International Journal of Industrial Engineering, 32(1), 67-92, 2025

DCNN-BIGRU: A PROFICIENT HYBRID CLASSIFIER FOR RELIABLE INTRUSION DETECTION AND PREVENTION: HYBRID APPROACH

Neeraj Sharma[1,*] and Neelu Nihalani[2]

[1]Department of Computer Science & Engineering
University Institute of Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya
Madhya Pradesh, India
[*]Corresponding author's e-mail: neerajsharmans12@gmail.com

[2]Department of Computer Applications
University Institute of Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya
Madhya Pradesh, India

Advances in networking devices have revolutionized many industries by enabling intercommunication and automation in multiple areas, such as healthcare, transportation, and manufacturing. However, the threat of cyber-attacks has also escalated with the increased connectivity and dependency on these devices. Cyber security has become critical in protecting networks from malicious activities, ensuring the privacy and integrity of the data transmitted. Multiple deep-learning methods face multiple challenges in identifying intrusion threats; however, deep learning can self-enhance and scale up for reliability. We propose an efficient hybrid deep-learning intrusion-detection classifier, DCNN-BiGRU. The classifier has a simple architecture and works well in environments that do not require saving long-term dependencies and where computational resources are limited. It achieved a multiclass-classification accuracy of 99.70% on the training and test datasets.

Keywords: Cyber Security, Intrusion Detection System, Hybrid, Anomaly Detection, Machine Learning, Network Attacks.

(Received on May 28, 2024; Accepted on October 5, 2024)

1. INTRODUCTION

Networks have reached millions of interconnected data-transmission devices or pieces of equipment that convey massive quantities of data. They offer a lot of advantages and benefits; however, they are susceptible to increased risk of fraud because they can be penetrated, enabling hackers to access information and data communicated through the networks. Intrusion detection systems (IDSs) are crucial in detecting, stopping, and preventing breaches in inter-networking networks (Khraisat *et al.*, 2019). Traditional intrusion-detection methods fail to effectively detect intrusion over networks (i.e., signature-based IDS methods are inefficient in evaluating attack patterns such as signature-based detection) and may not sufficiently address the sophisticated and evolving nature of network attacks. Therefore, advanced and efficient IDS models are needed (Kumar *et al.*, 2022).

Efficient intrusion detection techniques are an important requirement for any network IDS (NIDS) to effectively detect and prevent anomalies. Most NIDS models use signature-based techniques instead of anomaly-based techniques to prevent malicious attempts. Hackers employ new and complex methods; IDSs must be capable of self-learning and self-predicting to deal with malicious attempts. Thus, IDSs need continuous refinements to offer effective protection. Existing IDSs are continuously refined by upgrading and releasing more stable versions. Intrusion detection works on multiple paradigms to secure networks against malicious activities, with multiple theories being employed, such as fuzzy-based clustering or pattern-recognition techniques (Gosain *et al.*, 2022). Enhanced fuzzy clustering and deep-learning techniques (Altameem *et al.*, 2022) have been employed in behavior analysis to develop new frameworks for intrusion detection. Neural networks can learn hidden patterns and identify the behavior and relationships that help in prediction. Thus, the most advanced and successful frameworks based on artificial intelligence use deep-learning-based classifiers. In related studies, deep-learning models performed well over large datasets. However, in some scenarios, feature- or dimension-reduction techniques also help IDSs improve their efficiency by removing or transforming unusual attributes. Binary classification is the paradigm most adopted by multiple traditional IDS techniques to categorize the threat type and pattern. High-dimensional data impacts the performance of machine-learning (ML) classifiers by degrading their efficiency and

reducing their accuracy. This results in a low attack-detection rate in the training and validation model, which hampers the detection capability and performance.

Traditional intrusion-detection methods are not that capable as they use key-based transmission or other encryption and decryption methods to transfer data packets. Some of the most common algorithms are RSA and Diffie hellmen. On the other hand, ML techniques have outperformed traditional intrusion-detection methods. This paper explores previous research on feature selection in intrusion detection. Dimension-reduction techniques have been used in multiple classification scenarios to improve the efficiency of IDSs.

A model should be capable of selecting reliable features from high-dimensional datasets such that its classification accuracy is not affected. Figure 1 and Figure 6 shows the proposed model and how multiple individual classifiers are placed in a hybrid classifier framework, together with the order of data processing and flow. To understand the functioning of various network intrusion detection frameworks, we review various IDS approaches. Our study is based on convolutional neural network and bidirectional long short-term memory CNN-BiLSTM hybrid neural networks with attention mechanism (Shan *et al*., 2021; Gao *et al.*, 2021; Altunay *et al*., 2023; Zhang *et al.*, 2023), which we enhanced to develop a secure IDS. The proposed classifier is known as DCNN-BiGRU. Motivated by the hybrid nature of ML classifiers, we developed a framework that can effectively perform well in systems that are based on multiclass classification with higher accuracy. A major limitation of multiple classifiers is the decrease in efficiency in multiclass-based classification. In the proposed framework, we used a bidirectional variant of recurrent neural networks (RNN), which can save dependency patterns for data evaluation with limited resources.

The contributions of our study are described as follows.

- We develop a hybrid classifier by combining and blending convolutional neural network (CNN) and bidirectional gated recurrent unit (BiGRU) classifier units with dense neural network (DNN) layers to build a reliable IDS.
- The proposed classifier works well in the case where it is not required to save long-term dependencies for modeling sequential data, particularly in cases where computational resources are limited and a simple architecture is desired using a hybrid-classifier-based approach. It also addresses the gradient vanishing faced by traditional RNNs. It has less training time for gated recurrent unit (GRU) cells owing to simple architecture, as two gates and a candidate activation vector are used instead of individual memory cell units.
- The performance of the model is evaluated on a real-time traffic dataset, CICIDS2019, and the results are compared with those of other hybrid classifiers.

Manual intervention to remove less meaningful attributes results in an increase in the efficiency of the model.

The remainder of this article is structured as follows: Section 2 explains the novelty and contributions of the study. Related studies are reviewed in Section 3. Section 4 introduces the background of the models used for our analysis. The proposed model is presented in Section 5. Section 6 presents the experimental analyses and comparison with other models or classifiers. We evaluate the experimental outcomes and results and also present a comparative study with alternative intrusion-detection techniques. Section 7 concludes the paper.

2. MOTIVATION & CONTRIBUTION

(i) BiGRU units can re-iterate themselves using weight and enhance performance. GRU has a simpler architecture than long short-term memory (LSTM) cells as it performs calculations using only two gates (i.e., reset gate and update gate), whereas LSTM cells use three gates for calculations. Consequently, GRU cells require less computational resources and training time, making them computationally effective and simple.

(ii) GRU is a form of RNN; it effectively handles the vanishing gradient problem faced by traditional RNNs. In the calculations that require simple architectures, GRU is a good alternative to LSTM networks. GRU uses a single gate to control information flow in contrast to the three used by LSTM, which makes it faster to train and execute. However, GRU cannot save and analyze long-term dependencies; it may perform well in cases that require quick learning and adapting to new inputs and not in scenarios where networks require saving long-term dependencies. GRU. LSTM and GRU are types of RNN; they save information from previous inputs, which helps utilize dependencies and context observed between time steps.

(iii) GRU is used to resolve the problems that arise due to vanishing gradient (i.e., conditions when the weight threshold becomes so negligible, impacting network learning capability) in simple RNNs. GRU cells contain memory components capable of remembering information for longer periods. This is achieved using gates that control the flow of information in and out of a cell. The sigmoid activation function controls these gates using values that range from 0 to 1. The gates can

enrich the model by allowing it to selectively store or forget the information as per the input information and the previous cell state.

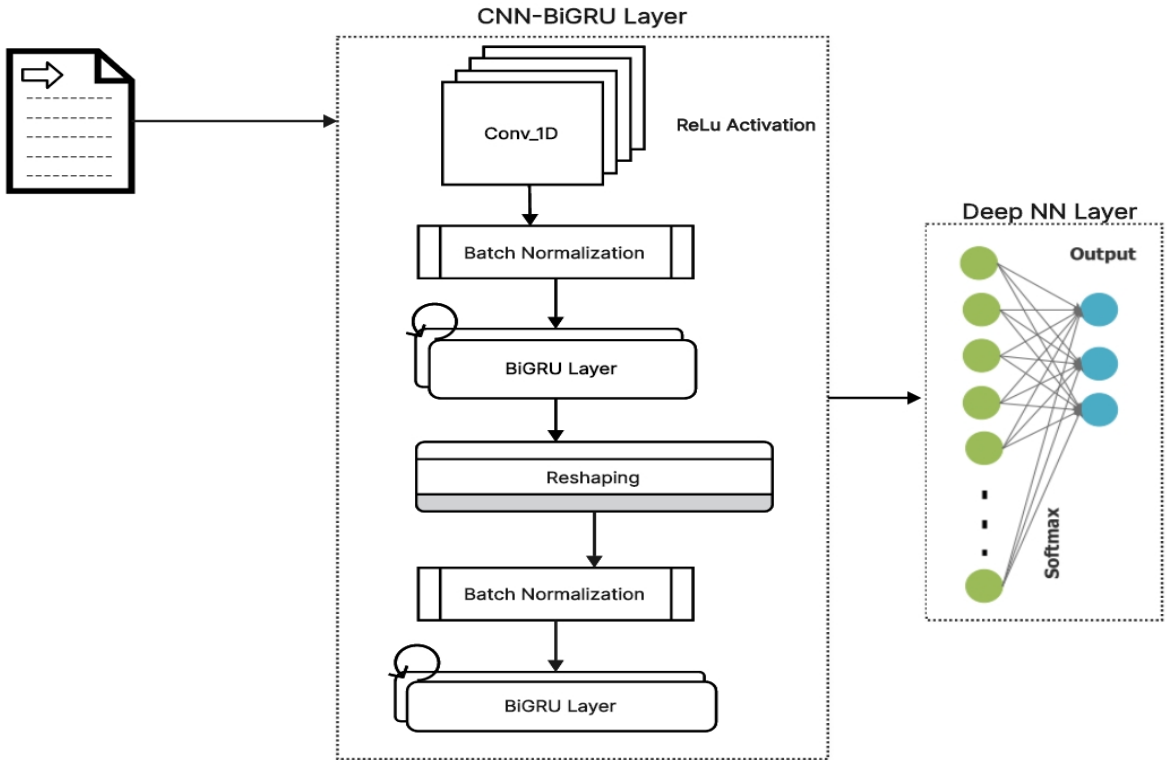


Figure 1. DCNN-BiGRU Hybrid Approach

We blend the DCNN-BiGRU model in a hybrid manner to inherit the properties of the best classifier network. The model is suitable for multiclass label scenarios and can adjust, enhance, and scale up. It is a dense form of CNN (i.e., DCNN) merged with BiGRU with minimized metrics that replicate in performance enrichment.

3. RELATED STUDIES

We propose a hybrid framework that merges the capabilities of multiple ML algorithms to create an accurate and precise network intrusion detection system. The proposed model leverages the strengths of individual algorithms to achieve an enhanced detection performance. The framework consists of three major steps: preprocessing the given data, selecting features of the required attributes, and training the hybrid algorithm. The deep-learning mechanism plays a very crucial role in identifying patterns in complex, emerging datasets with complicated representations. The hybrid model contains multiple IDS techniques to alleviate the impact of false positives or negatives during the prediction mechanism, which helps minimize the impact of risks that may be generated or faced in the case of individual model scenarios.

Deep learning (DL) plays a significant role in multiple disciplinary fields like healthcare (Shamshirband *et al.*, 2021), manufacturing, and security (Sarker *et al*., 2021). In cyber security, Van Huong *et al*. (2019) suggested a model for logging the log information and details of users of an Internet of Things (IoT) system, including address, location, and services, into a dataset. After preprocessing and cleaning, the dataset was transformed into an image-like sparse matrix and used to train a CNN classifier, which achieved an average accuracy of 98.9%. Wu *et al*. (2018) proposed a novel method that uses a CNN-based framework to detect network intrusion. The CNN model tended to improve accuracy even when the class size was small and reduced the false alarm rate. Gurung *et al.* (2019) detected intrusions on an NSL-KDD dataset using a DNN network, achieving 87.2 % accuracy; they aimed to lower the false alarm rate to a minimum threshold using the framework.

With the widespread use of IoT and Android frameworks, the risk of information being compromised has increased; the information flows between multiple channels and various malicious attempts to access the confidential information are encountered. A survey of malware detection frameworks was performed by Smmarwar *et al*. (2024) using deep-learning techniques.

