



A HYBRID GAN–DNN FRAMEWORK FOR INTELLIGENT ANOMALY DETECTION IN IOT NETWORKS

Vibhor Harit¹, Rajeev Dhaiya² and Umang Garg^{3*}

^{1,2}Department of Computer Science and Engineering, Desh Bhagat University, Mandi Gobindgarh, P.O BOX 147301, Punjab, India.

^{3*}School of Computer Science and Engineering, IILM University, Gurugram, P.O BOX 122011, Haryana, India.

¹<http://orcid.org/0009-0002-6203-6378>, ²<http://orcid.org/0009-0004-9314-7489>, ³<http://orcid.org/0000-0002-1815-5794>

E-mail: vibhordev@gmail.com, raj8878@gmail.com, umangarg@gmail.com

ARTICLE INFO

Article History

Received: March 16, 2026

Reviewed: April 16, 2026

Accepted: May 15, 2026

Published: September 30, 2026

Keywords:

IoT Network Anomaly,

DNN,

GAN-DNN,

Feature Selection,

Deep Learning,

IoT Security,

Internet of Things (IoT)

ABSTRACT

The Internet of Things (IoT) growth has significantly improved in a very short period that increased the network threats and risks of anomalies pattern and other threats such as exploitation attacks, denial of service (DoS), and fuzzing, which can disrupt services and compromise data security. Conventional anomaly detection algorithms, including ML and DL models, tend to fail to detect various dynamic patterns of attacks due to skewed datasets and low generalization. To address these limitations, this paper proposes an embedded and integrated Generative Adversarial Network Deep Neural Network (GAN-DNN) model of detecting abnormalities in IoT networks. An artificial data generation in a GAN is then applied to a DNN classifier, with feature optimization so that the accuracy of detection can be increased when the data is diverse because of an artificially generated realistic anomaly data. Integrated strategy performs better than standalone DNN models as it leverages the great learning capacity of DNNs and the capability of GANs to form complex data distributions. The proposed GAN-DNN model was evaluated experimentally on benchmark datasets of IoT network traffic, and its accuracy surpassed the accuracy of the baseline DNN of 0.84 with a model accuracy of 0.91. It is interesting to note that it significantly improves the recall of exploits and fuzzers and gives perfect detection of DoS attacks with performance measures such as recall, precision, and F1-score. The effectiveness of the results shows significant improvements in terms of real-time detection of IoT network anomalies.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

I. INTRODUCTION

The IoT refers to the connectivity of numerous devices, also known as things, such as sensors and actuators, which are connected through the Internet to exchange data, allowing them to be utilized in a variety of applications [1]. The connected devices of the network are only partially capable of storing and processing data. IoT networks are more attractive targets for attackers because of the enormous amount of data produced by the growing number of different physical objects linked to the Internet [2]. Despite their limited capability, fewer processing units, and less storage, IoT devices provide a wide range of healthcare and agriculture applications. The main issues with IoT applications include inequality, scalability, processing equipment (i.e., low processing and storage capacity), and safety barriers [3], making it difficult to implement appropriate safety measures. The various services provided by IoT applications are extremely beneficial to human life. IoT devices gather personal data as well as user behavior when a user enters or exits a room. However, these devices have some limitations, such as limited computing power and storage; thus, the data are subsequently sent to the cloud for storage [4]. Because personal information is stored in the cloud, hackers can easily access it. The primary concerns with IoT devices are security and privacy [5]. ML and DL techniques are considered appropriate and yield better computational outcomes when performing computations on large amounts of data. The machine learning model is applicable after feature selection and optimization [6].

To address safety concerns in IoT networks, an intrusion detection system (IDS) can be established. In the last several decades, systems that detect infiltration based on intelligent techniques have demonstrated promise in identifying infiltration into IoT networks. As a result, these systems are increasingly preferred in the field of cyber security and are ideal for identifying hazards in IoT networks. Depending on the behavior, anomaly-based IDS can identify new attacks in the system; any unusual behavior is considered an attack. The IoT architecture comprises different layers, including the perception, network, and application layers. Different types of attacks, including replay, data injection, and node capture, can compromise data secrecy at the perception layer. Data is encrypted before being sent over the network to address this problem. The adversary flood of packets to prevent the target server or device from functioning as intended are prevalent in the network layer. A MIM attack, in which a hostile party eavesdrops on data to compromise their privacy at the network layer, is also rather likely. This requires software and protocols to detect and defend against attacks on network objects [7]. At the application layer, application-specific services are monitored, as it is critical to keep an eye on who is sharing the data and how much data is being shared among users. Authentication and privacy are challenging because various applications require different methods. Several threats can occur at different layers of the IoT architecture, such as code injections, DDoS attacks, and node capture at the application, network, and perception layers, respectively. There are several risks identified by the researchers in IoT network like reconnaissance, DoS, and Sybil attacks. At IoT network layer, DoS/DDoS attacks are the most common. Several detection models exist for IoT-based DDoS attacks. Nevertheless, there are certain problems with applying these models to real-world scenarios. There are distinct datasets available publicly that contain the IoT network traffic data.

However, an imbalanced dataset can mislead the results due to data cleaning or augmentation. The model may behave as anomaly in case of larger samples for a single class. So, the dataset must be balanced before training the model. Both oversampling and undersampling were used to balance the dataset. Overfitting may occur in case of oversampling with majority of classes while undersampling provides fewer majority classes, whereas oversampling indicates more minority classes [8]. The impact of an uneven dataset on model training can be mitigated by generating data augmentation with minority classes. Research Gap: The computational cost can be reduced by selecting appropriate characteristics from the dataset, thus reducing the number of neurons, because large feature datasets require complex deep learning models with many neurons and massive processing. Choosing the best features for classification may limit the dataset rather than examining the complete set of features. This will shorten the time required to train the model. A less sophisticated model with fewer layers could result in an underfitted model that does not perform well on the training set. Conversely, a larger number of dense layers in a model with greater complexity could lead to an overfitted model that performs poorly on the testing dataset; this issue can be resolved by using various regularization strategies. It uses a filter-based correlation technique and applies a deep learning model for the detection and classification of features. Attribute selection is achieved by using dimension reduction and model training time, one feature is removed from the dataset, and only strongly associated features are retained. The model's parameters and hyperparameters were modified after testing it on a publicly available dataset. To improve its functionality, the model was regularized. To fill the research gaps, distinct contributions are provided in the current article such as:

- To detect anomalies in the IoT network using the DNN and GAN-DNN models.
- To categorize the features based on the embedded approach that integrates a correlation-based filter and a wrapper method.
- To compare and evaluate the existing system with the proposed method.

This is how the remaining sections of the paper are structured. The literary work done by several researchers is described in Section II. The suggested framework and the general technique used in this investigation are explained in Section III. The dataset's feature and attribute ranges are described in Section IV. The results are discussed and explained in terms of the evaluation criteria in Section V. The paper's ramifications are covered in Section VI.

II. LITERATURE REVIEW

The WFEU-FFDNN wireless intrusion detection system model was proposed by S. M. Kasongo et al. [9]. It is composed of a Feed-Forward Deep Neural Network (FFDNN) for classification and extraction of features based on a wrapper approach (WFEU). The best feature identification is one of the critical challenges during the feature extraction process. The Extra Trees approach reduces the number of features and parameters. The AWID IDS and UNSW-NB15 were the two datasets used to assess the model's effectiveness. Python Scikit-Learn was used to conduct the studies. Its binary classification accuracy was 87.10 and multiclass classification accuracy is 77.16 with UNSW-NB15 dataset. It achieved the 99.6% accuracy as binary classification and 99.7% accuracy for multiclass classification. Bae et al. [10] proposed a model with autoencoder that is based on deep learning techniques for the anomaly detection in industrial IoT networks. The model was constructed in TensorFlow library and Python programming which consisted of three encoder and decoder layers with different numbers of neurons. It was thereafter applied on real data of Korean steel businesses and benchmarking KDD cup 99 dataset. The result indicates that the accuracy of the KDD Cup, 99 data was between 84 and 100 percent and 95 percent in actual data.

