed to DNN for classification. The dataset is split into 75% training set and 25% testing set. The 75% training set with 40 input features are fed to DNN. The parameters used to execute DNN are listed in Table 6.

The complete methodology of naive (baseline) model for NSL-KDD dataset is shown in Figure 3 below. In CICIDS2017 dataset, there are a total of 225720 data records and 85 feature columns. The features-Flow ID, source IP, source port, destination IP, destination port, timestamp, are removed and not fed to AE. These features are already known to be informative from literature and will not be used for feature extraction, rather they will be fed directly to DNN. A random sample of 25000 records is taken. The records which have infinity or NaN values are removed. The column values are normalized in the range [0,1] using min-max scaler method as before. This yields 24994 data records and 78 columns for training the AE. The AE encodes these 78 feature columns into 23 columns. Once the AE is trained, the entire training dataset is input to AE for generating encoded representation. This encoded output of AE is fed to DNN for classification of attack and non-attack traffic. The parameters used to execute AE are listed in Table 7.

After the encoded features are obtained, they are fed to DNN for classification. The dataset is split into 75% training set and 25% testing set. The 75% training set with input features are fed to DNN. The parameters used to execute DNN are listed in Table 8.

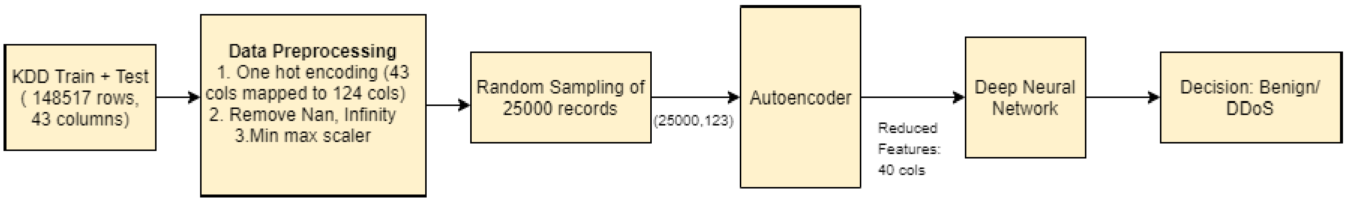

KDD Train + Test (148517 rows, 43 columns)
Data Preprocessing
1. One hot encoding (43 cols mapped to 124 cols)
2. Remove Nan, Infinity
3.Min max scaler
Random Sampling of 25000 records
(25000,123)
Autoencoder
Reduced Features: 40 cols
Deep Neural Network
Decision: Benign/ DDoS

FIGURE 3. Naive model construction for NSL-KDD.

TABLE 7. Naive AE Model Hyperparameters for CICIDS2017.

Model parameters	Values
Input Neurons	123
Output Neurons	123
Number of encode layers	6
Number of decode layers	6
Number of Neurons in code layer	20
Optimizer	Adadelta
Loss function	Binary cross entropy
Epochs	10
Batch size	2500

TABLE 8. Naive DNN Model Hyperparameters for CICIDS2017.

Model parameters	Values
Hidden Layers	9
Number of neurons to each subsequent layer	38, 39, 34, 42. 40, 35, 38, 28,32
Activation functions	ReLU, Sigmoid
Learning rate	0.01
Decay rate	0.9
Decay step	0.9
Batch size	20000
Optimizer	ADAM, lr-schedule
Loss function	Binary cross entropy

D. OPTIMIZED AE AND DNN MODEL

The main contribution of our proposed method is in the optimizations that have been done on combined AE and DNN architecture and the novel techniques that have been applied to improve the performance of the detection mechanism. The optimized AE and DNN architecture has given best results for this specific problem. This section describes the novel techniques introduced by this work and optimizations applied to generate high performance DDoS detection approach.

1) OPTIMIZED AE

In order to obtain an effective learned feature representation, this work applies the following optimizations to the AE:

- **Grid Search:** Grid Search has been applied for determining the optimal values of hyperparameters for AE. It works by automatically performing an exhaustive search on specific values for hyperparameters, thus saving time and resources. Grid Search has been used to determine best values for sparsity parameter, number of layers and neurons in each layer. The chosen values are the ones that lead to minimum reconstruction error. Table 9 below shows the list of values for these

TABLE 9. Optimized AE hyperparamters.