Tsogbaatar *et al.* (2021) proposed a 'DeL-IoT' intrusion-detection framework based on the principle of software-defined networking. The meaningful features were determined using deep and stacked, layered autoencoders. The proposed framework achieved a high detection rate and accuracy in classifying malicious attack attempts, with 99.5- 99.9% in the F-score and an accuracy of 91.04%-99.95%. Ullah *et al.* (2021) proposed a model that incorporates CNN1D, CNN2D, and CNN3D dense learning for anomaly detection on datasets based on IoT networks, with CNN as the core component of the model. The proposed IDS framework was evaluated on several IoT-based IDS datasets, such as BoT-IoT, IoT-DS-2, IoT-23, and MQTT-IoT-IDS2020. The accuracy of the model was observed in multiclass-classification-based scenarios (i.e., cases where datasets contain various attack labels).

Currently, malicious content writers create AI-driven malicious codes to bypass security checks—their nature is difficult to guess and identify. To counter this, hybrid classifiers based on explainable artificial intelligence (XAI) have been proposed, which inherit the capabilities of CNN and BiGRU networks. Smmarwar *et al.* (2023b) proposed XAI-AMD-DL and tested its performance on the CICAndMal2019 Android malware dataset.

An Android malware-detection framework was proposed by Smmarwar *et al.* (2022c), where data preprocessing was performed owing to the complex nature of datasets and feature selection was done using discrete wavelet transform. Further, a light-weighted CNN version was applied on the optimized dataset for classification purposes. The overall framework was composed of three major phases: discrete wavelet transform was used as a feature selector, a generative adversarial network was formed, and further classification was done in the third phase.

Roopak *et al.* (2020) proposed a hybrid intrusion-detection model consisting of CNN and LSTM. The model performance was tested on the CISIDS2017 dataset. A non-dominated sorting genetic algorithm was used for feature selection and simple dimension reduction. A model for anomaly identification using a vector convolutional DL technique was proposed by Amma *et al.* (2020). It achieved 99.7496% accuracy in multiclass classification on a BOT-IOT dataset. Popoola *et al.* (2021) proposed a hybrid LAE-BLSTM model for BOT-IOT datasets, achieving an accuracy of 91.89 %, with a reduction in dataset size.

Bhatt *et al.* (2018) proposed a hybrid anomaly-detection model based on DL and a decision module, achieving an accuracy of 98%. The model consists of four major components: isolation forest, self-organizing map, one-class support vector machines, and Gaussian mixture modeling. A multi-layer perceptron model was developed (Reddy *et al*., 2022) for intrusion detection and prevention. Its multiclassification performance was evaluated on NSL-KDD and wireless sensor network datasets. Alazab *et al.* (2023) built an IDS model using DL and the optimization technique used is the Harris Hawks optimization algorithm, which works by adjusting weight as well as bias sceneries.

Chandrasekar *et al.* (2023) proposed an ML classifier to predict heart diseases. It employs deep neural networks and convolution networks, and the data was collected from various sources and fed to public healthcare datasets. A DNN was used for feature selection, and fifteen convolution layers were used for prediction, achieving an accuracy of 95.46%.

Smmarwar *et al.* (2021) proposed a malware signature detection framework to identify data behavior from byte-level or assembly-level data by converting them into visual format and then classifying using a novel triple-fused CNN, which identifies outliers that represent abnormal patterns in the data. The proposed framework achieved an accuracy of 98% on the Microsoft Malware dataset. Ghani *et al.* (2023) proposed a DL intrusion-detection framework based on a feedforward neural network, which was evaluated over a small feature set on UNSW-NB15 and NSL-KDD datasets.

Another deep-learning classification framework was proposed by Aljuaid *et al.* (2024). The feature selection was performed in the first stage, the classification of the dataset was performed using a CNN as the base classifier, and analysis was performed over the cloud computing domain. The Pearson correlation coefficient was used for feature selection and optimization. Farhan *et al.* (2020) proposed a deep-learning-based classifier where data was preprocessed using binary particle swarm optimization before classification.

A cloud-based intrusion detection model was proposed by Attou *et al.* (2023). A random forest classifier was used for feature selection and classification owing to the complex nature of the cloud environment; the model achieved good accuracy. Jun-Ho Park *et al.* (2024) used time series data collected from the vibration of a blower motor—used in industry to maintain operational efficiency and maintenance regularization—to detect faults. They used ML classifiers in the first phase and an extreme gradient-boosting algorithm in the second phase.

Chen *et al.* (2023) proposed a fusion graph convolution network to detect faults in sensors used in wireless sensor networks. The sensors got faulty due to their hardware, complex working environments, or other attributes. The graph convolution network was merged with the ant colony optimization to predict faulty sensors in wireless sensor networks and enhance their durability and stability.

A hybrid classifier (Qazi *et al*., 2023) that combines RNN and CNN was proposed. Its efficiency was calculated on a network's real-time traffic. A 1D CNN layer was used as a feature selector, followed by a CNN-deep-learning classifier layer, and four RNN layers were applied to the data. The model works well with data with spatial and temporal dependencies. The four layers help improve efficiency, as demonstrated by the performance evaluation metrics. The overall objective was to find the best parameters to improve the learning efficiency of the classifier, minimizing the error ratio. In

our proposed study, the deep CNN (DCNN) fusion was done on the bidirectional GRU (i.e., a specification of RNN to increase efficiency). Table 1 shows the comparison of our proposed work with multiple techniques in the literature.

Table 1. Summary of Significant Related Studies on Intrusion Detection Frameworks

Paper	Learning Algorithms	Dataset	Feature Selection/ Optimization	Model Description	Accuracy Classification
Smmarwar *et al*. (2022a)	OEL-AMD	CICInvesAnd Mal2019	Binary Grey Wolf Optimization	Optimized Ensemble Learning-based Android malware detection and classification model	96.95%(M) and 83.49%(B)
Ghani *et al*. (2023)	Feed Forward neural network classifiers	UNSW-NB15 and NSL-KDD	the small feature vector is being selected to evaluate the performance	A machine learning-based Feedforward neural network classifier is being used	90.11%(B) and 91.21%(B)
Aljuaid *et al*. (2024)	CNN	CSE-CICIDS2018	Pearson correlation coefficient matrix heatmap	A Deep Learning Intrusion detection system for covering cyber-attacks in the cloud computing domain	98.67%(M)
Farhan *et al*. (2020)	DNN	CICIDS2018	binary particle swarm optimization (BPSO)	An Optimized Deep Learning Based Framework for Intrusion Detection	95%(B)
Ahmed *et al*. (2023)	Hybrid CNN-LSTM	NSL-KDD	CNN (Conv1D)	A Hybrid classifier based on Deep Learning by combining CNN and LSTM	99.20%(M)
Smmarwar *et al*. (2022b)	RF, Decision Tree, SVM RBF	CIC-InvesAndMal2019 dataset	Wrapping feature selection by combining GreedySW and Random Forest	The performance of multiple ML-based frameworks has been evaluated on the reduced feature and optimized dataset	91.80% (SVM RBF) achieves the highest accuracy
Qazi *et al*. (2023)	Hybrid RNN and CNN	CICID2018	CNN Feature extractor layer	A Hybrid Deep Learning base NIDS framework	98.90%(M)
Smmarwar *et al*. (2023a)	Hybrid CNN-LSTM	IoT malware, Microsoft BIG-2015, and Malimg	Double-Density Discrete Wavelet Transform (D3WT)	IoT-based IDS for cyber threats that is AI empowered	96.97%, 99.98%, 99.96% (Accuracy achieved for datasets)
Attou *et al*. (2023)	Random Forest (RF)	NSL-KDD, Bot-IoT	graphic data visualization &RF feature selection	A Cloud-Based Intrusion detection and prevention framework	98.3% and 98.99% (for the datasets)
Smmarwar *et al*. (2022c)	Light weight CNN	IoT malware, Mailing	Discrete Wavelet Transform (DWT)	Three-phase deep Learning framework for bug detection in smart Agriculture system based on IoT	99% accuracy achieved for datasets
Grace *et al*. (2022)	Hybrid LSTM-SVM	CIC-AndMal-2017 dataset	Aquila optimizer has been used for the feature optimization	A hybrid Intrusion detection framework has been proposed to identify malicious patterns	97% accuracy achieved for datasets
Albakri *et al*. (2023)	Adamax optimizer with attention	Andro-AutoPsy	Rock Hyrax Swarm Optimization-based feature subset	The proposed framework works on the principle of deep learning for the	99.05% accuracy achieved for datasets

Paper	Learning Algorithms	Dataset	Feature Selection/ Optimization	Model Description	Accuracy Classification
	recurrent autoencoder (ARAE)		selection	classification of cyber security and malware detection attempts	

4. BACKGROUND OF THE MODELS

4.1 CNN-based IDS

The main objective of CNNs is to determine the relevant characteristics of the incoming stream of data. The simple representation of a CNN model used as a simulated model for comparison is shown in Figure 2. Multiple learnable filters are applied to a layer that consists of multiple feature extractors, which make up the first layers. The filters use the principle of sliding windows, which have been applied to each incoming data flow point. The output is denoted as feature maps, and the overlying distance is called the stride. Each CNN layer can be considered as a collection of convolutional kernels, which are used to create multiple feature maps. An individual neuron in the feature map of the succeeding layer is associated with the adjacent neuron regions. To build the feature map using CNN, the kernel is made available across complete spatial locations of the provided input and once the polling and convolution layer are set up, single or multiple fully connected layers will be used to complete the classification task.

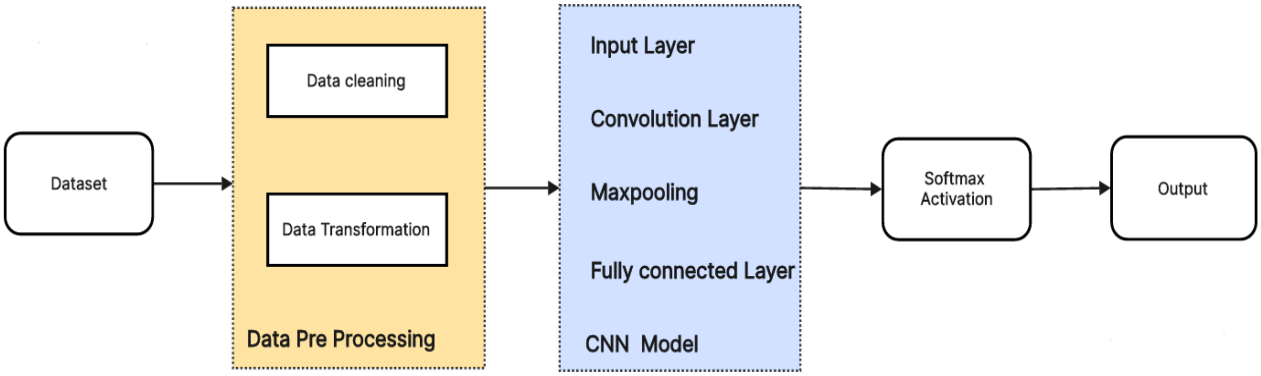


Figure 2. CNN Model Flow

The convolution layer is the key building block of the convolutional network used for feature extraction. The convolution layers perform both linear and nonlinear operations, that is, a convolution operation as well as an activation function (Yamashita *et al*. 2018). Additionally, convolution operations apply filters (or kernels) to protect the spatial relationship between pixels using learned features.

4.2 RNN-based IDS

RNNs are classified as recurrent because they complete the same task of an element in a sequence with the output that has been obtained by previous calculations. The RNN information flow and functioning are bidirectional; further, the next iteration of the input is reprocessed from the RNN network output. A feedforward RNN network is a combined unit of an input layer, a few intermediate hidden layers, and an output layer. The weight matrix is applied to the input, and applying the activation function to the derived result generates the output of a simple RNN network node.

A simple representation of GRU can be seen in Figure 3. In the RNN network (Nikolov *et al*., 2018), a backpropagation algorithm is used for network training. This requires calculating gradients for every weight in the neural network and adjusting each weight value such that the required output can be obtained. In our study, base learners (base classifiers) are the major building blocks of the hybrid framework, which leverages the capability of individual classifiers into one major hybrid classifier with great capabilities.

GRUs are a form of RNNs commonly denoted as GRU networks. Both types of RNN (GRU and LSTM) are similar in function and classification procedures; they both have nearly similar architectures, and GRUs work in a similar way as LSTMs, performing sequential or linear data modeling in a such a way that the information is stored as per some retention threshold and recalled over the time and will be forgotten after some time. GRU offers a simple structure compared with

LSTM, which makes it easier to train with greater computational efficiency using fewer parameters and resources. LSTM saves the long-term dependencies among data, which helps improve predictions. The data is saved using vectors called memory cell state and hidden state: the memory cell state contains information that is retained for a longer duration, whereas the hidden state provides information at the current time step for making predictions that will be further sent to the next timestamp steps. LSTM uses three gates, while GRU architecture uses two gates.