Ferrag et al. [11] proposed a model that based on the deep discriminative and unsupervised models and optimized them using different parameter sets such as count of hidden nodes and learning rates. The model was developed using two real publicly available traffic datasets: The Bot-IoT and CICIDS2018 datasets. The authors have used TensorFlow and a graphic card to produce datasets on multiple tests. The CNN model with CSE-CIC-IDS2018 dataset and 100 & 0.5 learning rate that achieved 97% and 98% on the Bot-IoT dataset respectively attained an accuracy of 97.376 and 98.371 respectively, being among the various deep discriminative models, the CNN, RNN, and DNN. These are various generative models, the highest accuracy of 97.372 in CSE-CICIDS2018 databank and the highest accuracy of 98.394 in Bot-IoT databank was obtained with deep autoencoders 0.5 learning rate and 100 hidden nodes. Alavizadeh et al. [12] suggested a model of unsupervised learning of features with a nonsymmetric deep autoencoder (NDAE). The experiment was run in Python on a graphics processing unit (GPU)-based TensorFlow and the model was implemented as stacked Denoising Autoencoders (NDAEs). The NSL-KDD and the KDD datasets were evaluated using the proposed model stacking autoencoder. The proposed model with KDD cup 99 dataset achieved 97.85% accuracy and 100% accuracy with NSLKD dataset.

George et al. [13] designed a DNN intrusion or attack detection model and employed the NSL-KDD dataset with multiple activation functions, optimization tools, and initialisation parameters. According to the author, the total accuracy of the experimental results was 97%. Using the Keras library, Haija et al. [14] proposed a DNN model with a ReLU activation function, 150 epochs, and 3 hidden layers. The simulation is conducted through Cooja simulator and Contiki OS with setup configuration. The experiment was conducted for the detection of anomaly that achieved good results in terms performance parameters. The DL methods outperformed existing IDS based on the inverse weight clustering technique for IoT networks, with precisions of 95.7%, 96%, 99.5%, and 96% for opportunistic, DDoS, sinkhole, and wormhole attacks, respectively. A novel IDS for IoT networks was introduced by Syed et al. [15], who utilized some deep learning algorithms with the BoT-IoT dataset for detection and mitigation. The FNN architecture was designed with two fully connected hidden layers, each comprising multiple neurons activated by the ReLU function. The output layer was implemented as a dense layer with two neurons and a softmax activation to generate class probabilities. To handle high-dimensional categorical attributes, transfer learning–based encoding techniques were applied.

This encoding strategy was consistently employed for both binary and multiclass classification tasks. Different regularization strategies, learning rates, optimizers, epochs, and number of hidden layers were used to modify the model. The experimental results using the mFNN model to classify multiclass simultaneously using four classes had a 99.79 percent accuracy, and the bFNN model which operates in binary classification had a 99 percent accuracy with a small percentage of the packets being classified as false alarms to different threats. Ahmad et al. [16] proposed unique deep learning frameworks combined with an autoencoder in classification of network threats and IoT network traffic. The UNSW-NB15 and NSL-KDD99 datasets were modeled in Python with the help of the Keras package. According to the experimental results, these datasets have maximum F1-scores of 99.28% and 95.76% respectively. Among all the models, DNN was the most accurate (99.24). Gulzar et al. [17] proposed a DFEL framework to identify intrusions in the IoT networks. The model is based on the transfer learning and DL methods of reducing the dimensions and the features of the dataset. DFEL learnt the data was nonlinear and feature embedding helped the classifier in identifying cyber-attacks in the IoT data. The model's accuracy was 93.13%.

III. PROPOSED METHODOLOGY

The DNN model shown in Figure 1 to detect intrusion and vulnerabilities in an IoT network. It has distinct modules, such as training module, feature selection, feature pre-processing, data augmentation, and data processing. The first step of data pre-processing involves extracting the data, encoding the symbolic data using different encoding techniques, and then normalizing the dataset using min-max normalization. The dataset is then balanced using GAN approaches in the subsequent data augmentation stage [18]. The important features are chosen for the model's training in the feature selection step. Feature pre-processing separates the dataset for distinct purposes that is followed by the training of the DNN model.

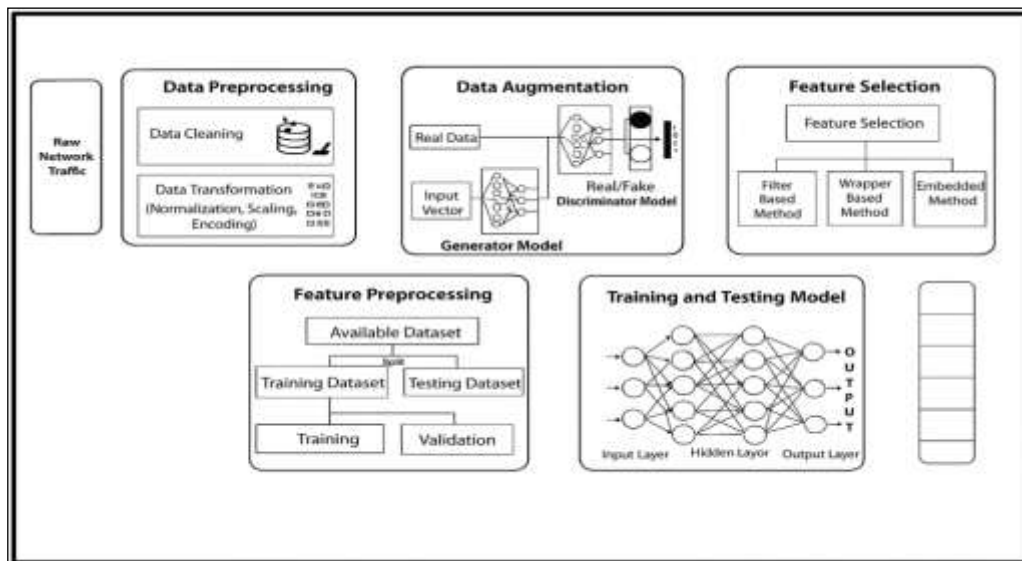


Figure 1: Proposed Methodology of the system.

Source: Authors (2026).

III.1 DATA PREPROCESSING

After duplicate data is removed, the dataset is extracted, processed, and normalized. During data transformation, a range of encoding techniques are used to translate the nominal values of the dataset's attributes into numerical values. The encoder technique is used to convert the nominal qualities into integer values. The dataset is then standardized using min-max normalization, which places values between 0 and 1, to enhance the performance of the model. After redundant data is removed from the dataset, it is first extracted, and then it is transformed and normalized. In data transformation, the nominal feature values of the dataset are categorized in the encoding technique, which converted them into numerical values. The nominal properties are encoded into integer values via label encoding. By utilizing Min-Max normalization, the dataset is normalized so that values are between 0 and 1, able to improve the overall performance of the model. It uploaded the CSV file with the extracted dataset to Google's Collaboratory after it was extracted. The dataset's category features were converted to integer values using label encoding after superfluous columns were eliminated. Every value in the column is converted to a numerical value during label encoding.

The dataset must be normalized to a single scale with values between 0 and 1 because it contains values on many scales. "Proto," "service," "state," and "attack cat" are four categorical value attributes that have 133, 13, 11, and 10 categories in UNSW-NB15, respectively. After transforming the category variables into integer values, it employed label encoding to encode them. Min-max normalization is used to normalize the dataset so that the value falls between 0 and 1.