Hyperparameters	Values	Selected Values
Sparsity parameter	10e-5,10e-3,10e-1	10e-5
Number of layers	2,3,4,5,6,7,8	2 (encoding and decoding)
Neurons in each layer	80,70,60,50,40,30,25,20,15,10	70, 50
Coding Dimension	80,70,60,50,40,30,25,20,15,10	25
Activation Function	Relu, Sigmoid, Tanh	Relu, Sigmoid
Optimizer	Default	Adadelta

  hyperparameters from which the selection has been made and the corresponding optimum values. After multiple experiments with Grid search, we have arrived at AE model with 2 encoding layers with 70 and 50 neurons respectively and ReLU activation, 25 neurons in coding layer, and 2 decoding layers with same neurons as in coding layer and ReLU activation. The output layer has sigmoid activation.
- **Sparsity:** Activity regularization has been used to introduce sparsity and avoid overfitting. The sparsity penalty constraints the representation when added to the hidden representation activity. It applies a penalty on the layer's output on a per layer basis and weighting between the activity regularizers remains same with the batch size. The L1 regularisation has been used with regularization factor value=10e-5.

The optimizer being used is Adadelta or adaptive delta. It is an extension of Adagrad which attempts to reduce monotonically decreasing learning rate by using moving windows of gradient updates. The effect of these steps is that the small network chosen through Grid search method makes the AE well posed with no overfitting. Additionally, a stacked non-symmetric AE is used to learn higher level features through multiple encoding layers and take output of encoder directly as input of DNN.

The other additional constraints that have been imposed on AE to make it well posed are:

- **Unit norm:** The weights on each layer have a unit norm which prevents weights from becoming too large, and avoids the exploding gradient problem. Very large gradients lead to large updates to the network weights resulting in an unstable network. In proposed optimized AE, the weights are unit norm as given in Equation (6).

$$\sum_{i=1}^{k} w_{i,j}^2 = 1 \tag{6}$$

  where i = 1...k

TABLE 10. Optimized DNN hyperparameters.

Hyperparameters	Values	Selected Values
Number of neurons	30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64	20, 12
Number of layers	2, 3, 4, 5	2
Optimizer	Adam, Adabound, Adagrad, Nadam and SGD + momentum	Adabound

- **Weight orthogonality:** The weight vectors are independent of each other and only informative weights are non-zero. This prevents vanishing gradients. It also leads to lesser redundancy and compact network since each feature has unique information and the information can be encoded with a smaller encoder. The weight orthogonality is enforced using Equation (7) given below.

$$W^TW = 1 \tag{7}$$

Figure 4 shows well posed AE architecture with input, encoding, code, decoding and output layers.

2) OPTIMIZED DNN

- **Feature selection:** A robust strategy was implemented to first evaluate the performance of DNN model with a mini batch gradient descent using optimizer Adam which uses adaptive learning rates. This functioned as the baseline model described in Subsection (III-C) above. The output of AE comprising 25 important features along with additional informative features (Flow ID, source IP, source port, destination IP, destination port, timestamp) are fed to DNN for classification. Instead of random sample as fed in forming baseline model above, now the whole dataset is fed to DNN.
- **Hyperband tuning:** Hyperband tuning is a recent technique for optimizing hyperparameters of iterative algorithms. It gives optimized results by performing random sampling on the dataset. It selects the best model by training multiple models for few epochs and keeps on training until one of them gives best result. This work uses hyperband tuning for tuning number of neurons, number of layers and optimizer. Table 10 below shows the list of hyperparameters, values from which the selection has been made for the optimized values, and the selected values.
  5-fold cross validation has been used when implementing hyperparameters optimization so that the values of hyperparameters work well for validation data also. This yielded 20, 12 neurons in 2 hidden layers respectively and optimizer Adabound gave the most stable performance.
- **Adabound:** The optimizer is a variant of Adam optimizer method [49]. Adabound places dynamic limits on learning rate to obtain fast initial learning while giving good generalization performance. It uses same update equations as Adam except that it adds gradient clipping for learning rate given by Equation (8) and (9).

$$\eta = bound(\frac{\alpha}{V_t}, \eta_l(t), \eta_u(t)) \tag{8}$$

$$\eta_t = \frac{\hat{\eta}_t}{\sqrt{t}} \tag{9}$$

  where bound is clipping function, α is step size, v_t is obtained from decaying average of past squared gradients η_u and η_l are upper and lower bound function respectively, t is iteration number. The output gets constrained to be in η_u and η_l.
- **Intelligent initial learning rate determination:** Further improvements have been achieved with an intelligent strategy for determining starting learning rate, which is then decayed using a simple learning rate schedule. The main aim is to find the highest value for learning rate which minimizes the loss. To determine this value, the model is trained for 1 epoch while increasing the learning rate after each batch. The loss is recorded and the learning rate is set to the value just before loss exploded. A value of initial learning rate=0.01 was obtained. Then a learning schedule is implemented with the help of lr_schedule using exponential decay, decay rate of 0.9 and step size=10,000.