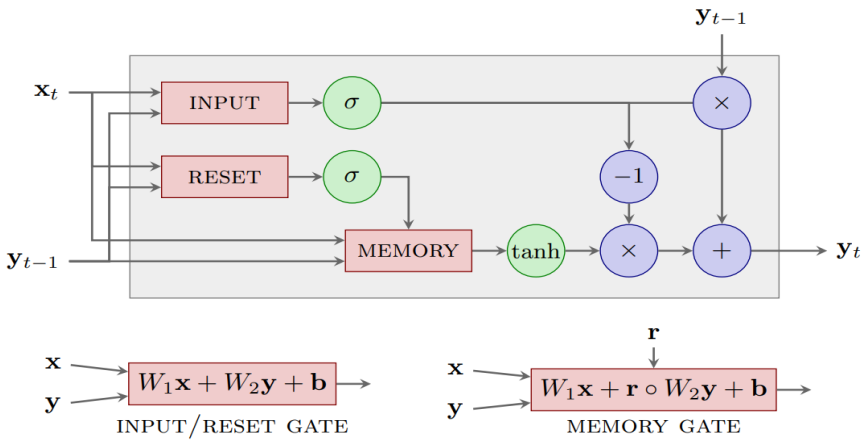
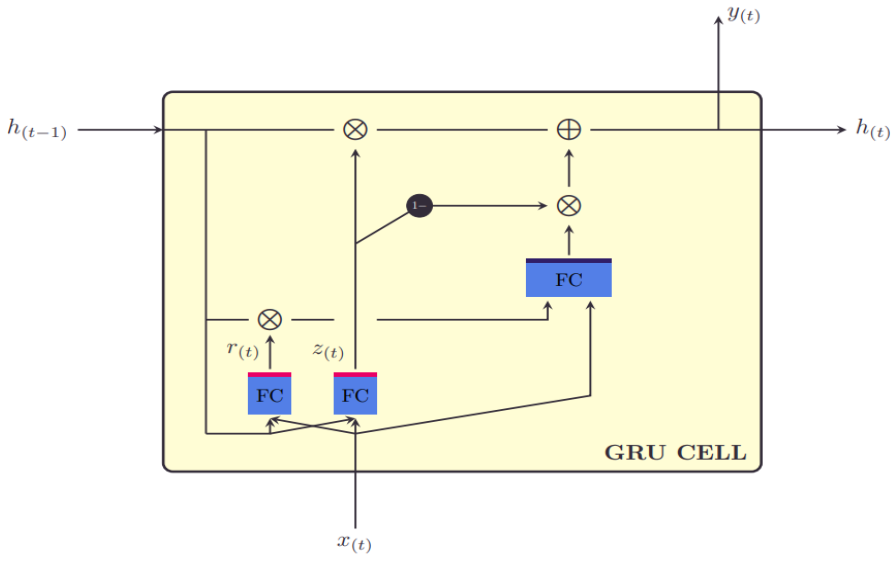


Figure 3. Gated Recurrent Unit

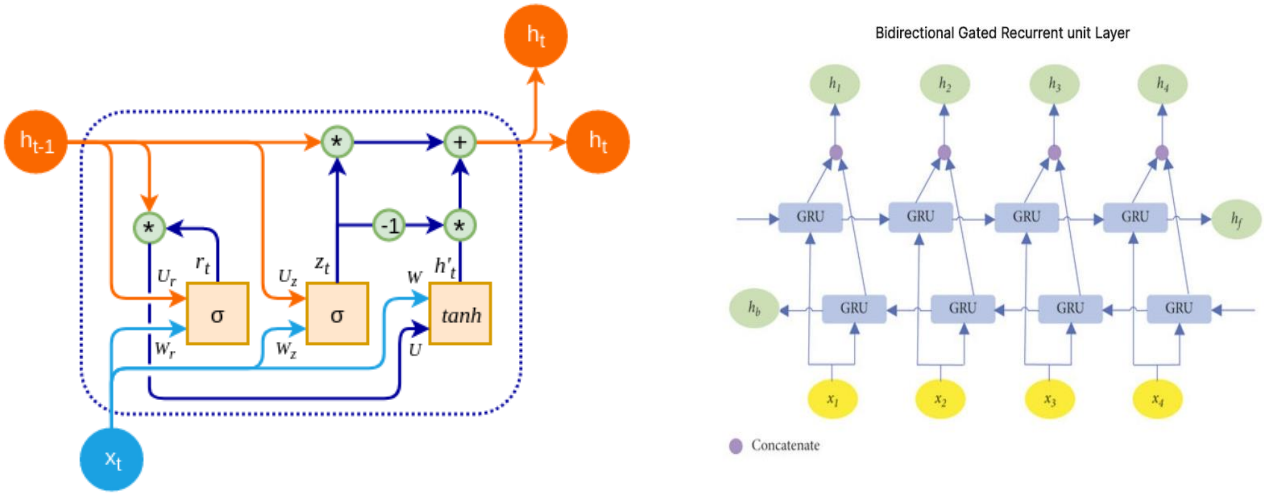
4.3 GRU Architecture

Every GRU cell consists of a reset gate and an update gate. A hidden cell-state concept is used in GRU instead of a separate cell state used in LSTM. A hidden cell state at timestamp '*t*' is denoted as h_t.

The overall information flow in a cell is regulated using a reset gate and update gate, which also helps address the problem of vanishing gradients faced by traditional RNNs. The reset gates in GRU cells evaluate the information carried that is not useful and discard it or not consider it. This is done by either examining the previous hidden state vector and deciding the part to ignore or to reset again. The update gate is responsible for identifying new information vectors to be considered or to be retained from the previous hidden state vector. At timestamp '*t*', we have four major components that impact the working of the GRU cell: the reset gate, update gate, previously hidden state vectors, candidate or new hidden state vectors, and the input data provided at every timestamp.



(a)



(b)

Figure 4. The internal architecture of a GRU cell; (a) GRU Cell; (b) BiGRU Single Layer

Figure 4 shows the internal architecture of a GRU cell: how the vectors flow and are added and multiplied to calculate new variables to perform reset and update operations at every timestamp.

To evaluate the information flow of the above GRU cell, the equation below represents the overall working of a single cell unit.

i. Reset Gate: The reset gate component is responsible for evaluating the previous information that is not necessary to be carried further for the next timestamps and to be discarded or forgotten. At timestamp t, the calculation done by the reset gate can be evaluated as below.

$$R_t = \sigma\,(W_r \cdot [h_{t-1}, x_t] + b_r) \tag{1}$$

where

R_t : The reset gate vector at timestamp t, x_t is the input variable.
W_r : The learning weighted matrix vector maintained at a timestamp by reset gate.
$[h_{t-1}, x_t]$: The mathematical expression used to represent the concatenation between the current input x_t and the received previous hidden state vector h_{t-1}, which is why the time stamp is being represented by the notation t-1 (it is showing the previous timestamp value).
b_r : The bias component used by the reset gate to adjust values accordingly.
$W_r . [h_{t-1}, x_t]$: The dot product of reset gate weighted vectors and the previous hidden state.
σ : The sigmoid function used as an activation function whose value fluctuates in the range of 0–1.

ii. Update Gate: The update gate component is responsible for identifying the important previous hidden state information from past timestamps to be retained, and the new information from the current timestamp is to be incorporated. It carries past meaningful information and updates important information in place of less meaningful data.

$$Z_t = \sigma\,(W_z \cdot [h_{t-1}, x_t] + b_z)\ , \tag{2}$$

where

Z_t : The update gate vector at timestamp t, x_t is the input variable.
W_Z : The learning weighted matrix vector maintained at a timestamp by the update gate.
$[h_{t-1}, x_t]$: The mathematical expression used to represent the concatenation performed between the current input x_t and the received previous hidden state vector h_{t-1}.
b_z : The bias component used by the update gate to adjust values accordingly.
$W_z . [h_{t-1}, x_t]$: The dot product of update gate weighted vectors and the previous hidden state; σ is the sigmoid function

iii. **Candidate hidden state**: It is calculated by finding the new capable hidden state vector $\bar{h_t}$. A reset gate is used to examine the values to forget from the previous hidden state.

$$\bar{h_t} = \textbf{tanh}\ (W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \tag{3}$$

where

tanh : The hyperbolic tangent activation function.
$r_t \odot h_{t-1}$: The element-wise multiplication of the reset gate and previous hidden state; $\odot$ represents the Hadamard product.
W_h : The learning weighted matrix vector maintained at a timestamp by the candidate's hidden state.
b_h : The bias component used by the candidate's hidden state to adjust values accordingly.

iv. Final hidden state: It is denoted as a vector h_t . The update gate is used to examine the values; it is a combination of the previous hidden state vector h_{t-1} and candidate hidden state vectors $\bar{h_t}$.

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \bar{h_t} \tag{4}$$

where

tanh : The hyperbolic tangent activation function
$z_t \odot h_{t-1}$: The element-wise multiplication of the update gate and previous hidden state; $\odot$ is the Hadamard product, which shows all hidden states to be retained as per the update gate.
$(1 - z_t) \odot \bar{h_t}$: The hidden state of the candidate.

The GRU architecture consists of the following components:

i. *Input layer:* This layer acts as an interface to input the data to be evaluated or processed.
ii. *Hidden layer:* It is responsible for recurrent operations. At the current timestamp, the new hidden state is evaluated using the input data and the previous hidden state. The hidden state is the memory component of RNN cells.
iii. *Reset gate:* It is responsible for eliminating or forgetting less meaningful hidden state variables. It performs calculations on the current input and previous hidden state and generates the vectors in the range of 0–1. It controls the degree to which the previous hidden state vectors reset a current timestamp.
iv. *Update gate:* It finds the candidate activation vectors in the range of 0–1 that need to be incorporated into the new hidden state. Its mathematical notation can be seen in Equation 2.
v. *Candidate activation vector:* It is the combination of the current timestamp input and the reset version of the previous hidden state. The tanh activation function is used for the calculation (Equation 3).
vi. *Output layer:* The final hidden state vectors evaluated can be in the output layer, as mathematically represented by Equation 4.

Now consider $\bar{y_t}$ as the prediction output of the model, and the actual output is y_t . Then, the error can be calculated using the formula below at timestamp '*t*':

$$E_t = - y_t \log(\bar{y_t}) \tag{5}$$

The total error summation is calculated for all timestamps $E_t = \sum_t E_t$, $E_t = \sum_t - y_t \log(\bar{y_t})$

The summation of gradients for all timestamps is calculated by the formula.

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial Et}{\partial W} \tag{6}$$

Using the Chain rule. $\bar{y_t}$ can be calculated as a function based on h_t and in parallel; h_t is calculated based on $\bar{h_t}$. On the above basis, the chain relationship can be expressed as below.

$$\frac{\partial Et}{\partial W} = \frac{\partial Et}{\partial \overline{y_t}} \frac{\partial \overline{y_t}}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots\dots\dots\dots\dots\dots \frac{\partial h_0}{\partial W} \tag{7}$$

The total error observed due to gradients can be calculated by the summation of differential vectors:

$$\frac{\partial Et}{\partial W} = \sum \frac{\partial Et}{\partial \overline{y_t}} \frac{\partial \overline{y_t}}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots\dots\dots\dots\dots\dots \frac{\partial h_0}{\partial W} \tag{8}$$

Now, putting the value of h_t final hidden state vector from Equation 4 in the chain of derivative expression $\frac{\partial h_t}{\partial h_{t-1}}$, the total gradient loss due to the hidden state will be calculated as below.

$$\frac{\partial h_t}{\partial h_{t-1}} = z + (1 - z) \frac{\partial \overline{h_t}}{\partial h_{t-1}} \tag{9}$$

Substituting the expression of $\overline{h_t}$ hidden state vector from *E*quation 3 in the expression $\frac{\partial \overline{h_t}}{\partial h_{t-1}}$, the total gradient loss error for the final state will be calculated as below.

$$\frac{\partial \overline{h_t}}{\partial h_{t-1}} = \frac{\partial \, \boldsymbol{tanh} \, (W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h)}{\partial h_{t-1}} \tag{10}$$

Now Equations 3 and 4 will be kept on the above variables to minimize the gradient descent loss. The sigmoid function is used in the GRU cell by both the reset and update gates; thus, it can either take a value of 0 or 1. Now consider the conditions below per reset gate (r) and update gate vector (z):

Condition 1: if z=1, then irrespective of the value of r, the expression $\frac{\partial \overline{h_t}}{\partial h_{t-1}}$, will evaluate to be z only when I equal 1.