III.2 DATA AUGMENTATION

In case of non-uniform and imbalanced data, the distribution is placed by using data augmentation. It may occur that the skewness of data if it favors the class with more samples. The technique of resampling can be used to address class imbalance. Oversampling and undersampling techniques are used for resampling [19]. More minority-class samples are obtained through oversampling, while fewer majority-class samples are obtained through undersampling. The issue of underfit can be brought on by inadequate sampling. The oversampling technique is frequently used by researchers to expand the dataset for building the ML-based model. Oversampling can result in the overfitting problem when the minority class's sample size is increased using the same data from the class. A range of oversampling techniques are used by the researcher. The SMOTE generates noise and class overlaps to choose samples from the minority class and generate new samples [20]. Recently, GANs have been utilized to create synthetic data that resembles real data. Compared to other traditional techniques like SMOTE, GANs are superior because they eliminate overfitting, class overlaps, and noise issues. Additionally, they enable model tuning, which contributes to the development of an accurate model. The main aim is to create a balanced dataset by creating new samples from the current one. GANs are employed to augment the minority attack classes by generating additional packet samples. This synthetic data generation process enhances class representation and leads to a more balanced dataset. Two distinct neural networks work together to form GANs; one creates synthetic data, and the other recognizes both real and synthetic data.

New samples are synthesized from the original dataset to achieve class balance. Consequently, GANs are utilized to produce additional packet instances for the underrepresented attack categories, improving their representation in the dataset. A more balanced dataset results from the creation of synthetic data. Goodfello et al. initially introduced GANs, which are composed of two distinct neural networks that cooperate to generate duplicate data and recognize both real and duplicate data. Generic networks create synthetic data from an input dataset, whereas discriminating networks distinguish between actual and fake data. In the current use, it only saw five classes; The most prevalent people were general, exploitation, and phasers. To generate new data for minority dos category, it employs a GAN after selecting 25K packets to reduce the majority classes. The sample size was randomly chosen to be 25K packets for the class with more packets than that size, while all packets were chosen for the class with less packets than that sample size. Specifically, it randomly selected 25K size packets from the dataset for each class that had more than 25K packets. A total of 115599 packets, comprising four attack type classes and normal packets, were recovered from the dataset. Since there were fewer than 25,000 DoS packets, it used GANs on the data and used the generator and discriminator model to create the packets. Twenty-five thousand packets are created by concatenating the generated packets with the original packets. Ian Goodfellow's Gan model consists of two neural networks: to generate the synthetic data, a neural network generator function G is utilized. And this synthetic data segregates from the real data by using discriminator neural network function D (Figure 2). Whereas the generator seeks to produce data that closely matches genuine data, the discriminator model seeks to differentiate between produced and true data. How the GAN model works is as follows:

The generator G creates data by using z as its noise input. The system transmits both generated data and actual data to discriminator D . The discriminator (D) distinguishes between synthetic and authentic data by assigning a value of 0 to generated samples and 1 to real samples. The generator aims to minimize the loss function, whereas the discriminator attempts to maximize it. The objective of the GAN is formulated as a minimax problem, expressed as $\text{Min}(G) \text{Max}(D)$, as presented in Equation 1.

$$V(D,G)=E_{x \sim P_{data}(x)}[\log D(x)] + E_{z \sim P_z}[\log(1-D(G(z)))] \quad (1)$$

Where D represents the discriminator, G indicates generator, and x indicates the sample collected from real data. And z is the sample collected from generator data, $P_{data}(x)$ represents the distribution of real data, P_z indicates the distribution of generator data, $D(x)$ is discriminator generator, and $G(z)$ indicates the generator network. The generator $G(z)$ generates distribution P_g from noise distribution P_z while attempting to match real data distribution $P_{data}(x)$. Discriminators attempt to distinguish between actual data and data that has been generated. GAN is trained in the form of the sequence of updates to the generator (G) and the discriminator (D) until the latter cannot reliably differentiate between actual and generated samples. The discriminator attains its optimal value when the product of the generated data distribution (P_g) is the same as the real data distribution ($P_{data}(x)$). The three-hidden-layers architecture has 32, 64, and 128 neurons, respectively, and features the Leaky ReLU activation, and the output layer has 39 neurons with a sigmoid activation function, as shown in [21]. The discriminator consists of four hidden layers that use Leaky ReLU activations, and have 128, 64, 32, and 16 neurons, and a single-neuron output layer with a sigmoid activation. Training of the model and the weights was done after putting the model together and 1000 epochs with the binary cross-entropy loss function and Adam optimizer. The dataset maintains its balance because it contains 25000 packets which belong to each class [22]. The dataset contains one label which shows the average assault type class together with 44 columns and 39 numerical values and four category values. The graph in Figure 3 shows how the discriminator and generator losses change at the x-axis where generator and discriminator loss represents blue and orange lines. Figure 2 illustrates the overview of Feature Selection Model.

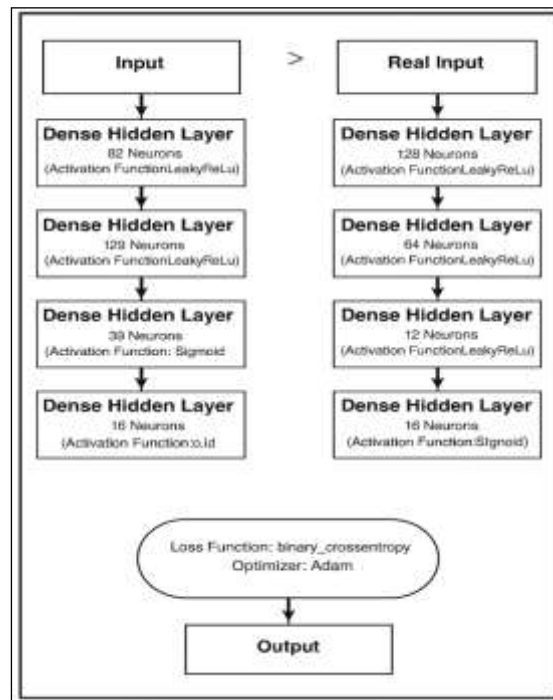


Figure 2: Feature Selection Model.
Source: Authors, (2026).

III.3 FEATURE SELECTION

Feature selection approaches lower computing and storage overhead by minimizing the number of features. In the correlation-based filter strategy, which selects features using a correlation-based solution, score and the threshold value are utilized to evaluate which features or properties are advantageous. This is among other feature selection methods. When the score of correlation of a feature is higher than the threshold, the feature is strongly correlated. It is filtered with screening to eliminate repetitive or overlapping features, and features with very strong correlation between them to drop one of the very correlated characteristics in the dataset. Wrapper Methods: This method involves the induction method to choose the subset of features. Forward selection and backward removal processes are used to select the subset of features. Backward elimination involves inclusion of all the features in the dataset and then features that worsen the performance of the model are dropped. The forward selection begins with an empty subset to which it then adds features that improve the functionality of the model to produce the optimum feature subset. Embedded Techniques: The embedded technique is combined with wrapper techniques and correlation filters techniques. Filter methods, like correlation or mutual information, rank features with high statistical relevance quickly; features with low statistical relevance are pruned off search space, minimizing search. Thereafter, wrapper methods, such as recursive feature elimination using a predictive model, estimate the subsets of the filtered features on model performance metrics, to guarantee the best relevance and least redundancy. This is an intermediate process that incorporates efficiency and accuracy without the tediousness of the exhaustive search of pure wrapper procedures and with predictive quality superior to solitary filters. It is widely used in high-dimensional datasets for improved model generalization. The correlation-based filtering method was used to choose the attributes. The feature selection improves storage efficiency and cuts down on computation time. The correlation-based filter strategy, which chooses features based on correlation score, and the optimized features from the dataset. Strongly correlated features are selected and any one of them is eliminated in favor of features with a PCC value higher than the 0.95 cutoff values as shown in Figure 3.