The block diagram for optimized AE and DNN model is shown in Figure 5. The pseudo-code for the algorithm is given below:

Algorithm 1 Proposed Optimized AE and DNN Detection

```
**Input:** Complete Dataset D = $(X_1, X_2 \ldots X_k)$ where $X_i \epsilon R^d$
**Output:** Prediction result (binary class label) Y
Preprocess_data (D) $\rightarrow D^{'}$
Random_sample($D^{'}$) $\rightarrow S$ s.t. $|S| = \frac{|D|}{4}$
Train_AE ($S, f_\phi, g_\theta$) $\rightarrow$ Opt_AE
Apply_AE ($D^{'}$, Opt_AE) $\rightarrow F_{AE}$
$D^{''} = D^{'}{}_{FAE} ||$ informative features
Train_DNN($D^{''}$) $\rightarrow$ Opt_DNN
Add Opt_DNN after Opt_AE to form Opt_AE+DNN
Input test data to generate class label Y
**return** Y
```

IV. EXPERIMENTATION AND VALIDATION

The experiments have been conducted using a Windows 10-64 bits PC with 16 GB RAM and CPU Intel(R) Core-i7. For simulation of cloud environment, VMware workstation has been used in which a cloud server is made with different virtual machines. The performance of the detection system is determined by how correctly the proposed system is able to classify the incoming traffic into corresponding category. To evaluate our proposed method, we have used four performance metrics-accuracy, precision, recall and F-score. The description of these metrics is given below.

Detection Accuracy (Acc): It is the proportion of correctly classified instances to the total number of classified instances.

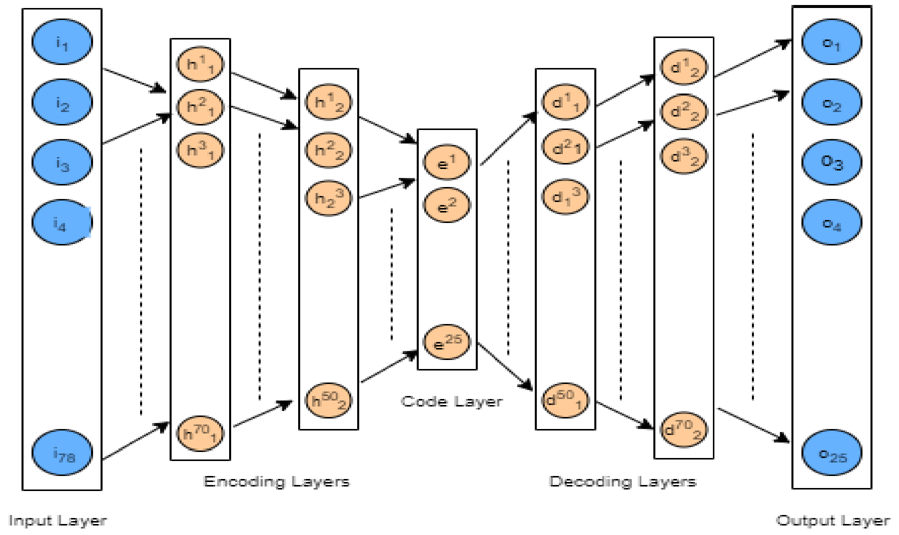
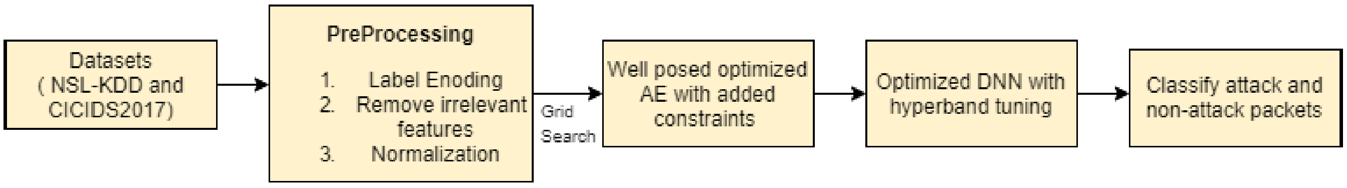


FIGURE 4. Architecture of optimized AE.



Datasets (NSL-KDD and CICIDS2017)
PreProcessing
1. Label Enoding
2. Remove irrelevant features
3. Normalization
Grid Search
Well posed optimized AE with added constraints
Optimized DNN with hyperband tuning
Classify attack and non-attack packets

FIGURE 5. Optimized AE+DNN model construction.