Condition 2: (r=0,z=0) when both sigmoidal values are 0, then the expression $\frac{\partial \overline{h_t}}{\partial h_{t-1}}$ evaluate*s* to 0.

Condition 3: (z=0, r=1), the value evaluates to close to 1; in this way, the vanishing gradient will be addressed such that it will be prevented from being negligible in the threshold.

This study focuses on building a hybrid framework for intrusion detection to be applied to datasets, training of models, and validation of accuracy. The loss incurred is calculated, and the proposed model is compared with other ML models. Many studies provide solutions based on various ML models; however, very few employ the concept of a hybrid classifier in intrusion detection. In our study, we used commonly used datasets. This study advances our knowledge of network intrusion detection and offers valuable data for the creation of DL models for IDSs in the future.

5. PROPOSED METHOD

We used a hybrid model composed of CNN, DNN, and BiGRU, individual classifiers with different scopes and domains of classification tasks. Our model is highly influenced by the hybrid combination of classifiers, that is, CNN-BiLSTM (LSTM is a form of RNN). Thus, we propose an approach for IDS by developing a hybrid framework, assembling a CNN layer with a more dense form using DNN layers followed by CNN layers (i.e., DCNN). The two major blocks, DCNN and BiGRU, were merged. Therefore, the hybrid model was composed of CNN (for feature enforcement) BiGRU layers, followed by DNN layers (to reduce the error and loss incurred). The setup was run on Python 3.8. The efficiency and results were compared with those of other methods. In addition to the proposed hybrid model, we mutated and calculated the efficiency of individual classifiers—CNN, DNN, the autoencoder (AE), and the decoder. The proposed model achieved the best accuracy, as shown in Table 8. The overall study consisted of the following steps.

a) Selecting the dataset for the study.
b) Transforming and preprocessing the data.
c) Finding the scope of improvement.
d) Selecting the suitable ML models for evaluation on the dataset.
e) Calculating the metrics and comparing them with the metrics of other ML classifiers.

5.1 Preprocessing

To build an intrusion-detection model, we performed a hybrid merger by combining CNN, bidirectional GRU, and DNN (i.e., DCNN-BiGRU). We preprocessed, cleaned, and transformed the data before applying classification models to the dataset. Most classifiers show performance improvement if the data are preprocessed. Data preprocessing helps to maintain the integrity of data by removing null values, duplicate records, irrelevant attributes, dirty data, and so on.

Artificial neural networks (ANNs) were trained on high-dimensional data from previous statistics of fraudulent detection to predict the possibility of data or information breaches in existing systems, and if any, the system automatically stops all the data or information flows without losing information. We discuss IDSs implemented through DL algorithms, the parameters on which DL procedures work, and how they will prevent cyber intelligence systems from getting breached.

Networks generate massive amounts of data, which requires preprocessing to remove noise, handle missing values, and normalize the data. Techniques such as data cleaning, feature scaling, and feature engineering are applied to improve data format in terms of values. Feature selection is one of the most important steps that helps ML models learn and evaluate better by removing unnecessary features that are insignificant in the overall evaluation. Many feature-selection algorithms exist. Dimension-reduction algorithms are used in cases where datasets have huge feature complexity. We first removed irrelevant features. The benefit of using a hybrid framework of ML classifiers for malicious intent detection in cyber security is their ability to gain or increase in overall performance, accuracy, and robustness with the combination of different classifiers. The hybrid framework can better generalize patterns and adapt to evolving attack techniques. In cases of unbalanced datasets where there are huge variations in the number of records between multiple class labels (attack types), hybrid frameworks outperform individual classifiers.

5.2 Data Collection and Processing Using One Hot Encoding

We used the CICIDS2019 dataset, the common network dataset that accurately reflects modern-day traffic. By utilizing the collective wisdom of CNN or correlation analysis, the selection process helps identify the most influential features in the dataset, contributing to better model accuracy and interpretability. Features with higher importance are considered more relevant for prediction and are retained, while features with lower importance are dropped. Manual features dropped can also be performed. Table 3 shows the feature list of the dataset considered for analysis. For preprocessing, the one-hot-encoding technique was used to help convert the string or categorical attributes into numerical form because the majority of DL classifiers work on numerical data.

5.3 Dataset

Table 2 presents the total attack labels contained by the dataset. It contains 18 attack classes, and we used the dataset for a multiclass classification-based scenario. To determine the efficiency of the proposed DCNN-BiGRU algorithm, we split the dataset into training and testing (validation) sets at a ratio of 75:25 (training: testing).

Our proposed model achieved the best accuracy of all the simulated models owing to the inherent best properties of all individual classifiers such as DNN, CNN, and AE. Table 2 shows the attributes of the dataset. The hybrid model was trained on labeled data where both normal and attack instances are included. The training stage consisted of optimizing the model parameters and determining the best performance setting. After training, the hybrid framework was deployed in real-time scenarios of networks to detect and respond to attacks. Incoming network traffic and device data were continuously monitored, and the hybrid model evaluated the data to identify suspicious patterns or anomalies. In the event of an attack being detected, appropriate actions can be taken, such as blocking malicious traffic or isolating affected devices.

We obtained data from multiple authentic sources. The data contained several intrusions that were simulated in an intelligence network context. The simulated models were tested on raw TCP/IP network traffic data. Several networks were bombarded with multiple known attacks in the simulated environment, and all normal and abnormal parameters were saved in some CSV, Excel, or Notepad files. These files were further considered as the raw datasets for IDS scenarios. The attributes were stored for a few days, hours, and so on. The simulated environments had two types of data according to their behavior: normal and abnormal. If the classifier is trained using unbalanced data labels, there is a possibility of getting variations in classifier precision capability. Some specific class labels have to underfit or overfit data records, and so accordingly the predictions will be made.

Table 2. Dataset Used With Label Types in Current Scenario

Used Dataset	**File specified**	**Label Types(Total Classes)**	**Training & Training size** X_train Shape,Y_train Shape train_test_split=(75% , 25%)	**Total Dataset size** Total Attributes:76 Total Class Label:18 Total records: 431371
CICIDS2019	CICIDS2019	Benign,DrDoS_NTP,TFTP,Syn,UDP, DrDoS_UDP,UDP-lag,MSSQL,DrDoS_MSSQL,DrDoS_DNS,DrDoS_SNMP,LDAP,DrDoS_LDAP,Portmap,NetBIOS,DrDoS_NetBIOS,UDPLag,WebDDoS	X_train Shape: (323528, 76) Y_train Shape: (323528, 18) X_test Shape: (107843, 76) Y_test Shape: (107843, 18)	X Shape: (431371, 76) Y Shape: (431371, 18)

Table 3. Dataset Features Used

Dataset	**Attributes used (76 Attributes in total after dropping 2 attributes,** i.e**.** Idle_min and class attribute): as we focus on multiclass classification, we opted "label" attribute after removing the class attribute as both contain duplicate data from the dataset **)**
CICIDS2019	index, Protocol, Flow Duration, Total Fwd Packets, Total Backward Packets, Fwd Packets Length Total, Bwd Packets Length Total, Fwd Packet Length Max, Fwd Packet Length Min, Fwd Packet Length Mean, Fwd Packet Length Std, Bwd Packet Length Max, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd Packet Length Std, Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Flow IAT Min, Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max, Fwd IAT Min, Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min, Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Length, Bwd Header Length, Fwd Packets/s, Bwd Packets/s, Packet Length Min, Packet Length Max, Packet Length Mean, Packet Length Std, Packet Length Variance, FIN Flag Count, SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count, CWE Flag Count, ECE Flag Count, Down/Up Ratio, Avg Packet Size, Avg Fwd Segment Size, Avg Bwd Segment Size, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate, Subflow Fwd Packets, Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes, Init Fwd Win Bytes, Init Bwd Win Bytes,Fwd Act Data Packets, Fwd Seg Size Min, Active Mean, Active Std, Active Max, Active Min, Idle Mean, Idle Std, Idle Max'.

5.4 Data Normalization

Multiple data-normalization methods are available. We used the standard scalar technique to normalize data in the range of [1, +1] or [0, +1]. The data normalization works in such a way that it normalizes the data as per the nature of DL procedures. It normalizes the feature values in the range of [1, +1] or [0, +1]. Data Normalization can also be referred to as a standardization procedure, as it boosts up the classifier efficiency in terms of memory consumption, execution time, and accuracy.

5.5 Deep Neural Network (DNN) Model

DNN is the type of ANN that can be used to build multiple strong IDS frameworks; DNN may also be mutated by itself and applied to datasets for attack or class label prediction. We used a DNN model depicted in Figure 5, consisting of the input layer, hidden layer (128 * 256 * 128 neurons), and rectified linear unit (ReLu) activation function, with a learning rate of 0.0001 and dropout scale of 0.1 (the same for all simulated models below).

A DNN can be modified in terms of the hidden layers it contains; the more hidden layers with more neurons or processing, the more complexity. DNNs may be used in multiple research in some ways. They take inputs and do more refined calculations or computations on them and provide results through output layers. DNN is one of the best frameworks for real-life problems like classification and regression of incoming data through prediction and learning.

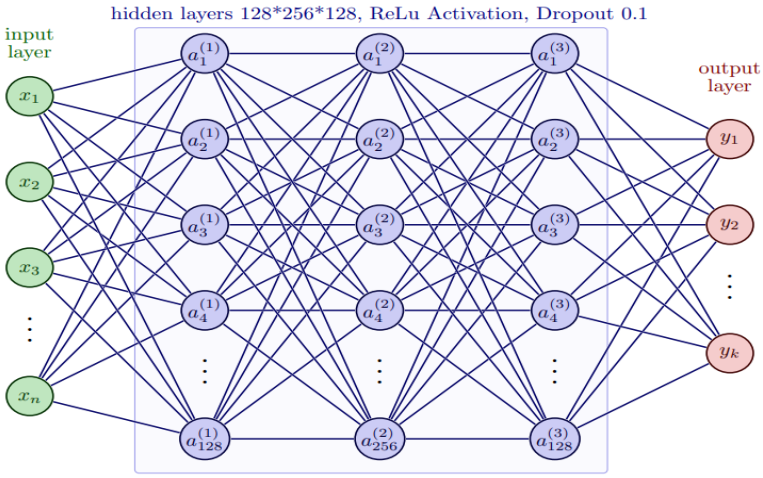


Figure 5. DNN Model Applied with Input, Hidden, and output Layers

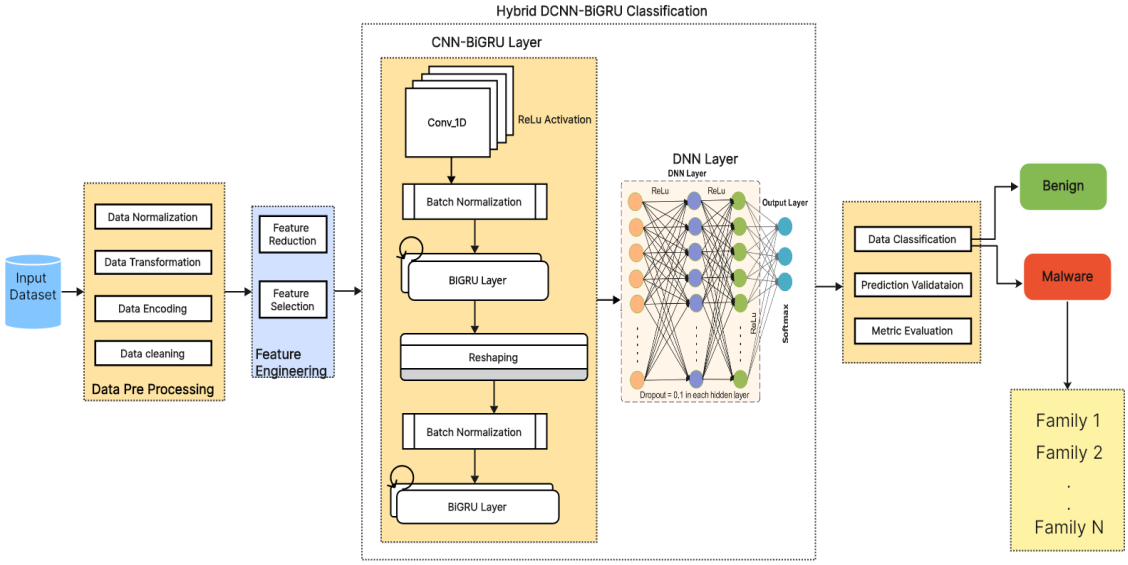


Figure 6. Proposed Deep CNN-BiGRU Model

5.6 Convolution Neural Network (CNN)

CNN consists of multiple array units to process data (Albawi *et al*., 2017). The primary objective of CNN is to extract features, although by maintaining the sequential data information, CNN produced the best results for some problems like image recognition. It works by finding spatial data correlations in the input data. Figure 7 shows the simple model for the CNN information flow that we applied to our dataset as described above. First used, Conv_1D (128) layer, ReLu as an activation function, Adam as an optimizer, Conv_1D (256) layer, and a dropout rate of 0.1, as depicted in Figure 7.