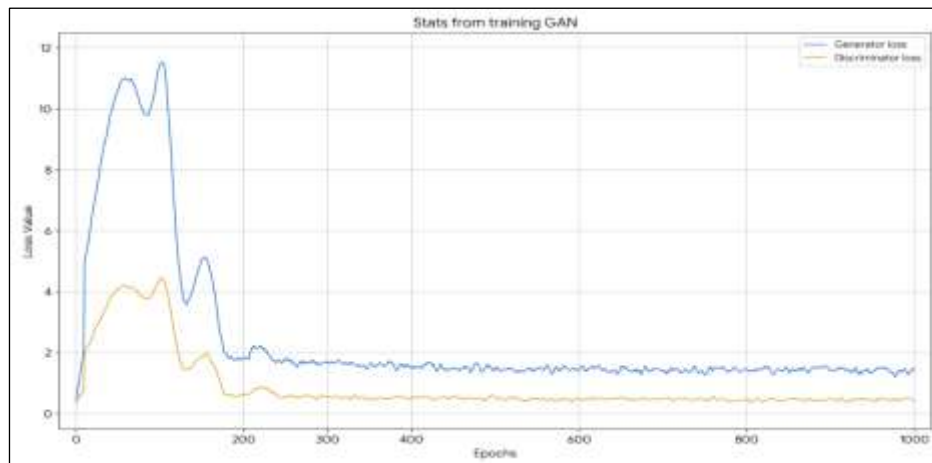


Figure 3: GAN LOSS.
Source: Authors, (2026).

In the model development process, a correlation-based facility selection technique was employed to refine the UNSW-NB15 dataset by removing the unrelated characteristics. This approach analyzes the interdependence between characteristics and eliminates those that are highly correlated, as they can introduce excesses and reduce the efficiency of the model. Properties such as CT_SRC_Dport_Ltm, DLOSS, DWIN, CT FTP CMD, Label, and CT SRV DST were excluded after corruption analysis, resulting in 38 features [23]. Figure 4 shows the correlation of the dataset to the matrix, highlighting significant relations between the remaining characteristics. The feature selection criterion was based on a range where one of the highly corrected variables was removed if the correlation price exceeded 0.95. This is ascertained to eliminate the least informative and non-confidential features and consequently, an increment in the data quality, enhanced performance of the model and consequential analysis and classifications that simplify the computational complexity in classification and classification functions.

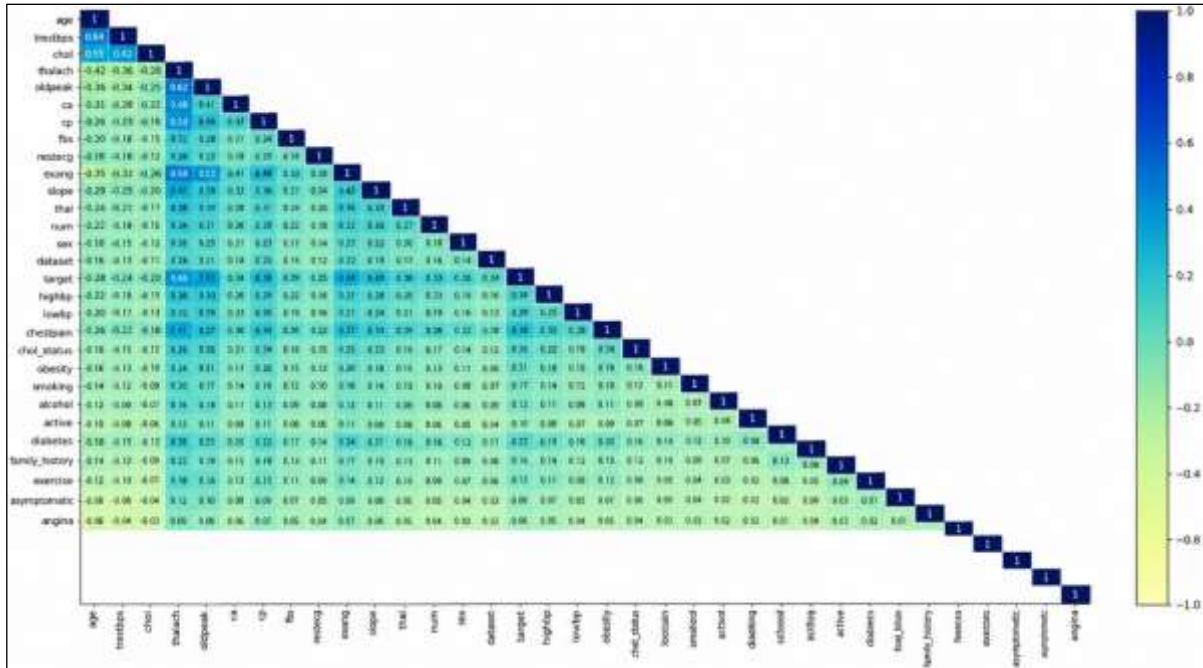


Figure 4: Correlation Matrix of the dataset.
Source: Authors, (2026).

III.4 FEATURE PREPARATION

The features were encoded and normalized into a structured format suitable for machine learning, followed by dimensionality reduction to refine the dataset. To ensure reliable training and evaluation, the dataset was partitioned into training, validation, and testing subsets. Initially, the data were split into 75% for training and 25% for testing, allowing a substantial portion of the dataset to be used for assessing the model's performance. The training set was divided within this, with 60% used only in model training and 15% used as the validation set, where fine-tuning parameters had to be fine-tuned to avoid overfitting. This enables one to use the structured separation training data to learn patterns, verify set to be used to adapt the model, and test set to situate the fair performance [24]. Through this approach, the processed data becomes properly structured to be easily trained so that the final model is not only correct but also normal to unseen data.

III.5 TEST AND TRAIN MODEL

The deep learning model is made to learn on a processed set of data that has normal and attacking classes, which is in the right format to be used as the input of the model. In training, the verification dataset is also used to check the accuracy and make sure the model normalizes well. The test data is then applied to test the models on the ability to classify unseen data as being either general or attack. Training and test datasets are formed out of the initial preprocessed dataset, the training is applied to fit the model, particularly using the most and learn patterns. This systematic method initiates high model development and valid assessment on unseen network traffic. It then evaluates the model in terms of its generalizability to the test set, which can be a new dataset or unknown dataset. In the model training, only 60 percent of the training data will be used to train the model, and another 15 percent will be used as a model validation dataset. During the training stage, the model differentiates between normal and attack type and hence determines the training accuracy. After that, in the process of testing the model using the testing dataset, the accuracy of the model is tested to determine how well it works on the unknown data. The model architecture comprises multiple dense hidden layers, each characterized by a distinct number of neurons and an activation function. The model is created using the Keras package with TensorFlow and the Google Colab platform for simulation. To create the deep neural network model, three dense hidden layers are utilized. Every covered layer consists of 64 neurons that use the ReLU activation function. As there are five different classes in the set, the output layer will have five neurons whose activation function will be softmax to produce the probability of the classes. This model is then summed and the loss, or the difference between the predicted values and the actual values, is found using sparse category cross-entropy which finds the index of the most likely class. The model weights are then updated with the Adam optimizer based on the calculated loss as in Figure 5.

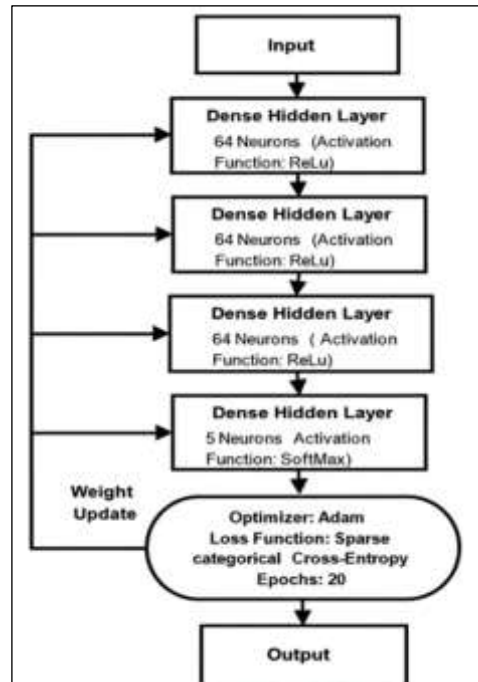


Figure 5: DNN Architecture.