Algorithm 2 Preprocess_Data(D)

```
Input: Dataset D=($X_1$, $X_2$....$X_k$) with k features
Output: Dataset D'=($X_1$, $X_2$....$X_k$) with k features
Apply label encoding to each char value
Remove NaN, $\infty$ from D
for i = 1 to k do
    Compute $X_{norm\_i} = \frac{X - X_{min\_i}}{X_{max\_i} - X_{min\_i}}$
end for
return D' = ($X_{norm\_1}$,.....,$X_{norm\_n}$)
```

Algorithm 3 Train_AE for Training of Optimized AE

```
Input: AE training dataset -S, encoder $f_\phi$, decoder $g_\theta$
Output: Optimized AE model
Initialize weight matrices s.t. $\sum_{i=1}^{k} w_{i,j}^2 = 1$ and $W^TW=1$
Divide S into batches $b_1 \ldots, b_n$ s.t $|b_i| = 2500$ for all i except n, $|n| < 2500$
for t=0 to epochs do
    for i=1 to n do
        h = $f_\theta(S_{b1})$
        R = $g_\theta$(h) = $g_\theta$ ($f_\phi(s_{bi})$)
        loss d(X,R) = −(X.log(R)+(1-X).log(1-R))
        Update $\theta$ and $\phi$ using BPA of adadelta s.t. d(X,R) is minimum
    end for
    Use Grid search for best model
end for
Store optimized AE model → Opt_AE
return Opt_AE
```

It is computed using Equation (10) given below:

$$Acc = \frac{TN + TP}{TN + TP + FN + FP} \times 100 \tag{10}$$

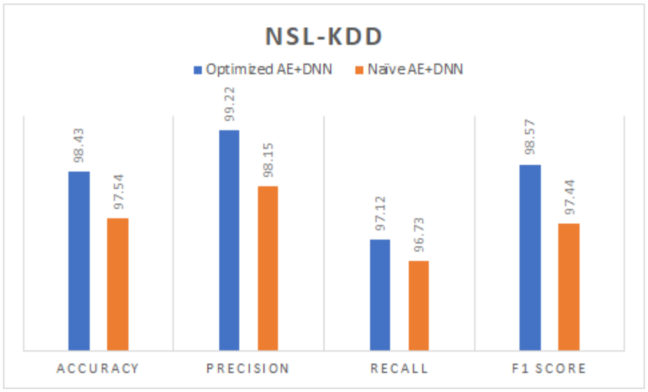
Precision (P): It is a measure of quality of the detection method i.e. the ratio of samples that are correctly classified as an attack to total samples in the test set that are classified as attack. It is computed using Equation (11) given below:

$$P = \frac{TP}{TP + FP} \tag{11}$$

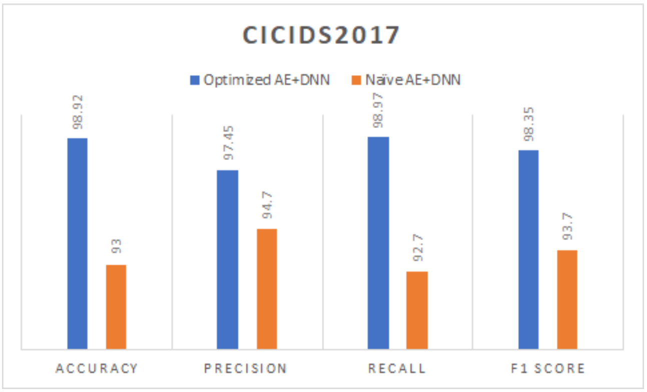
Recall (R): It is the measure of completeness of the classifier i.e. the ratio of samples that are correctly classified as an attack to total samples that are labelled as attacks in the test set. It is computed using Equation (12) given below:

$$R = \frac{TP}{TP + FN} \tag{12}$$

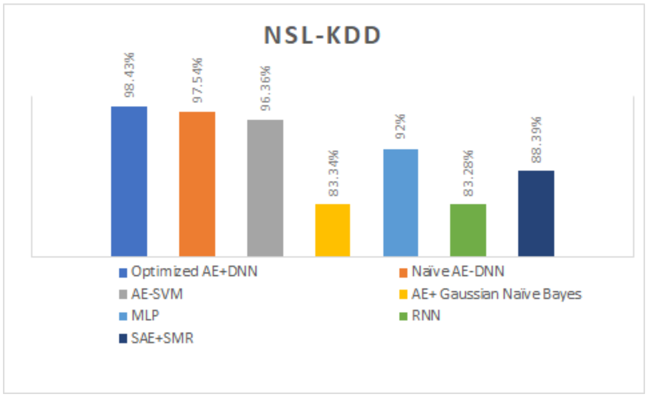
F1-score: It calculates the balance between precision and recall. It is considered as the harmonic mean of recall and



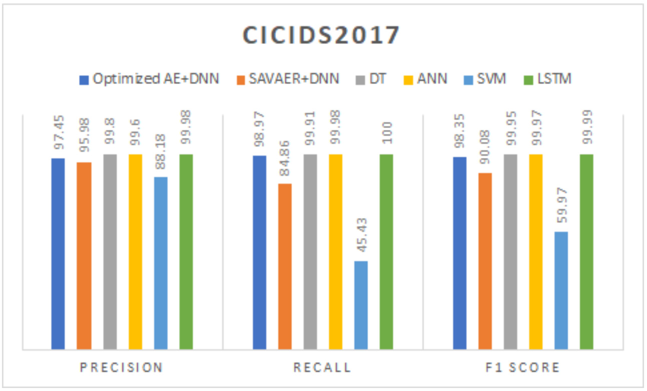
(a) Comparison of Performance Metrics for Optimized AE+DNN and Naive AE+DNN on NSL-KDD dataset