CNN is applied to visual or image data classification tasks. It consists of three layers: the input layer, the hidden layer, and the output layer. Data is fed into the network; the output from the layer is obtained by a mechanism called feedforward. Errors are measured using error functions, such as square error loss function or cross-entropy. The calculation done by error functions helps identify whether the model is performing well or not. The backpropagation method is deployed between layers to minimize the loss.

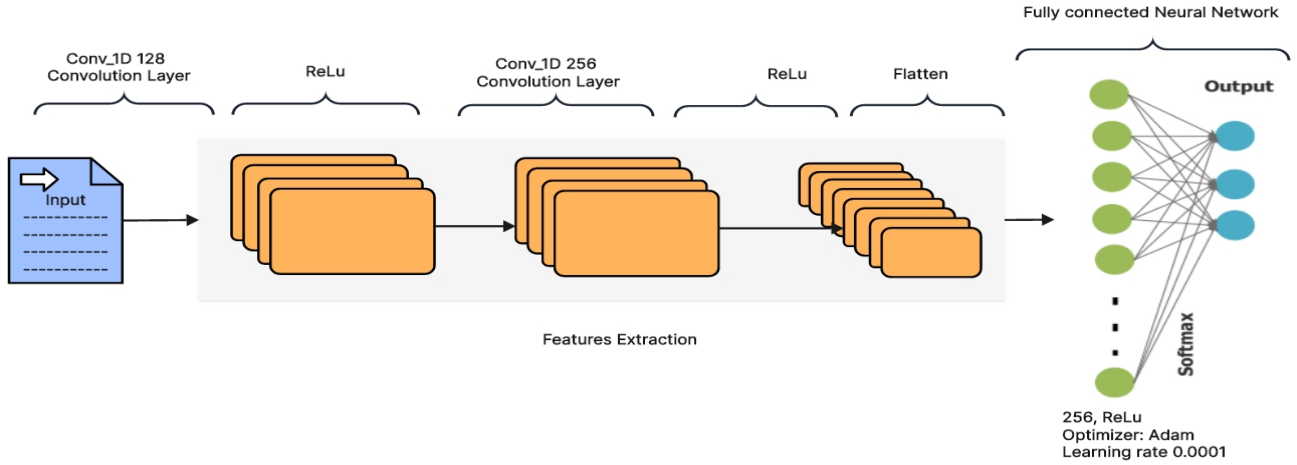


Figure 7. CNN Model

5.7 Autoencoder (AE)

Autoencoders encode data post-compressing. They follow unsupervised learning. They are a type of ANN or simply derived from ANNs. They learn and reconstruct data back into its original form from the encoded compressed form. The compression and successive rebuild phenomena are extremely complicated operations. The input data features or characteristics are completely autonomous from each other. The goal of an autoencoder (Tschannen *et al*., 2018) is to learn how to perform lower-dimensional representation (encoding) of complicated high-dimensional data (.i.e., data dimension reduction). The AE model of nature (128*64*32*16) was used in our study, as depicted in Figure 8, with the same parameters.

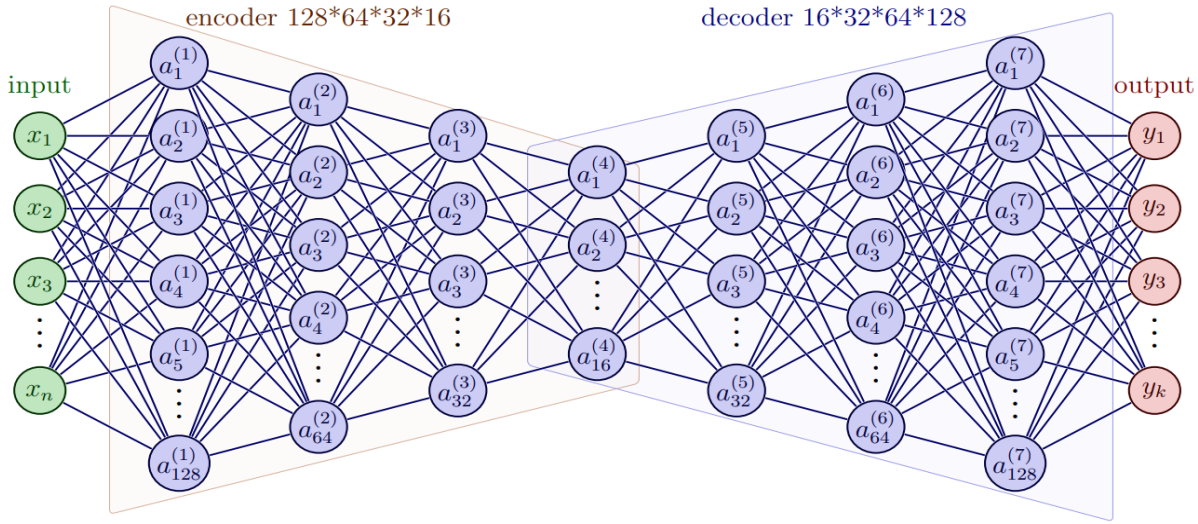


Figure 8. AE Encoder & Decoder Model Layers Applied

5.8 DCNN-BiGRU

The DCNN-BiGRU architecture is composed of the CNN model, kept as the first layer for the feature selection and boosting process (Gu *et al*., 2018). We performed manual removal after BiGRU sequence forecasting or classification to get a reliable output. The DNN layer was placed at the end to boost the performance by minimizing the loss and error ratio. We built DCNN-BiGRU by first enclosing CNN layers as the first layer, followed by BiGRU layers, and at last, DNN. Figure 6 shows the proposed DCNN-BiGRU model, and the steps we used to implement the model are given in the algorithm.

5.9. Proposed DCNN-BiGRU Algorithm:

Algorithm: DCNN-BiGRU- Model Training, Validation

Data: Dtrain: The training dataset for the ML models (df), multiple Hybrid learners

Result: A hybrid fusion model, with an Accuracy level applied

Step 1: To select relevant features in nature from the Input set of features that can be denoted as *f1,f2…….fn*.
Where 'n' represents the number of features considered.
Step 2: Divide the Input dataset $df_{Features}$ into training and testing sets.
Step 3: Layered Stack of model ( ).
Step 4: Apply Convolution1D Layer, where Kernel attributes shape = *n,* with Input shape (76, 1).
Step 5: Apply the Activation function on the flow: ReLu.
Step 6: Application of BatchNormalization ( ) Layer on Model.
Step 7: Apply Bidirectional GRU Cell Layer to the framework, (*Neurons =64).*
Step 8: Apply Reshape Layer on the flow, i.e. (*input size =128*).
Step 9: Apply the Batch Normalization Layer to the model ().
Step 10: Apply Bidirectional GRU Cell Layer to the framework, (*Neurons =128).*
Step 11: Apply Dropout Layer.
Step 12: Apply Deep Neural Network (DNN) of capacity, *Neurons =256,128,16(output Layer).*
Step 13: Apply the Activation function on the flow: ReLu.
Step 13: Here Dense (n_classes) for our dataset is *16*, *Neurons (n_classes*) = *Neurons16(output Layer).*
Step14: Apply the Activation function on the flow: Softmax

The first layer (CNN) is boosted and activated using the ReLu activation function as in the equation is below. The equations for Softmax activation and batch normalization procedure are also given below.

$$\text{Relu (K)} = \text{maximum (0, k)}$$

$$\text{BatchNormalization (k)} = \frac{(k-Min)}{(Max-Min)}$$

$$\text{Softmax (Input}_K\text{)} = \frac{\exp(K_i)}{\sum \exp(K_i)}$$

Below are some parameters used in the proposed algorithm. The optimizer function used was Adam, with a learning rate of 0.0001, used for all the models—DNN, AE, CNN, and DCNN-BiGRU. We also employed the ReLu activation for multiclass classification, Softmax activation, one hot encoding, and categorical entropy loss function.

5.9. Proposed DCNN-BiGRU Algorithm Step-7 Detailed Illustration

The computation steps are shown below. The update gate component is responsible for identifying the previous hidden state information that is important and to be retained from past timestamps and the new information from current timestamps to be incorporated. It conveys past meaningful information and updates important information in place of less meaningful data.

- The mathematical computations for the update gate, $z = \sigma (W_z \cdot x_t + U_z \cdot h_{(t-1)} + b_z)$, can be evaluated using Equation 2.
- The reset gate component is responsible for evaluating whether the previous information is necessary to be carried further for the next timestamps or must be discarded. At timestamp t, the reset-gate calculation can be evaluated as below.
- $r = \sigma (W_r \cdot x_t + U_r \cdot h_{(t-1)} + b_r)$, which can be evaluated using Equation 1.
- The intermediate memory component, $h\sim = \tanh (W_h \cdot x_t + r * U_h \cdot h_{(t-1)} + b_z)$, can be evaluated using Equation 3.
- The output layer takes the final hidden state as input and produces the network output. This could be a single number, a sequence of numbers, or a probability distribution over classes, depending on the task at hand.
- Output $h = z * h_{(t-1)} + (1-z) * h\sim$, which can be evaluated using Equation 4.
- In Figure 9, x_t represents the sequential input data with a dimension (Dim) of S * B *V, where 'S' is the sequence length, 'B' is the batch size, and 'V' is the number of inputs during a single iteration.
- r_t is the reset gate unit in the GRU cell, and z_t is the update gate; the calculation equation is described above and in Section 4.

The two more major components used in every GRU cell are candidate hidden state $\bar{h_t}$ and final hidden state h_t denoted at any current timestamp. The reset gate determines the values from the previous hidden state to further forget. The update gate determines the values from the hidden and candidate hidden states to be considered further.

The legends in the above step represent the element-wise multiplication and addition between the learning weighted matrix at a time stamp between multiple components, W_z is the weighted learning matrix for the update gate, W_r is the weighted learning matrix of reset gate and W_h is a weighted learning matrix for the candidate's hidden state.

$r_t \odot h_{t-1}$ represents the element-wise multiplication of the reset gate and previous hidden state, and the notation $\odot$ represents the Hadamard product.

$z_t \odot h_{t-1}$ represents the element-wise multiplication of the update gate and previous hidden state; it shows all the hidden states to be retained as per the update gate.