Source: Authors, (2026).

Over-fitting and under-fitting are two major issues in training the model. A model is said to be overfitted if it performs well on the training dataset but has trouble generalizing to a novel or untested dataset. Overfitting can be prevented in several ways, including by regularizing the model and modifying its parameters. It also adjusted several parameters, like hidden layers and neurons, and regularized hyperparameters, including weight decay and dropout rate, to improve classification accuracy and avoid overfitting. With a 0.0001 dropout rate and 0.0001 weight decay, it employed the L2 regularization algorithm. Due to the size of the dataset, various regularization strategies do not increase the classification accuracy. The model is constructed with diverse DNN structures featuring varied dense hidden layers and neuron quantities within each layer to enhance accuracy. Following that, it experimented with different neuron and dense hidden layer values before using the DNN model to optimize three layers with 64 neurons. The number of neurons and dense hidden layers in the model determines its complexity. A little number indicates that the model is underfit and has little complexity, whereas a large number indicates that the model is overfit and has a lot of complexity. The Adam optimizer is a learning rate optimization algorithm that combines ideas from RMSprop and momentum approaches. The learning rate would be adjusted according to the gradient parameters with 0.001 default learning rate and enhance the accuracy of the model. Experimenting different learning rates allows you to observe how sensitive the model is to variations in this hyperparameter.

III 6 FEATURE PREPROCESSING

The attributes that were not eliminated under the feature reduction were then converted into machine learning format in an organized form through encoding and normalization. To support sound training and evaluation, the dataset was divided into training, validation, and testing parts. To begin with, 75 percent of the data was to be used in training and 25 percent in testing to provide adequate data to evaluate performance. In the training section, model learning consumed 60 percent, and the remaining 15 percent were a validation set to be used in hyperparameter tuning and overfitting prevention. This enables one to use the structured separation training data to learn patterns, verify set to be used to adapt the model, and test set to situate the fair performance. Through this approach, the processed data becomes properly structured to be easily trained so that the final model is not only correct but also normal to unseen data.

Algorithm 1: GAN–DNN–Based IDS Framework

Input: Raw UNSW-NB15 dataset D.

Output: Trained DNN model for intrusion classification

Step 1: Data Preprocessing

- 1.1 Remove duplicate and irrelevant records from D.
- 1.2 Encode categorical attributes (proto, service, state, attack_cat) using label encoding.
- 1.3 Apply Min–Max normalization to scale numerical features into [0,1].

Step 2: Data Augmentation Using GAN

- 2.1 Identify minority and majority attack classes.
- 2.2 Select a fixed sample size (25K) for each class.
- 2.3 Train the GAN using real samples from minority classes.
- 2.4 Generate synthetic samples until class balance is achieved.
- 2.5 Combine real and generated samples to form a balanced dataset D_b .

Step 3: Feature Selection

- 3.1 Compute Pearson Correlation Coefficient (PCC) among features.
- 3.2 Remove one feature from each highly correlated pair ($PCC > 0.95$).
- 3.3 Obtain the optimized feature subset F_{opt} .

Step 4: Feature Preprocessing and Dataset Splitting

- 4.1 Construct final dataset using D_b and F_{opt} .
- 4.2 Split dataset into training (60%), validation (15%), and testing (25%) sets.

Step 5: DNN Model Training

- 5.1 Initialize DNN with three hidden layers (64 neurons each, ReLU activation).
- 5.2 Apply L2 regularization and dropout.
- 5.3 Train the model using the training set.
- 5.4 Validate model performance using the validation set.

Step 6: Model Testing and Evaluation

- 6.1 Evaluate the trained DNN using the test set.
- 6.2 Compute performance metrics (Accuracy, Precision, Recall, F1-score).

End Algorithm

IV. DATASET DESCRIPTION

To evaluate the model, researchers are using several publicly accessible datasets. UNSW-NBAnd NSL-KDD are the standard evaluation datasets that academics utilize most often for NIDS (Table I). To handle square imbalance in the dataset, it implemented a balanced sampling strategy. For classes with records less than 25,000, such as futures, dos, and exploits, all available records were included to preserve data. In contrast, for general and general classes, which had many records, it chose 25,000 samples randomly to maintain balance in categories. This ensured that no class dominates the dataset, reducing prejudice during model training. The final balanced dataset, which has a representative distribution of all classes, is summarized with complete convenience of the set in Table 1.

Table 1: Features of UNSW-NB15 Dataset.

Attribute	Attribute
dpkts	dtepb
dur	stcpb
dbytes	tcprtt
spkts	dwin
rate	ackdat
sbytes	synack
dttl	dmean
sttl	smean
dload	responsebodylen _
sload	transdepth
sloss	_ isftlogin
dloss	_ ctsrvsrc
dinpkt	_ ctdstltm
sinpkt	ctsrcdportltm
djit	_ ctdstsportltm
sjit	_ ctstatettl
ctsrcltm	_ ctsrvdst
ctftpcmd	label
swin	state
issmipsports	service
attackcat	proto

Source: Authors, (2026).

UNSW-NB15 dataset: This dataset is generated by Australian Centre for Cyber Security (ACCs), which is frequently used to identify infiltration [25]. It contains 44 features, including 39 numerical features, 4 classified characteristics, and a label that recognizes whether a record represents general traffic or attack. As detailed in Table 1, dataset includes a general class and nine different categories: generic, reconnaissance, dos, exploitation, analysis, insects, shellcode, backdoor and phases. This diverse mixture of features and types of attacks makes UNSW-NB15 an extensive benchmark for evaluation of infiltration systems and cyber security models. As indicated in Table 2, the dataset includes 93000 records in the Normal category and 164673 recordings across all attack categories, for a total of 257673 records. It selected four attack types of Generic, Exploits, DoS, and Fuzzers as well as one normal class for experimental evaluation. To choose a sample size of 25K for classes with more than 25K records.

Table 2 :DatasetStatistics.

Classification	Packet Numbers	Classification	Packet Numbers
Generic	58871	Analysis	2677
Normal	93000	Exploits	44525
Reconnaissance	13987	Backdoor	2329
Worms	174	DoS	16353
Fuzzers	24246	Shellcode	1511
		TotalPackets	257673

Source: Authors, (2026).

IV. RESULTS AND DISCUSSIONS

The 28900-testing dataset is used to confirm the accuracy of the DNN model, which was constructed utilizing original samples and 86699 training datasets over 20 epochs for classification of data as multi-class. After training with additional samples (a total of 93184 training datasets and 31062 testing datasets across 20 epochs) the DNN model with improved no of attacks that can harm the system using GAN, it evaluated the performance of the model with some characteristics. Figure 6 illustrates how it able to calculate the confusion matrix based on the quantity and proportion of distinct class packets. Confusion Matrix results highlight the better identification capacity of the GAN-DNN model compared to the traditional DNN model, especially for dos attacks. In the DNN model, 2,901 True positives (TP) were identified for DOS attacks, which represent 10.04% of the test dataset. In contrast, the GAN-DNN model detected 6,203 TP, which indicates an increase of 10% in DOS samples correctly identified due to the size of the larger sample. Fewer FPs result in a high precision rating, while fewer FNs are indicated by a high recall score.