(b) Comparison of Performance Metrics for Optimized AE+DNN and Naive AE+DNN on CICIDS2017 dataset



(c) Comparison of Accuracy of state-of-the-art approaches on NSL-KDD dataset



(d) Comparison of Precision, Recall and F1-score of state-of-the-art approaches on CICIDS2017 dataset

FIGURE 6. Comparative performance of state-of-the-art approaches.

```
Algorithm 4 Apply_AE(D) to Obtain Encoded Features
  Input: Dataset, optimized AE model
  Output: Encoded feature representation
  Opt_AE(D)→ $h(D)$
  return $h(D)$
```

precision, and is computed using Equation (13) below:

$$F1_{score} = 2 \times \frac{P \times R}{P + R} \tag{13}$$

```
Algorithm 5 Train_DNN for Training Optimized DNN
  Input: Optimal feature subset dataset $D''$
  Output: Optimized DNN model
  Initialize network parameters
  Determine initial learning_rate
  Divide $D''$ into batches $D''_{b1} \ldots D''_{bn}$ s.t. $|D''_{bn}|$=batch_size for all i
  for t = 0 to epochs do
    for i=1 to n do
      Update network parameters using Adabound
      and accuracy as criteria
    end for
    Use Hyperband tuning for optimization
  end for
  Store Optimized DNN model → $Opt\_DNN$
  return $Opt\_DNN$
```

V. RESULTS AND DISCUSSION

To evaluate our proposed model, we compared its performance with ten state-of-the-art DDoS detection approaches based on deep learning available in literature. The comparison has been done over the NSL-KDD and CICIDS2017 standard datasets. The approaches chosen for comparison over NSL-KDD dataset are SAE+SMR [18], AE+Gaussian NB [20], RNN [21], MLP [22], AE+SVM [35], and SAVAER+DNN [39]. The approaches chosen for comparison over CICIDS2017 dataset are DT [32], ANN [32], SVM [32], LSTM [32] and SAVAER+DNN [39]. These approaches have been discussed in Section II. They have been selected for comparison as these are the most recent works reported over these datasets. Additionally, we have reported the results for the Naïve AE+DNN model implemented in this article to highlight the improvements achieved by proposed optimized model. The comparison has been done on the basis of four performance metrics viz. detection accuracy, precision, recall and F1-score.

TABLE 11. Comparison of Performance Metrics for State-of-the-art Approaches.

Approach	Dataset	Accuracy	Precision	Recall	F1-Score
Proposed optimized AE+DNN	NSL-KDD	98.43%	99.22%	97.12%	98.57%
	CICIDS	98.92%	97.45%	98.97%	98.35%
Naive AE+ DNN	NSL-KDD	97.54%	98.15%	96.73%	97.44%
	CICIDS	93%	94.7%	92.7%	93.7%
SAE+SMR [18]	NSL-KDD	88.39%	—-	—-	—-
AE+ Gaussian Naïve Bayes [20]	NSL-KDD	83.34%	—-	—-	—-
RNN [21]	NSL-KDD	83.28%	—-	—-	—-
MLP [22]	NSL-KDD	91.7%	—-	—-	—-
AE+ SVM [35]	NSL-KDD	96.36%	—-	—-	—-
	CICIDS	99.41%	99.67%	99.66%	99.66%
DT [32]	CICIDS	—-	99.8%	99.91%	99.95%
ANN [32]	CICIDS	—-	99.6%	99.98%	99.97%
SVM [32]	CICIDS	—-	88.18%	45.43%	59.97%
LSTM [32]	CICIDS	—-	99.98%	100%	99.99%
SAVAER+DNN [39]	CICIDS	89.36%	95.98%	84.86%	90.08%

The experimental results of naive as well as optimized model for all metrics are depicted graphically in Figure 6(a) over NSL-KDD and Figure 6(b) over CICIDS2017, respectively. Experimental results show that proposed approach based on optimized AE+DNN has given average accuracy of 98.43% over NSL-KDD dataset which is the highest among all techniques. Comparison of the accuracy results of state-of-the-art approaches over NSL-KDD dataset is plotted in Figure 6(c).

The accuracy of the Naïve model at 97.54% is also better than other techniques over NSL-KDD. It is better than AE+SVM and AE+Gaussian NB approaches which reported accuracy of 96.36% and 83.28%, respectively. The proposed approach has also performed better than the stacked AE and soft-max regression for classification of attacks [18] with 88.39% accuracy, RNN [21] with 83.28% accuracy, and MLP [22] with 91.7% accuracy, respectively. The proposed approach has also outperformed all other methods in terms of precision, recall and F1-Score over NSL-KDD dataset.