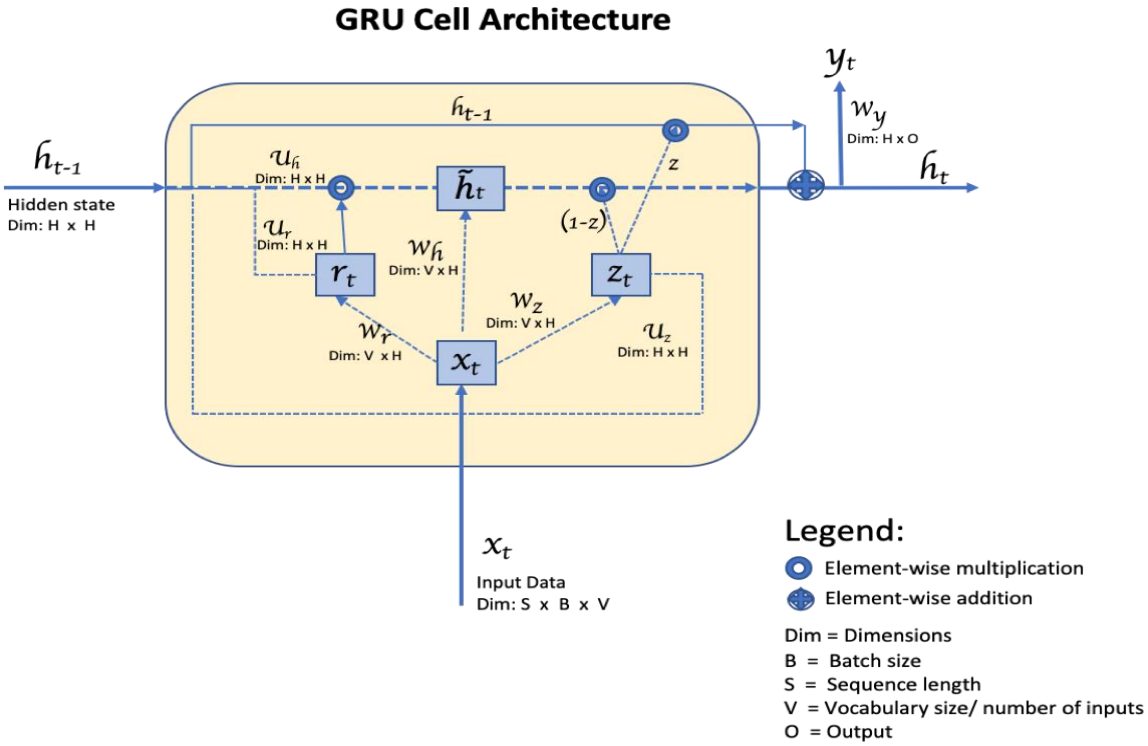


Figure 9. GRU Cell Architecture

6. RESULT ANALYSIS AND DISCUSSION

We briefly analyze the metrics obtained by the IDS classifiers, comparing CNN, DNN, AE (conventional models), and the proposed framework (DCNN-BiGRU). Convolution networks have convolution layers as the basic building blocks. They are widely adopted to select distinct features. They are modeled after the concept of biological neurons, where characteristics from the previous layer of convolution are used in high-level feature abstraction. Numerous artificial neurons work in various layers to compute the weighted factor and perform its sum at the input and output of activation. After the CNN layer is the BiGRU layer, which performs well in cases that require quick learning and adapting to new inputs. GRU performs well by saving short-term dependencies in the data and the DNN layer reduces the error and loss. The accuracy-evaluation experiments were performed using several ML methods to assess the performance of our proposed model. We used 431371 samples, 18 class labels, and 78 attributes from the cicddos2019 dataset from Mendeley, out of which we removed 2 attributes. We manually eliminated features owing to duplicity and irrelevancy. Table 3 represents the attribute names considered for analysis, and Table 2 shows the label types in the dataset. Table 5 shows the percentage distribution of records in data classwise. It can be observed that the data distribution is uneven. Table 6 shows the performance evaluation of the proposed model compared with that of other studies.

One of the problems faced with the available datasets in network analysis or IoT analysis is the distribution of class labels for attacks in terms of tuples in CSV files or text files; the distribution of data among various class labels is uneven, which compromises the model training and testing. If we want the classifier to have good prediction capability and accuracy for the class attack labels, then we cannot compromise in terms of the provided records in the datasets because, for any specific class, fewer records or tuples result in training issues in terms of performance. If the classifier is trained on small amounts of data for class labels and expectations are high, then it cannot be guaranteed to perform well and give good precision and f1 scores. We proceed with the current distribution; however, we can use techniques like SMOTE to remove the class label imbalance ratios when required.

We compared the performance in terms of accuracy and the loss rate between DNN, CNN, AE, and DCNN-BiGRU techniques. The models were trained and validated on datasets on GPU environments or simply platforms that show a decrease in the training duration as shown in Table 8 for model performance description for the cicddos2019 dataset. Table 7 shows the model description and hyperparameter tuning for all four algorithms used on the cicddos2019 dataset. Table 4 shows the metric calculation base, and Figures 10 to 13 show the confusion matrix of all the classifiers. Our proposed model gives a higher accuracy of 99.70 % in multiclass classification. Figures 14 to 17 show the training, validation, accuracy, and loss incurred by all the algorithms. The efficiency of the IDS algorithm depends on multiple parameters like dataset features selected, the learning rate activation function, and so on. Developing an ideal classifier is an interesting field of research. Thus, it will not be justified to say the proposed model is the best fit for every dataset.

6.1 Confusion Matrix (CM)

The confusion matrix is used to show the correlation between the actual and expected class. It is also helpful in estimating AUC and ROC curves, specificity, precision, recall, and accuracy. Table 4 shows the confusion matrix.

Table 4. Confusion Matrix (CM)

Class	Actual Positive Class	Actual Negative Class
Predicted Positive Class	TP	FP
Predicted Negative Class	FN	TN

i. Accuracy

The percentage of instances that are correctly classified is calculated as follows:

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

Table 5**.** Percentage Classwise Distribution of Records on Cicddos2019 Dataset

Traffic Type	Class Name	Instances	Percentage
Normal	Benign	97381	22.679
Abnormal	DrDoS_NTP	121368	28.135
Abnormal	TFTP	98917	22.931
Abnormal	Syn	49373	11.446
Abnormal	UDP	18090	4.194
Abnormal	DrDoS_UDP	10420	2.416
Abnormal	UDP-lag	8872	2.057
Abnormal	MSSQL	8523	1.976
Abnormal	DrDoS_MSSQL	6212	1.44
Abnormal	DrDoS_DNS	3669	0.851
Abnormal	DrDoS_SNMP	2717	0.63
Abnormal	LDAP	1906	0.442
Abnormal	DrDoS_LDAP	1440	0.334
Abnormal	Portmap	685	0.159
Abnormal	NetBIOS	644	0.149
Abnormal	DrDoS_NetBIOS	598	0.139
Abnormal	UDPLag	55	0.013
Abnormal	WebDDoS	51	0.012

If datasets face class label imbalance, multiple algorithms work for the input datasets and help remove the class label imbalance ratios, and we can fit the SMOTE function as per need. We can directly apply the algorithm function on the complete dataset in a way that automatically adjusts the data records classwise, either by replication or generation of more records for the underfit class label. We can also specify the value in terms of integer up to which we want the class label records to be scaled or increased.

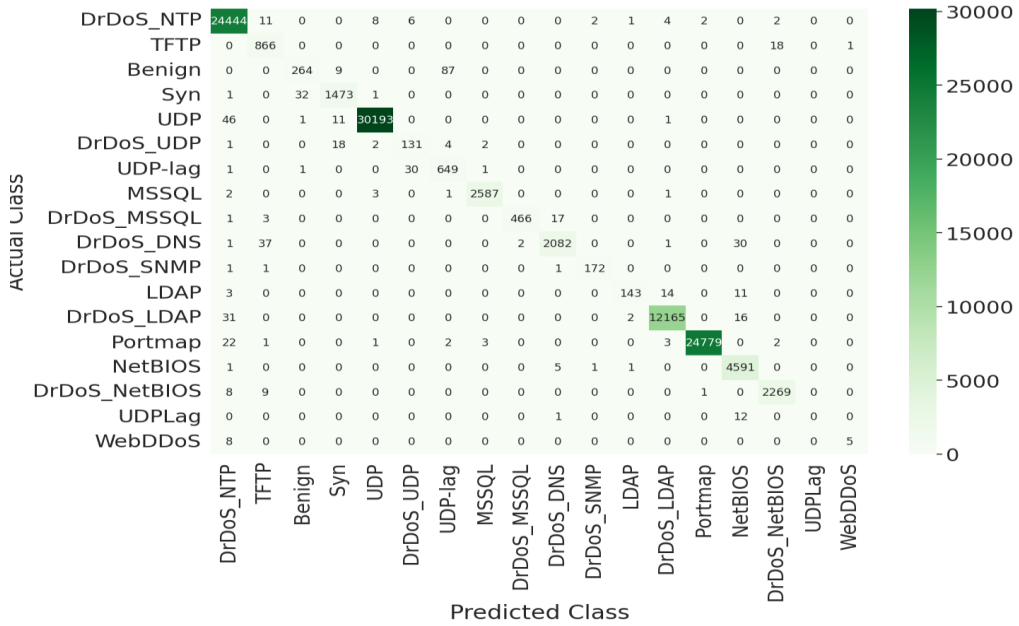


Figure 10. DNN Confusion Matrix

ii) Error Rate

The percentage of predicted values that are incorrectly categorized is determined as follows:

Error rate = 1- Accuracy

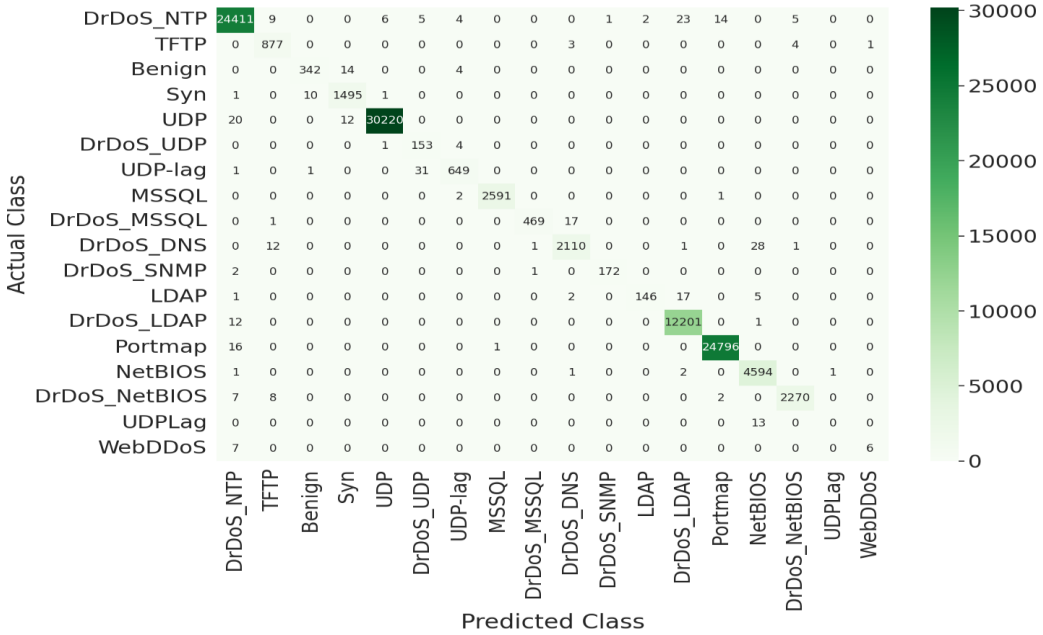


Figure 11. CNN Confusion Matrix

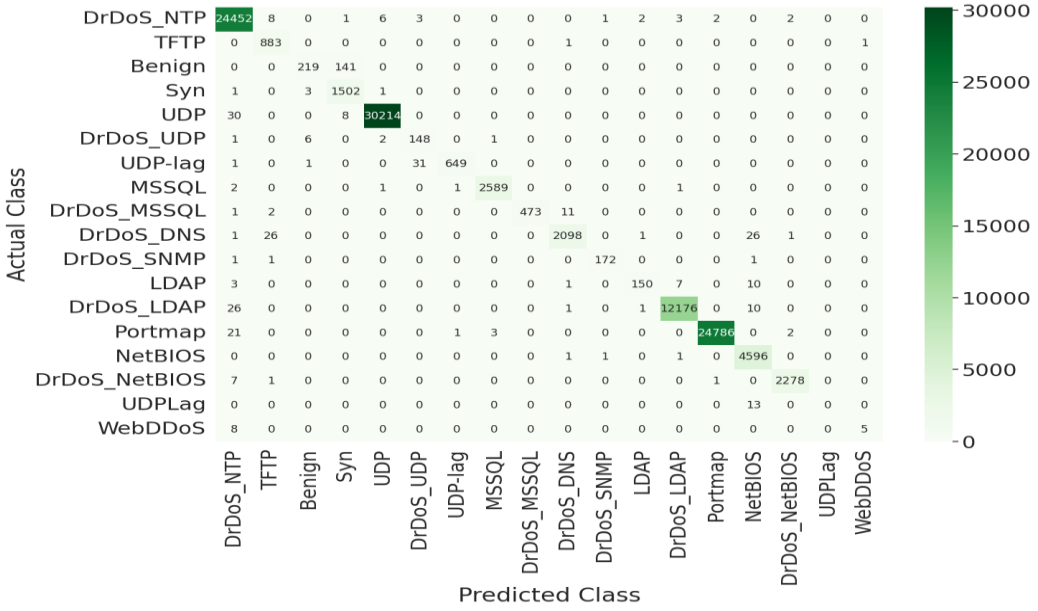


Figure 12. Auto Encoder Confusion Matrix

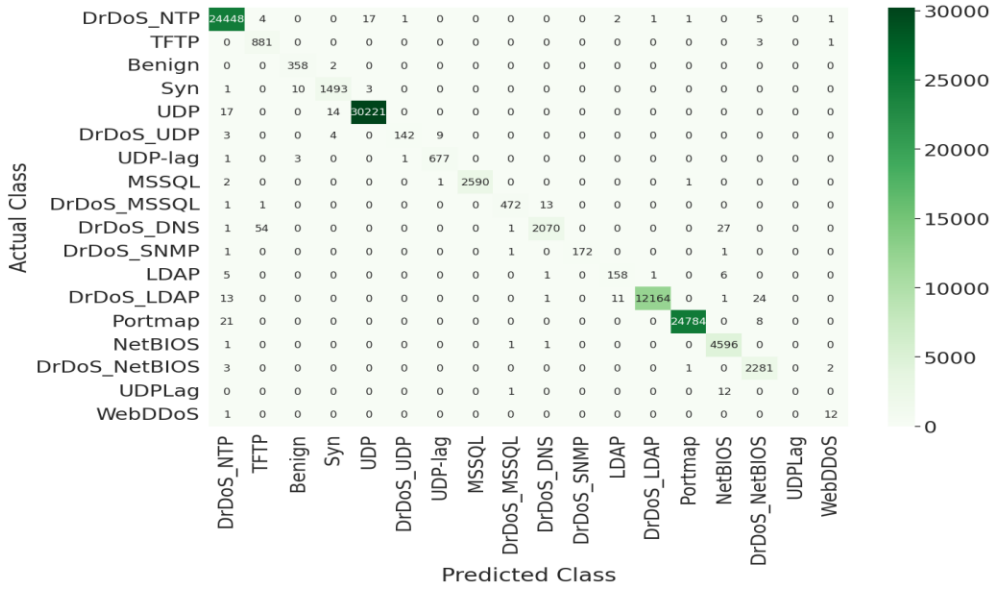


Figure 13. DCNN-BiGRU: Hybrid Confusion Matrix

Table 6. Performance Evaluation of the Proposed Model and Those of Other Studies

Paper	Model	Dataset	Accuracy	Loss	Inference Time
Reddy *et al*. (2022)	MLP	NSL-KDD, wireless sensor networks	98 %(M)	0.0198	13.25 seconds
Alazab *et al*. (2024)	HHO-MLP	KDD Datasets	93.17(M)	*	*
AL-Hawawreh *et al*. (2018)	ADS	NSL-KDD and UNSW-NB15	92.40%(M) and 98.60%(M)	8.2, 1.8	5.50 seconds,2.25 seconds
Halbouni *et al*. (2022)	CNN-LSTM	CIC-IDS 2017, UNSW-	99.59%(B),	*	763 seconds, 244

Paper	Model	Dataset	Accuracy	Loss	Inference Time
	Hybrid	NB15, WSN-DS	93.68%(B), 99.61%(B)		sec, 112 sec
Altunay *et al*. (2022)	CNN+LSTM	UNSW-NB15, X-IIoTID	93.21%(B), 92.9% (M)(both for UNSW-NB15) & 99.84% (B), 99.80%(M) for IIoTID	6.21%(B), 6.28%(M) (UNSB-NB15) & 0.12%(B), 0.12%(M) for IIoTID	*
Sun *et al*. (2020)	CNN-LSTM Hybrid	CICIDS207	99.50%[M]	*	*
Popoola *et al*. (2021)	LAE-BLSTM	BOT-IOT	91.80%(M)	*	0.2359sec
Zhai *et al*. (2023)	CNN–GRU	NSL-KDD	78.79%	*	*
De La Torre Parra *et al*. (2020)	DCNN LssSTM	Version N_BaIoT	94.80% (B) 94.20% (M)	*	*
Proposed study	DCNN-BiGRU	cicddos2019	**0.9970(M)**	0.0123	39.38 seconds

* represents: Not available, M: multiclassification of class label, B: binary label classification

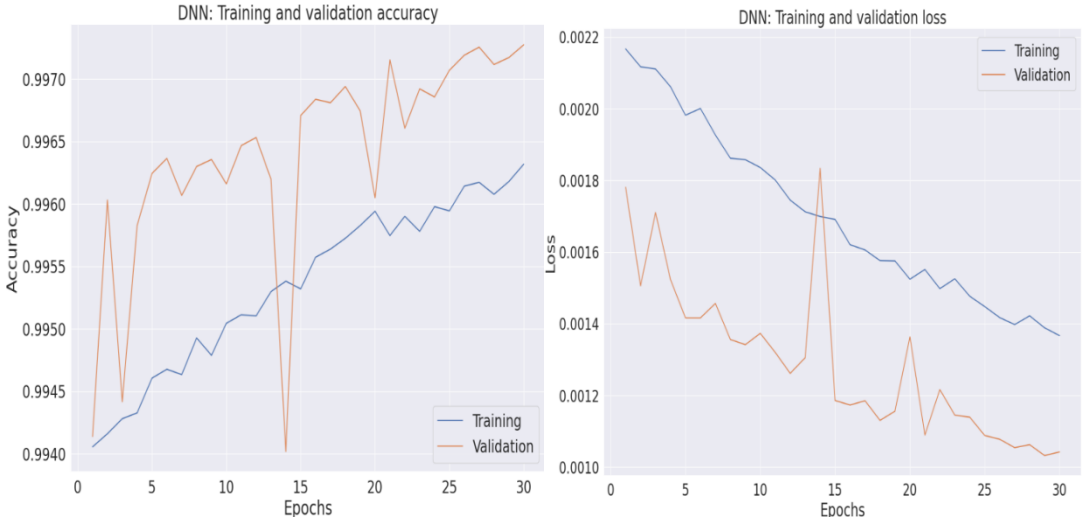


Figure 14. DNN Training, Validation, Accuracy, Loss

Figure 15. CNN Training, Validation, Accuracy, Loss

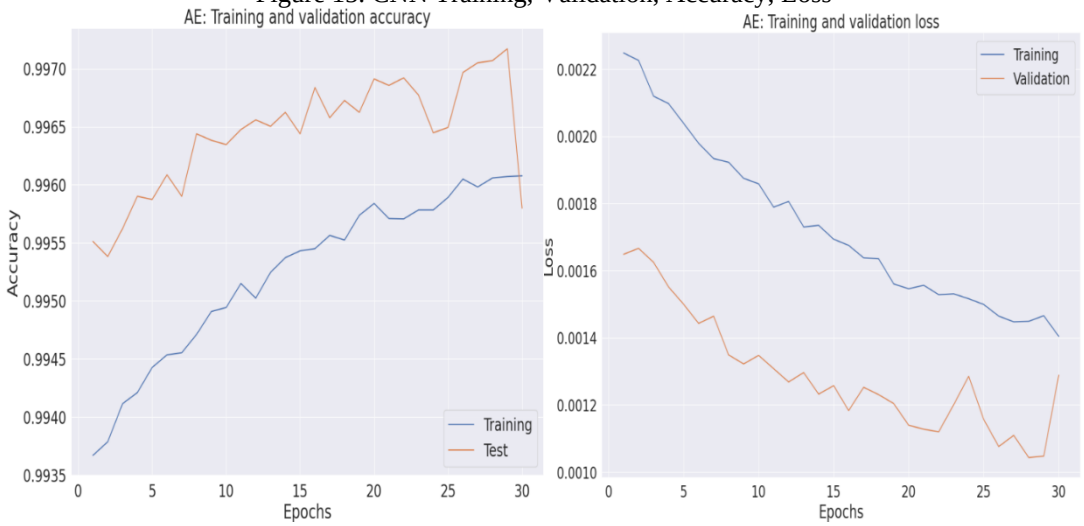


Figure 16. AE Training, Validation, Accuracy, Loss

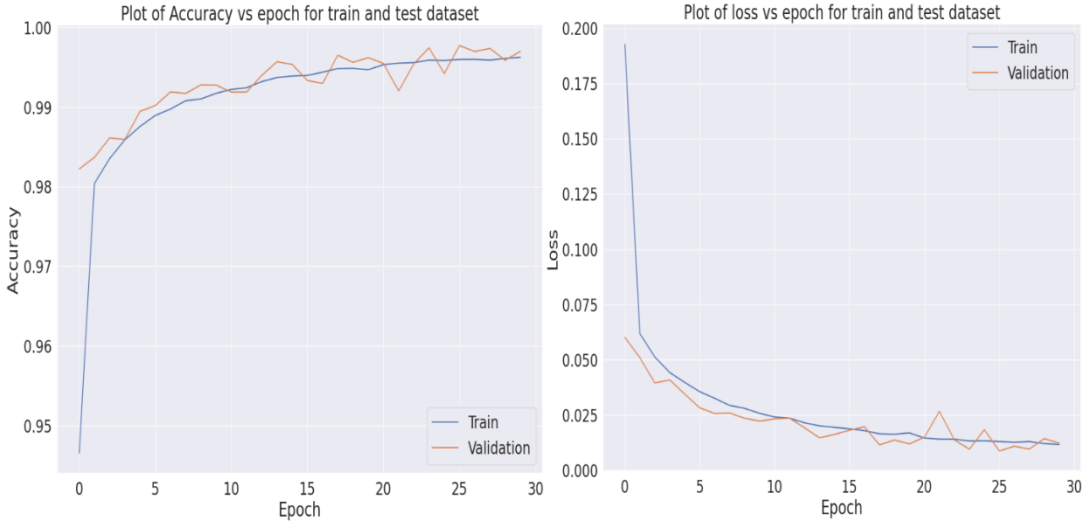


Figure 17. DCNN-BiGRU Training, Validation, Accuracy, Loss

Table 7. Model Description With Hyperparameter Tuning Used on the Cicddos2019 Dataset

Model Name(Dynamic)	**Model Description**							
	Layer Architecture	Dropout	Loss entropy	Learning Rate	Platform	Activation	Epochs	Batch Size
DNN(Dense Neural Network)	Dense(128 * 256 * 128)	0.1	BinaryCrossentropy	0.0001	GPU	ReLu/softmax	30	128
CNN	Convolution1D(128) *Convolution1D(256)* Dense(256) * Dense(18)	0.1	categorical_crossentropy	0.0001	GPU	ReLu/softmax	30	128
Auto Encoder/decoder	Dense(128 * 64 *32 *16)(16*32*64*128)	0.1	Mae	0.0001	GPU	elu	30	128
DCNN-BiGRU	Convolution1D(128) * BiGRU(64) * BiGRU (128) * Dense(256)*Dense (128)*Dense (n_classes)	0.1	BinaryCrossentropy	0.0001	GPU	ReLu/softmax	30	128

Table 8. Model Performance Description Used on the Cicddos2019 Dataset

Model Name	**Model Performance**									
	Layer Architecture	Optimizer	Dataset	Inference Time	Recall	Precision	F1 score	Loss	**Accuracy**	ROC-AUC
DNN(Dense Neural Network)	Dense(128 * 256 * 128)	Adam	cicddos2019	10.36 seconds	0.9947	0.9946	0.9946	0.0018	**0.9948**	0.9325
CNN	Convolution1D(128)* Dense(256) * Dense(18)	Adam	cicddos2019	10.38 seconds	0.9966	0.9966	0.9966	0.0129	**0.9967**	0.9431
Auto Encoder	Dense(128 * 64 *32 *16)(16*32*64*128)	Adam	cicddos2019	6.76 seconds	0.9957	0.9957	0.9956	0.0013	**0.9958**	0.9351
DCNN-BiGRU	Convolution1D(128) * BiGRU(64) * BiGRU (128) * Dense(256)*Dense (128)*Dense (n_classes)	Adam	cicddos2019	39.38 seconds	0.9969	0.9969	0.9969	0.0123	**0.9970**	0.9615

7. CONCLUSION

Using our proposed model, we analyzed real network traffic of the cicddos2019 dataset and compared the efficiency of multiple models of a similar type. The proposed hybrid DCNN-BiGRU model outperformed all the other standard benchmark classifiers of a similar type evaluated in this study. The major objective was to enhance the multiclass classification accuracy of ML classifiers on the applied datasets, where correct predictions had to be made under multiple attack labels. The performance of ML classifiers decreases when making multiclass-based predictions. Our proposed model proved dependable from an accuracy perspective. DCNN-BiGRU architecture leverages the strengths of each model; the proposed hybrid model can detect abnormal attack patterns in datasets with a higher accuracy percentage than the standard models. Experimental results show that the proposed DCNN-BiGRU hybrid model outperformed other standard models in accuracy, with a value of 99.70 %. As DCNN-BiGRU is a hybrid model, it takes advantage of individual learners like CNN and GRU, outperforming isolated models. The proposed DCNN-BiGRU is an optimized solution from an industry perspective as it is suitable for network and system administrators to deploy; it requires fewer resources than other homogeneous models. Although DCNN-BiGRU performed very well on the current dataset, there is always a scope for improvement in terms of memory and time complexity, which can be improved by using multiple techniques. The limitations of this study are that the performance may degrade when the data have long-term dependencies The proposed system has a slightly high training time in comparison to other models, despite being more accurate. A further merger with other suitable optimization or feature-reduction techniques can improve the achieved accuracy.