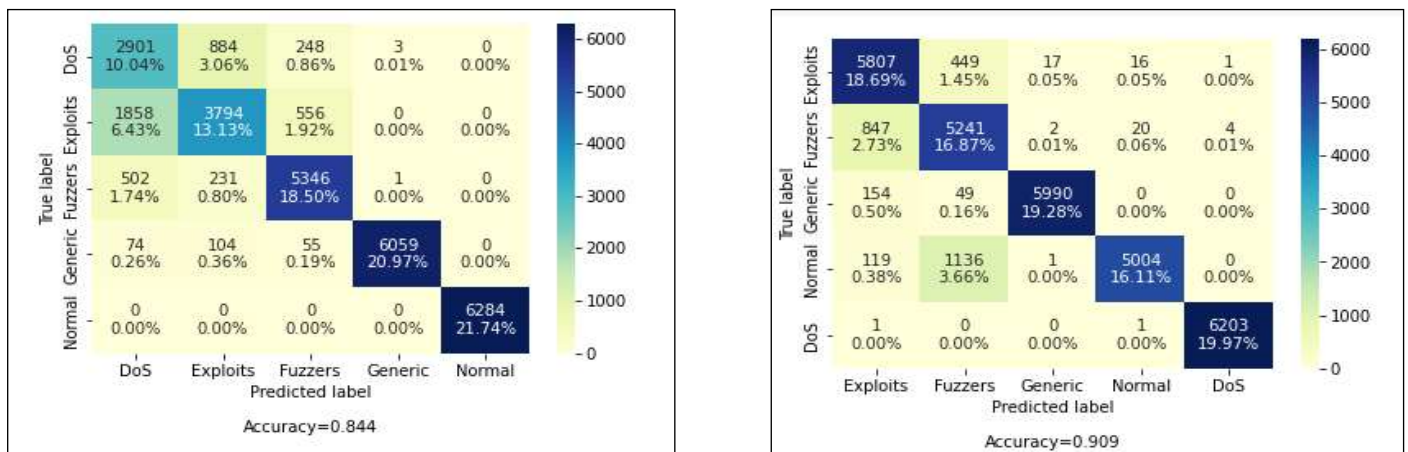


Figure 6: DNN-based, GAN-DNN Confusion Matrix.

Source: Authors, (2026).

As shown in Table 3, the assessment matrix for multi-class classification also shows significant improvements. DNN recalled the accuracy of 0.84 and 0.91 respectively, while GAN-DNN achieved 0.91 in total. Especially for dos, accurate, recall and F1-score, 0.46, 0.28 and 0.38 improved significantly, reflecting classification accuracy.

Table 3: Classification report for DNN and GAN-DNN.

ClassificationReport						
DNNModel				GAN-DNNModel		
	Recall	Precision	F1-score	Recall	Precision	F1-score
DoS	0.72	0.54	0.62	1	1	1
Fuzzers	0.88	0.86	0.87	0.86	0.76	0.81
Exploits	0.61	0.76	0.68	0.92	0.84	0.88
Normal	1	1	1	0.8	0.99	0.89
Generic	0.96	1	0.98	0.97	1	0.98
Accuracy			0.84			0.91

Source: Authors, (2026).

A high F1-Score indicates that the test dataset was correctly classified into the class. In Figure 7, the DNN and GAN-DNN models' "Precision," "Recall," and "F1-Score" are compared in Figure 7, the DNN and GAN-DNN models' "Precision," "Recall," and "F1-Score" are compared.

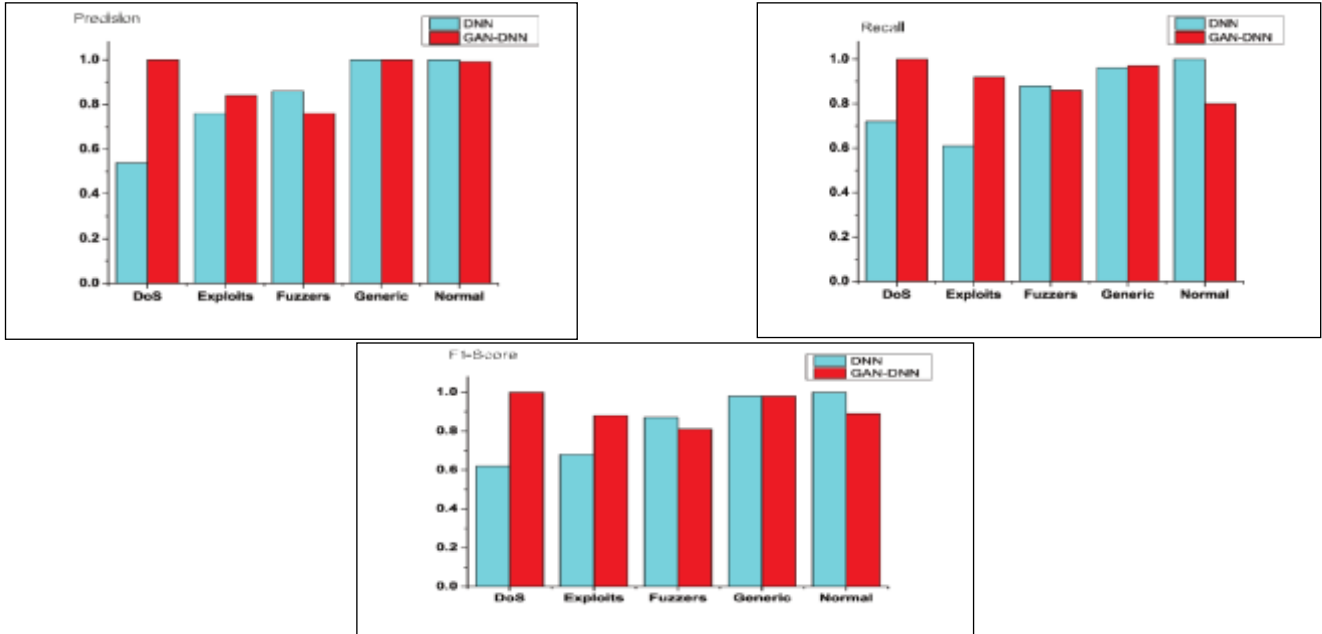


Figure 7: DNN and GAN-DNN Precision, Recall, and F1-Score.

Source: Authors, (2026).

When it comes to classifying normal and attack-categorized packets with 84.4% accuracy achieved from the proposed model. Using the GAN model, it may boost accuracy by increasing the amount of minority attack packets. Figure 8 suggests that by applying a GAN-based model, it gained 91% accuracy with low damage, especially when the DoS attack in the dataset included more packets than those in the DOS attack category. The disadvantage function value, which reflects the prediction error, was minimized by adjusting the model weight during training. As the number of ages increased, training accuracy has steadily improved, while losses declined, indicated effective learning. Similarly, Figure 8 (a) shows the training and verification accuracy of DNN models, where both the decrease gradually grows and changes, reaching 0.844. This overlap indicates that DNN model is well generalized without significant overfitting during training. Figure 8(b) showsthat during the training and validation curve, the loss value dropped to about 0.38 after the 20th epoch. Moreover, Figures 8(c) and 8(d) indicate that the loss further reduced to 0.25 by the 20th epoch, and both training and validation accuracy of the GAN-DNN model began to reach 0.91 and became constant, indicating that the model could not learn much more without overfitting. With the testing dataset, the GAN-DNN model achieved an accuracy of 90.9, which was better than the standalone DNN model, which had 84% accuracy.