Experimental results for CICIDS dataset have been depicted graphically in Figure 6(b) and 6(d). Figure 6(b) shows that the accuracy of proposed approach is 98.92% and Naïve model has a low accuracy of 93%. The accuracy of proposed approach is slightly lower than 99.41% of AE+SVM [35]. The proposed approach outperformed SVM method which achieved 88.18% accuracy on CICIDS2017. In terms of other metrics also, proposed approach outperforms SVM [32]. The method SAVAER+DNN [39] reported 89.36% accuracy which is less than our proposed method.

However, the precision of proposed method is 97.45% whereas LSTM [32] reported precision value of 99.98%. Recall is 98.97% for proposed method whereas LSTM reported recall of 100%. The F1-score of proposed method is 98.35% whereas LSTM [32] reported 99.99%. This suggests that LSTM [32] may be overfitting the model to the training data and further investigation is needed to see the generalization ability of these methods over unseen data. LSTM is also known to need high memory bandwidth while training because of linear layers in each cell. In [32], DT and ANN reported recall 99.91% and 99.98% respectively on CICIDS2017 dataset. However, decision trees have the disadvantage of being over sensitive to small changes in data and may not work on unseen attack data. The ANN is fairly complex needing a vast amount of computing power which may not give output in near real time for timely detection of the DDoS attacks. Furthermore, the metrics over CICIDS2017 are lower because in proposed approach, the AE is trained over a sample of data. This has been done intentionally so that the model does not learn identity function and actual relationships between features can be effectively learned. The proposed model is expected to perform better over unseen data such as new attack vectors.

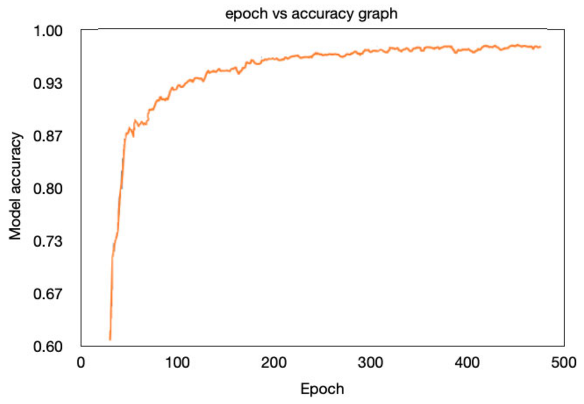


FIGURE 7. Optimized AE+DNN plot.

Table 11 shows the comparison between proposed approach AE+DNN and other state-of-the-art approaches over NSL-KDD and CICIDS2017 datasets in terms of various performance metrics.

Apart from analyzing the four performance metrics as given above, we have plotted the epoch versus accuracy graph for optimized AE+DNN in Figure 7 to show the convergence of the algorithm. As the graph indicates, the proposed optimized AE+DNN is stable with reasonably fast convergence. This faster training and stability of learning without overfitting and while using a compact model, are the main strengths of proposed approach. This can be attributed to the optimization techniques and novel mechanisms introduced by this article, which are detailed in Section IIII-D.

VI. CONCLUSION AND FUTURE WORK

Detection of DDoS attacks in cloud environment is imperative since there has been a surge in the intensity and frequency of such attacks which can potentially bring down entire computer networks such as critical networks of power grid, healthcare, etc. There is lack of classification methods that can handle imbalanced, voluminous, noisy data, high dimensionality data and still give accurate results. This work proposed an optimized AE and DNN architecture for classification of DDoS attacks. First, a naive AE and DNN model is constructed as a baseline model using random values for hyperparameters.This baseline model is further improved to yield an optimized AE and DNN model. Enhancements to the basic AE such as sparsity, unit norm, orthogonality, and hyperparameter optimization using Grid search, have resulted in optimized AE which has demonstrated potential in producing effective latent representation to give improvements in the classification results. These reduced and significant features are fed to an improved DNN for classification. DNN has been enhanced by intelligent learning rate determination and optimization using hyperband tuning of hyperparameters.The proposed approach has outperformed other state-of-the-art approaches for DDoS detection over NSL-KDD dataset by giving accuracy of 98.43%. For CICIDS2017, accuracy is reported as 98.92% which is competitive as compared to other state-of-the-art methods. In terms of precision, recall and F1-score, the proposed approach outperforms all other approaches over NSL-KDD and gives promising results over CICIDS2017 dataset.

In the future, further validation of the proposed approach will be undertaken to check its generalization ability across different datasets. Future work will be done to enhance the proposed approach to detect attacks in real-time traffic flows with reduced detection time and less computational complexity for analyzing big real time data. The ability to correctly identify attack traffic and recovery of system from attack is an important network security requirement, so future efforts will be devoted to propose technique for mitigation of effects of detected DDoS attack on the system.

REFERENCES

[1] (2019). *Cloud Attack: Economic Denial of Sustainability (EDoS)*. Accessed: May 4, 2019. [Online]. Available: http://www.elasticvapor.com/2009/01/cloud-attack-economic-denial-of.html

[2] (2020). *AWS Said it Mitigated a 2.3 Tbps DDoS Attack, the Largest Ever*. Accessed Jun. 30, 2020. [Online]. Available: https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever

[3] (2019). *Wikipedia Goes Offline Following DDoS Attack*. Accessed Jan. 12, 2020. [Online]. Available: https://www.techradar.com/in/news/wikipedia-taken-down-after-major-ddos-attack

[4] *Academic Research Reports Nearly 30,000 DoS Attacks Per Day*. Accessed Dec. 16, 2019. [Online]. Available: https://www.corero.com/blog/853-academic-research-reports-nearly-30000-dos-attacks-per-day

[5] J. Srinivas, A. K. Das, and N. Kumar, "Government regulations in cyber security: Framework, standards and recommendations," *Future Gener. Comput. Syst.*, vol. 92, pp. 178–188, Mar. 2019.

[6] E. Ozer and M. Iskefiyeli, "Detection of DDoS attack via deep packet analysis in real time systems," in *Proc. Int. Conf. Comput. Sci. Eng. (UBMK)*, Oct. 2017, pp. 1137–1140.

[7] B. Meng, W. Andi, X. Jian, and Z. Fucai, "DDOS attack detection system based on analysis of Users' behaviors for application layer," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, vol. 1, Jul. 2017, pp. 596–599.

[8] D. Sun, K. Yang, Z. Shi, and Y. Wang, "A distinction method of flooding DDoS and flash crowds based on user traffic behavior," in *Proc. IEEE Trustcom/BigDataSE/ICESS*, Aug. 2017, pp. 65–72.

[9] J. David and C. Thomas, "Efficient DDoS flood attack detection using dynamic thresholding on flow-based network traffic," *Comput. Secur.*, vol. 82, pp. 284–295, May 2019.

[10] S. Saharan and V. Gupta, "Prevention and mitigation of DNS based DDoS attacks in SDN environment," in *Proc. 11th Int. Conf. Commun. Syst. Ntw*, Jan. 2019, pp. 571–573.

[11] J. Hou, P. Fu, Z. Cao, and A. Xu, "Machine learning based DDos detection through NetFlow analysis," in *Proc. MILCOM-IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 1–6.

[12] A. Praseed and P. S. Thilagam, "DDoS attacks at the application layer: Challenges and research perspectives for safeguarding Web applications," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 661–685, 1st Quart., 2019.

[13] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019.

[14] A. Aldweesh, A. Derhab, and A. Z. Emam, "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues," *Knowl.-Based Syst.*, vol. 189, Feb. 2020, Art. no. 105124.

[15] (2018). *How Much Data Is Generated Every Minute? [Infographic]*. Accessed Dec. 16, 2019. [Online]. Available: https://www.socialmediatoday.com/news/how-much-data-is-generated-every-minute-infographic-1/525692/

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "A detailed analysis of the CICIDS2017 data set," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 172–188.

[17] *A Deeper Dive into the NSL-KDD Data Set*. Accessed Mar. 2020. [Online]. Available: https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657

[18] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (Formerly BIONETICS)*, 2016, pp. 21–26.

[19] S. A. Ludwig, "Intrusion detection of multiple attack classes using a deep neural net ensemble," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–7.

[20] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861.

[21] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[22] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (DoS) attack," in *Proc. IEEE Conf. Appl., Inf. Netw. Secur. (AINS)*, Nov. 2017, pp. 81–84.

[23] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[24] M. Idhammad, K. Afdel, and M. Belouch, "Distributed intrusion detection system for cloud environments based on data mining techniques," *Procedia Comput. Sci.*, vol. 127, pp. 35–41, Jan. 2018.

[25] D. A. Singh, R. Priyadharshini, and E. J. Leavline, "Cuckoo optimisation based intrusion detection system for cloud computing," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 11, p. 42, 2018.