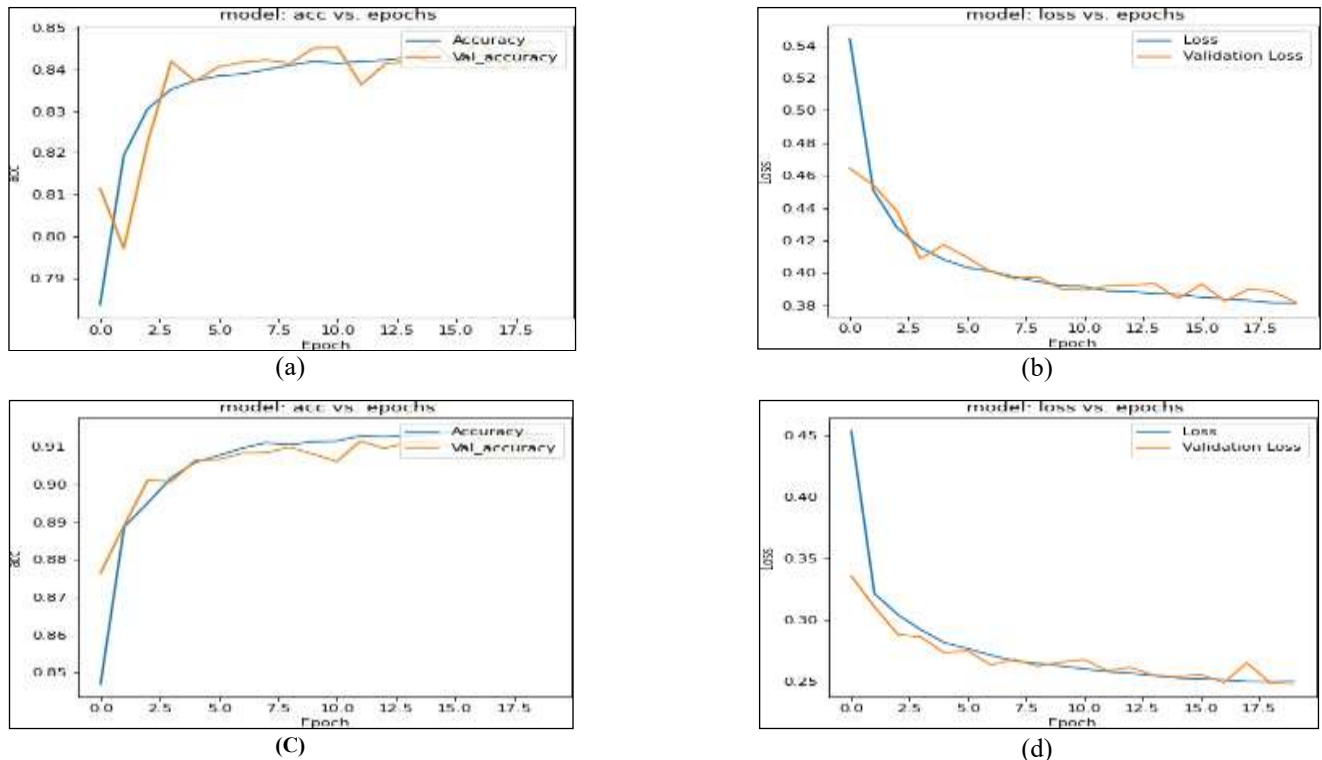


Figure 8: (a, b) Accuracy and Loss vs Epochs for DNN model (c, d) Accuracy and Loss vs. Epochs for GAN-DNN model.

Source: Authors, (2026).

Comparison was also made to the proposed model to the conventional machine learning methods to assess the performance of the methods in the form of accuracy and computational time when used in both binary and multiclass classification. The models that tend to be most popular in terms of the interpretability of the results are traditional models, such as K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF), and Support Vector Machine (SVM). Nevertheless, deep learning methods are typically more useful with large-scale and high-dimensional data and achieve better predictive power. Table 4 summarizes the results of these models being tested on UNSW-NB15 as a multiclass classifier. Preprocessing Data was normalized by removing numerical data and label encoding was performed on the categorical ones. As a way of managing class imbalance, random sampling was conducted where the number of packets was past 25,000 instances. Tools of feature reduction like Pearson correlation coefficient were used to improve the efficiency and accuracy of classification.

Table 4: Comparison results on UNSW-NB dataset.

Model Name	Accuracy
DNN	77.16%
DNN	78%
RNN, LSTM and XGBoost	78.40%
LSTM model	85.8%
Proposed DNN model	84.4%
Proposed GAN-DNN model	90.9%

Source: Authors, (2026).

The SVM model as shown in Figure 9 is an implementation of SVC. In general, the proposed model of the DNN showed the best results, having the highest F1-score, recall, and precision of DoS attack detection, and the highest accuracy of 91%.

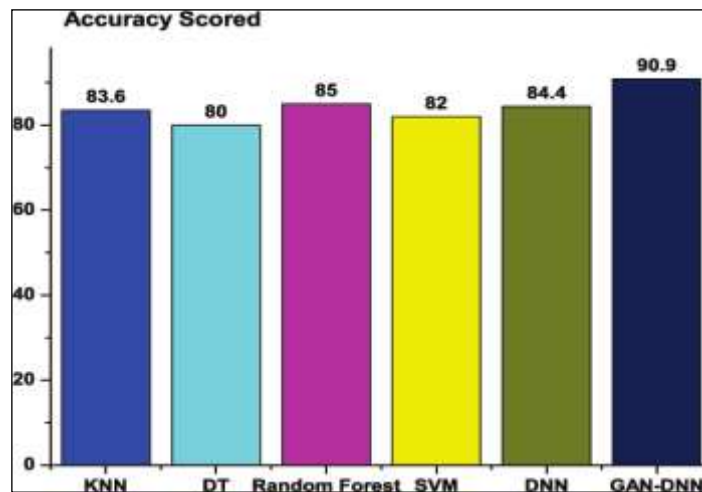


Figure 9: Model Accuracy Comparison.

Source: Authors, (2026).

The statistical significance validation of the results and other parameters. To ensure the reliability of reported results, multiple statistical validation tests were conducted. The 10-fold cross validation gives the average accuracy $90.91\% \pm 0.21$ with McNemar test in comparison to random forest classifier ($\chi^2 = 47.8$, $p < 0.00001$). The paired t-test is evaluated for DNN, $t=6.13$, $p=0.0003$ confirms the performance improvements. The Cohen's Kapps score of the model is $k=0.97$ predicted the agreement labels. The computational cost analysis is conducted in terms of memory usage, training time, number of parameters, and inference speed. There are several components utilized in the model such as 3 numbers of hidden layers, trainable parameters 12928, ReLU activation function, 64 neurons per layer, and Adam optimizer. The training time per epoch is 1.24 sec, average inference time 0.21 ms, and throughput approx. 4760 prediction per second. The memory usage by the model includes 812 MB training peak usage, and inference 100 MB. The computational cost is analyzed for distinct models shown in Table 5.

Table 5: Computational cost analysis of distinct models.

Model	Accuracy	Train Time	Parameters	Inference Speed
Proposed DNN	84.4%	2 min	12.9K	0.21 ms
Proposed GAN-DNN	90.9%	2 min	80K	0.20 ms
CNN	78.82%	6 min	89K	0.45 ms
LSTM	79.11%	13 min	112K	0.82 ms
Random Forest	80.5%	9 min	—	1.3 ms

Source: Authors, (2026).

Table 6 shows the performance comparison of the DNN and GAN-DNN Model for normal Vs attacked data. The proposed GAN-DNN model highlights better performance in terms of precision, recall, and F1-score.

Table 6: Performance Comparison of models.

DNN Model				GAN-DNN Model		
	Precision	Recall	F1-score	Precision	Recall	F1-score
DoS	0.54	0.72	0.62	1	1	1
Exploits	0.76	0.61	0.68	0.84	0.92	0.88
Fuzzers	0.86	0.88	0.87	0.76	0.86	0.81
Generic	1	0.96	0.98	1	0.97	0.98
Normal	1	1	1	0.99	0.8	0.89
Accuracy			0.84			0.91

Source: Authors, (2026).

V. CONCLUSIONS

The current work consists of an integrated and embedded GAN-DNN model that handles the problems with the detection of anomalies in the IoT networks especially with the problems of unbalanced information and the changes in the attack patterns. The proposed approach enhances the diversity of training data and classification accuracy through GAN model data augmentation generation and augmented feature selection enhancement into DNN architecture. Compared to a standalone DNN model, the GAN-DNN model has been shown to be more accurate in general (0.91 vs. 0.84) and has much higher success rates in detecting relevant attack categories, such as DoS and exploits. The fact that the model had a perfect detection when used with specific categories of attacks highlights its capability to be implemented in real time IoT attack detection systems. Moreover, the enhanced recall and F1-scores of the various types of attacks confirm its strength and its ability to adapt to complicated network setups

VI. AUTHOR'S CONTRIBUTION

Conceptualization: Vibhor Harit, Rajeev Dhaiya, Umang Garg.

Methodology: Vibhor Harit and Rajeev Dhaiya.

Investigation: Vibhor Harit and Rajeev Dhaiya, Umag Garg.

Discussion of results: Vibhor Harit, Rajeev Dhaiya, Umang Garg.

Writing – Original Draft: Vibhor Harit.

Writing – Review and Editing: Vibhor Harit.

Resources: Umang Garg.

Supervision: Rajeev Dhaiya, Umang Garg.

Approval of the final text: Vibhor Harit, Rajeev Dhaiya, Umang Garg.

VII. REFERENCES

- [1] T. Zhukabayeva, L. Zholshiyeva, N. Karabayev, S. Khan, and N. Alnazzawi, "Cybersecurity Solutions for Industrial Internet of Things–Edge Computing Integration: Challenges, Threats, and Future Directions," *Sensors*, vol. 25, no. 1, p. 213, Jan. 2025, doi: 10.3390/s25010213.
- [2] M. Rodríguez, D. P. Tobón, and D. Múnera, "A framework for anomaly classification in Industrial Internet of Things systems," *Internet of Things*, vol. 29, p. 101446, Jan. 2025, doi: 10.1016/j.iot.2024.101446.
- [3] N. U. Ain, M. Sardaraz, M. Tahir, M. W. Abo Elsoud, and A. Alourani, "Securing IoT Networks Against DDoS Attacks: A Hybrid Deep Learning Approach," *Sensors*, vol. 25, no. 5, p. 1346, Feb. 2025, doi: 10.3390/s25051346.
- [4] K. Naveeda and S. M. H. S. S. Fathima, "Real-time implementation of IoT-enabled cyberattack detection system in advanced metering infrastructure using machine learning technique," *Electr Eng*, vol. 107, no. 1, pp. 909–928, July 2024, doi: 10.1007/s00202-024-02552-z.
- [5] S. Potharaju, R. K. Tirandasu, S. N. Tambe, D. B. Jadhav, D. A. Kumar, and S. S. Amiripalli, "A two-step machine learning approach for predictive maintenance and anomaly detection in environmental sensor systems," *MethodsX*, vol. 14, p. 103181, June 2025, doi: 10.1016/j.mex.2025.103181.
- [6] B. H. Mohammed et al., "Anomaly Detection of Distributed Denial of Service (DDoS) in IoT Network Using Machine Learning," *Studies in Systems, Decision and Control*. Springer Nature Switzerland, pp. 41–64, 2025. doi: 10.1007/978-3-031-78412-5_3.
- [7] M. Benmalek and A. Seddiki, "Particle swarm optimization-enhanced machine learning and deep learning techniques for Internet of Things intrusion detection," *Data Science and Management*, Feb. 2025, doi: 10.1016/j.dsm.2025.02.005.
- [8] A. Yazdinejad, A. Dehghantanha, G. Srivastava, H. Karimipour, and R. M. Parizi, "Hybrid Privacy Preserving Federated Learning Against Irregular Users in Next-Generation Internet of Things," *Journal of Systems Architecture*, vol. 148, p. 103088, Mar. 2024, doi: 10.1016/j.sysarc.2024.103088.
- [9] S. M. Kasongo and Y. Sun, "A deep learning method with wrapper-based feature extraction for wireless intrusion detection system," *Computers & Security*, vol. 92, p. 101752, May 2020, doi: 10.1016/j.cose.2020.101752.
- [10] G. Bae, S. Jang, M. Kim, and I. Joe, "Autoencoder-Based on Anomaly Detection with Intrusion Scoring for Smart Factory Environments," *Communications in Computer and Information Science*. Springer Singapore, pp. 414–423, 2019. doi: 10.1007/978-981-13-5907-1_44.
- [11] M. A. Ferrag, L. Maglaras, S. Moschyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.
- [12] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep Q-Learning Based Reinforcement Learning Approach for Network Intrusion Detection," *Computers*, vol. 11, no. 3, p. 41, Mar. 2022, doi: 10.3390/computers11030041.

- [13] B. Olanrewaju-George and B. Pranggono, "Federated learning-based intrusion detection system for the internet of things using unsupervised and supervised deep learning models," *Cyber Security and Applications*, vol. 3, p. 100068, Dec. 2025, doi: 10.1016/j.csa.2024.100068.
- [14] Q. A. Al-Haija and A. Droos, "A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things," *Expert Systems*, vol. 42, no. 2, Sept. 2024, doi: 10.1111/exsy.13726.
- [15] M. Ge, N. F. Syed, X. Fu, Z. Baig, and A. Robles-Kelly, "Towards a deep learning-driven intrusion detection approach for Internet of Things," *Computer Networks*, vol. 186, p. 107784, Feb. 2021, doi: 10.1016/j.comnet.2020.107784.
- [16] J. Ahmad et al., "An interpretable deep learning framework for intrusion detection in industrial Internet of Things," *Internet of Things*, vol. 33, p. 101681, Sept. 2025, doi: 10.1016/j.iot.2025.101681.
- [17] Q. Gulzar and K. Mustafa, "Interdisciplinary framework for cyber-attacks and anomaly detection in industrial control systems using deep learning," *Sci Rep*, vol. 15, no. 1, July 2025, doi: 10.1038/s41598-025-89650-5.
- [18] A. Borji, "Pros and cons of GAN evaluation measures," *Computer Vision and Image Understanding*, vol. 179, pp. 41–65, Feb. 2019, doi: 10.1016/j.cviu.2018.10.009.
- [19] M. M. Rahman, F. Bouhafs, S. A. Hoseini, and F. den Hartog, "UNSW HomeNet: A network traffic flow dataset for AI-based smart home device classification," *Computers & Industrial Engineering*, vol. 204, p. 111041, June 2025, doi: 10.1016/j.cie.2025.111041.
- [20] H. Hairani, T. Widiyaningtyas, and D. Dwi Prasetya, "Addressing Class Imbalance of Health Data: A Systematic Literature Review on Modified Synthetic Minority Oversampling Technique (SMOTE) Strategies," *JOIV : Int. J. Inform. Visualization*, vol. 8, no. 3, p. 1310, Sept. 2024, doi: 10.62527/joiv.8.3.2283.
- [21] A. Maniatopoulos and N. Mitianoudis, "Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function," *Information*, vol. 12, no. 12, p. 513, Dec. 2021, doi: 10.3390/info12120513.
- [22] Ilham Kachbal, Imrane Errafi, Mousaab EL Harmali, Said El Abdellaoui, and Khadija Arhid, "YOLO-GARNet: A High-Quality Deep Learning System for Garment Analysis and Personalized Fashion Recommendation," *Engineering Letters*, vol. 33, no. 11, pp4448-4462, 2025.
- [23] Chom Panta, Yupawadee Samranrit, Jiraroj Tosasukul, Anupong Sukprasert, and Aweeporn Panthong, "Machine Learning-Based Forecasting of Water Levels in the Chao Phraya River, Thailand," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 11, pp3685-3692, 2025.
- [24] Changxi Ma, Xiaoyu Huang, and Bo Du, "Traffic Speed Prediction with Congestion Data: An ISGA-Optimized Hybrid Deep Learning Model," *IAENG International Journal of Applied Mathematics*, vol. 55, no. 11, pp3743-3753, 2025.
- [25] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," 2015 Military Communications and Information Systems Conference (MilCIS). IEEE, pp. 1–6, Nov. 2015. doi: 10.1109/milcis.2015.7348942.