[26] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[27] Z. Chiba, N. Abghour, K. Moussaid, A. El Omri, and M. Rida, "Intelligent approach to build a deep neural network based IDS for cloud environment using combination of machine learning algorithms," *Comput. Secur.*, vol. 86, pp. 291–317, Sep. 2019.
[28] R. Patil, H. Dudeja, and C. Modi, "Designing an efficient security framework for detecting intrusions in virtual network of cloud computing," *Comput. Secur.*, vol. 85, pp. 402–422, Aug. 2019.
[29] W. Elmasry, A. Akbulut, and A. H. Zaim, "Empirical study on multiclass classification-based network intrusion detection," *Comput. Intell.*, vol. 35, no. 4, pp. 919–954, Nov. 2019.
[30] B. Hajimirzaei and N. J. Navimipour, "Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm," *ICT Express*, vol. 5, no. 1, pp. 56–59, Mar. 2019.
[31] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Appl. Sci.*, vol. 9, no. 16, p. 3414, Aug. 2019.
[32] X. Liang and T. Znati, "A long short-term memory enabled framework for DDoS detection," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.
[33] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future Gener. Comput. Syst.*, vol. 111, pp. 763–779, Oct. 2020.
[34] W. Elmasry, A. Akbulut, and A. H. Zaim, "Evolving deep learning architectures for network intrusion detection using a double PSO Metaheuristic," *Comput. Netw.*, vol. 168, Feb. 2020, Art. no. 107042.
[35] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Comput. Netw.*, vol. 180, Oct. 2020, Art. no. 107390.
[36] M. Wang, Y. Lu, and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," *Comput. Secur.*, vol. 88, Jan. 2020, Art. no. 101645.
[37] S. Zavrak and M. Iskefiyeli, "Anomaly-based intrusion detection from network flow features using variational autoencoder," *IEEE Access*, vol. 8, pp. 108346–108358, 2020.
[38] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach.* Newton, MA, USA: O'Reilly Media, 2017.
[39] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, "Network intrusion detection based on supervised adversarial variational auto-encoder with regularization," *IEEE Access*, vol. 8, pp. 42169–42184, 2020.
[40] N. N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf.*, Nov. 2015, pp. 1–6.
[41] *Center for Applied Internet Data Analysis*. Accessed: Oct. 2019. [Online]. Available: http://www.caida.org/data/overview/
[42] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 184–208, 1st Quart., 2016.
[43] R. Lippmann, J. W. Haines, D. J. Fried, J. J. Korba, and K. Das, "The 1999 DARPA off-line intrusion detection evaluation," *Comput. Netw.*, vol. 34, no. 4, pp. 579–595, 2000.
[44] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho, "Creation of flow-based data sets for intrusion detection," *J. Inf. Warfare*, vol. 16, no. 4, pp. 40–53, 2017.
[45] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier, "Label encoding algorithm for MPLS segment routing," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2016, pp. 113–117.
[46] Y. K. Jain and S. K. Bhandare, "Min max normalization based data perturbation method for privacy protection," *Int. J. Comput. Commun. Technol.*, vol. 2, no. 8, pp. 45–50, Oct. 2011.
[47] (2017). *Autoencoders (Draft: Version 0.7.2)*. Accessed: Jan. 2020. [Online]. Available: https://www.math.snu.ac.kr/~hichoi/machinelearning/lecturenotes/Autoencoder.pdf
[48] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.
[49] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive gradient methods with dynamic bound of learning rate," 2019, *arXiv:1902.09843*. [Online]. Available: http://arxiv.org/abs/1902.09843

AANSHI BHARDWAJ received the Master of Engineering degree in information technology from UIET, Panjab University, India, in 2014, where she is currently pursuing the Ph.D. degree. She has an experience of five years in teaching. Her research interests include web mining, machine learning, and security in cloud computing.

VEENU MANGAT (Member, IEEE) received the Master of Engineering degree in computer science and engineering from the Punjab Engineering College (PEC), in 2004, and the Ph.D. degree in engineering and technology (computer science) from Panjab University, India, in 2016. She is currently working as an Associate Professor in information technology with UIET, Panjab University. She has a teaching experience of more than 15 years. Her research interests include data mining, machine learning, privacy, and security. She is also a Co-Principal Investigator in research project on "Monitoring of Active Fire Locations and Precision in Allied Agricultural Activities Using Communication Technologies" funded by the Ministry of Electronics and amp; IT of Government of India worth Rs. 75.75 lakhs, from 2020 to 2022. She has also worked on Research Project titled "Pedestrian Detection From Thermal Imaging" funded by the Design Innovation Centre of Ministry of HRD and consultancy project in the area of machine learning. She has successfully guided 21 Master of Engineering dissertations and is currently guiding seven Ph.D. scholars.

RENU VIG received the Ph.D. degree in engineering and technology in the field of artificial intelligence and neural networks from the Punjab Engineering College, in 1997. She was the Ex-Director of UIET, Panjab University, India, where she is currently working as a Professor of electronics and communications engineering. She has guided more than 12 Ph.D. students and successfully completed several research projects funded by the Government of India and corporate sector. She has published more than 120 research papers in reputed journals and conferences. Her research interests include fuzzy systems, artificial intelligence, neural networks, and next-generation networking technologies.