Availability of supporting data: Below are the links where datasets are available. The datasets generated and/or analyzed during the current study are not publicly available owing to [security reasons] but are available from the corresponding author on reasonable request. Data will be made available on request.
For CIC-DDoS2019 URL: https://data.mendeley.com/datasets/ssnc74xm6r/1

REFERENCES

Ahmed Issa, A. S., & Albayrak, Z. (2023). DDoS attack intrusion detection system based on hybridization of CNN and LSTM. *Acta Polytechnica Hungarica*, 20(2), 105–123. DOI: https://doi.org/10.12700/aph.20.2.2023.2.6.

Alazab, M., Abu Khurma, R., Castillo, P. A., Abu-Salih, B., Martín, A., & Camacho, D. (2024). An effective networks intrusion detection approach based on hybrid Harris Hawks and multi-layer perceptron. *Egyptian Informatics Journal*, 25(100423), 100423. DOI: https://doi.org/10.1016/j.eij.2023.100423.

Albakri, A., Alhayan, F., Alturki, N., Ahamed, S., & Shamsudheen, S. (2023). Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification. *Applied Sciences (Basel, Switzerland)*, 13(4), 2172. DOI: https://doi.org/10.3390/app13042172.

Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. *2017*

International Conference on Engineering and Technology (ICET). Presented at the 2017 International Conference on Engineering and Technology (ICET), Antalya. DOI: https://doi.org/10.1109/icengtechnol.2017.8308186.

Aljuaid, W. H., & Alshamrani, S. S. (2024). A deep learning approach for intrusion detection systems in cloud computing environments. *Applied Sciences (Basel, Switzerland)*, 14(13), 5381. DOI: https://doi.org/10.3390/app14135381.

Altameem, A. A., & Hafez, A. M. (2022). Behavior analysis using enhanced fuzzy clustering and deep learning. Electronics, 11(19), 3172. DOI: https://doi.org/10.3390/electronics11193172.

AL-Hawawreh, M., Moustafa, N., & Sitnikova, E. (2018). Identification of malicious activities in industrial internet of things based on deep learning models. *Journal of Information Security and Applications*, 41, 1–11. DOI: https://doi.org/10.1016/j.jisa.2018.05.002.

Altunay, H. C., & Albayrak, Z. (2023). A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. *Engineering Science and Technology an International Journal*, 38(101322), 101322. DOI: https://doi.org/10.1016/j.jestch.2022.101322.

Amma, B., & Selvakumar. (2020). Anomaly detection framework for Internet of things traffic using vector convolutional deep learning approach in fog environment. *Future Generations Computer Systems: FGCS*, 113, 255–265. DOI: https://doi.org/10.1016/j.future.2020.07.020.

Attou, H., Guezzaz, A., Benkirane, S., Azrour, M., Farhaoui, Y. (2023). Cloud-Based Intrusion Detection Approach Using Machine Learning Techniques. *Big Data Mining and Analytics*, 2023, 6, 311–320.

Bhatt, P., & Morais, A. (2018, December). HADS: Hybrid anomaly detection system for IoT environments. *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, Hamammet, Tunisia. DOI: https://doi.org/10.1109/iintec.2018.8695303.

Chandrasekar, S., Subburathinam, K., & Kannan, S. (2023). Analysis for heart disease prediction using deep neural network and vgg_19 convolution neural network. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 30(4). DOI: https://doi.org/10.23055/ijietap.2023.30.2.8603.

Chen, H., & Ren, L. (2023). ACO-GCN: A fault detection fusion algorithm for wireless sensor network nodes. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 30(2). https://doi.org/10.23055/ijietap.2023.30.2.8801.

De La Torre Parra, G., Rad, P., Choo, K.-K. R., & Beebe, N. (2020). Detecting Internet of Things attacks using distributed deep learning. *Journal of Network and Computer Applications*, 163(102662), 102662. DOI: https://doi.org/10.1016/j.jnca.2020.102662.

Farhan, R. I., Maolood, A. T., & Hassan, N. F. (2020). Optimized Deep Learning with Binary PSO for Intrusion Detection on CSE-CIC-IDS2018 Dataset. *Journal of Al-Qadisiyah for Computer Science and Mathematics*, 12(3). DOI: https://doi.org/10.29304/jqcm.2020.12.3.706.

Gao, J. (2022). Network intrusion detection method combining CNN and BiLSTM in Cloud Computing Environment. *Computational Intelligence and Neuroscience*, 2022, 7272479. DOI: https://doi.org/10.1155/2022/7272479.

Ghani, H., Virdee, B., & Salekzamankhani, S. (2023). A deep learning approach for network intrusion detection using a small features vector. *Journal of Cybersecurity and Privacy*, 3(3), 451–463. DOI: https://doi.org/10.3390/jcp3030023.

Gosain, A., & Dahiya, S. (2016). Performance analysis of various fuzzy clustering algorithms: A review. *Procedia Computer Science*, 79, 100–111. DOI: https://doi.org/10.1016/j.procs.2016.03.014.

Grace, M., & Sughasiny, M. (2022). Malware detection for Android application using Aquila optimizer and Hybrid LSTM-SVM classifier. *ICST Transactions on Scalable Information Systems*, e1. DOI: https://doi.org/10.4108/eetsis.v9i4.2565.

Gurung, S., Sikkim Manipal Institute of Technology, Sikkim Manipal University, Majitar, Sikkim, India, Kanti Ghose, M., & Subedi, A. (2019). Deep learning approach on network intrusion detection system using NSL-KDD dataset. *International Journal of Computer Network and Information Security*, 11(3), 8–14. DOI: https://doi.org/10.5815/ijcnis.2019.03.02.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., … Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, 77, 354–377. DOI: https://doi.org/10.1016/j.patcog.2017.10.013.

Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: Hybrid deep neural network for network intrusion detection system. *IEEE Access: Practical Innovations, Open Solutions*, 10, 99837–99849. DOI: https://doi.org/10.1109/access.2022.3206425.

Jun-Ho Park, & Baek, S. H. (2024). Two-Step Methodology for Statistical Anomaly Detection and Prediction Using XGBoost Regression in Blower Motor Vibration Time Series Data. *International Journal of Industrial Engineering: Theory, Applications and Practice*, 31(4). DOI: https://doi.org/10.23055/ijietap.2024.31.4.9989.

Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1). DOI: https://doi.org/10.1186/s42400-019-0038-7.
Kumar, R., Kumar, P., Tripathi, R., Gupta, G. P., Garg, S., & Hassan, M. M. (2022). A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network. *Journal of Parallel and Distributed Computing*, 164, 55–68. DOI: https://doi.org/10.1016/j.jpdc.2022.01.030.

Mohammadpour, L., Ling, T. C., Liew, C. S., & Aryanfar, A. (2022). A survey of CNN-based network intrusion detection. *Applied Sciences (Basel, Switzerland)*, 12(16), 8162. DOI: https://doi.org/10.3390/app12168162.

Nikolov, D., Kordev, I., & Stefanova, S. (2018, September). Concept for network intrusion detection system based on recurrent neural network classifier. *2018 IEEE XXVII International Scientific Conference Electronics - ET*. Presented at the 2018 IEEE XXVII International Scientific Conference Electronics (ET), Sozopol. DOI: https://doi.org/10.1109/et.2018.8549584.

Popoola, S. I., Adebisi, B., Hammoudeh, M., Gui, G., & Gacanin, H. (2021). Hybrid deep learning for botnet attack detection in the internet-of-things networks. *IEEE Internet of Things Journal*, 8(6), 4944–4956. DOI: https://doi.org/10.1109/jiot.2020.3034156.

Qazi, E. U. H., Faheem, M. H., & Zia, T. (2023). HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System. *Applied Sciences (Basel, Switzerland)*, 13(8), 4921. DOI: https://doi.org/10.3390/app13084921.

Reddy, G. V., Kadiyala, S., Potluri, C. S., Saravanan, P. S., Athisha, G., Mukunthan, M. A., & Sujaritha, M. (2022). An intrusion detection using machine learning algorithm Multi-Layer Perceptron (MlP): A classification enhancement in Wireless Sensor Network (WSN). *International Journal on Recent and Innovation Trends in Computing and Communication*, 10(2s), 139–145. DOI: https://doi.org/10.17762/ijritcc.v10i2s.5920.

Roopak, M., Tian, G. Y., & Chambers, J. (2020, January). An intrusion detection system against DDoS attacks in IoT networks. *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. Presented at the 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, USA. DOI: https://doi.org/10.1109/ccwc47524.2020.9031206.

Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6), 420. DOI: https://doi.org/10.1007/s42979-021-00815-1.

Shamshirband, S., Fathi, M., Dehzangi, A., Chronopoulos, A. T., & Alinejad-Rokny, H. (2021). A review on deep learning approaches in healthcare systems: Taxonomies, challenges, and open issues. *Journal of Biomedical Informatics*, 113(103627), 103627. DOI: https://doi.org/10.1016/j.jbi.2020.103627.

Shan, L., Liu, Y., Tang, M., Yang, M., & Bai, X. (2021). CNN-BiLSTM hybrid neural networks with attention mechanism for well log prediction. *Journal of Petroleum Science & Engineering*, 205(108838), 108838. DOI: https://doi.org/10.1016/j.petrol.2021.108838.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2021). Design of a fused triple convolutional neural network for malware detection: A visual classification approach. In *Communications in Computer and Information Science. Communications in Computer and Information Science* (pp. 279–289). DOI: https://doi.org/10.1007/978-3-030-81462-5_26.

Smmarwar, S. K., Gupta, G. P., Kumar, S., & Kumar, P. (2022a). An optimized and efficient android malware detection framework for future sustainable computing. *Sustainable Energy Technologies and Assessments*, 54(102852), 102852. DOI: https://doi.org/10.1016/j.seta.2022.102852.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2022b). A hybrid feature selection approach-based android malware detection framework using machine learning techniques. In *Lecture Notes in Networks and Systems. Lecture Notes in Networks and Systems* (pp. 347–356). DOI: https://doi.org/10.1007/978-981-16-8664-1_30.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2022c). Deep malware detection framework for IoT-based smart agriculture. *Computers & Electrical Engineering: An International Journal*, 104(108410), 108410. DOI: https://doi.org/10.1016/j.compeleceng.2022.108410.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2023a). AI-empowered malware detection system for industrial internet of things. *Computers & Electrical Engineering: An International Journal*, 108(108731), 108731. DOI: https://doi.org/10.1016/j.compeleceng.2023.108731.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2023b, July 29). XAI-AMD-DL: An explainable AI approach for android malware detection system using deep learning. *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. Presented at the 2023 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India. DOI: https://doi.org/10.1109/aic57670.2023.10263974.

Smmarwar, S. K., Gupta, G. P., & Kumar, S. (2024). Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: *A comprehensive review. Telematics and Informatics Reports*, 14(100130), 100130. DOI: https://doi.org/10.1016/j.teler.2024.100130.

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., & Chen, J. (2020). DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and Communication Networks*, 2020, 1–11. DOI: https://doi.org/10.1155/2020/8890306.

Tschannen, O. Bachem, M. Lucic (2018), Recent advances in autoencoder-based representation learning, *Third workshop Bayesian Deep Learn. (NeurIPS 2018)*, DOI: https://doi.org/10.48550/arXiv.1812.05069.

Tsogbaatar, E., Bhuyan, M. H., Taenaka, Y., Fall, D., Gonchigsumlaa, K., Elmroth, E., & Kadobayashi, Y. (2021). DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT. *Internet of Things*, 14(100391), 100391. DOI: https://doi.org/10.1016/j.iot.2021.10ssss0391.

Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IoT networks. *IEEE Access: Practical Innovations, Open Solutions*, 9, 103906–103926. DOI: https://doi.org/10.1109/access.2021.3094024.

Van Huong, P., Thuan, L. D., Hong Van, L. T., & Hung, D. V. (2019, December). Intrusion detection in IoT systems based on deep learning using convolutional neural network. 2019 *6th NAFOSTED Conference on Information and Computer Science (NICS)*. Hanoi, Vietnam. DOI: https://doi.org/10.1109/nics48868.2019.9023871.

Wu, K., Chen, Z., & Li, W. (2018). A novel intrusion detection model for a massive network using convolutional neural networks. *IEEE Access: Practical Innovations, Open Solutions*, 6, 50850–50859. DOI: https://doi.org/10.1109/access.2018.2868993.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9(4), 611–629. DOI: https://doi.org/10.1007/s13244-018-0639-9.

Zhai, F., Yang, T., Chen, H., He, B., & Li, S. (2023). Intrusion detection method based on CNN–GRU–FL in a smart grid environment. *Electronics*, 12(5), 1164. DOI: https://doi.org/10.3390/electronics12051164.

Zhang, J., Zhang, X., Liu, Z., Fu, F., Jiao, Y., & Xu, F. (2023). A network intrusion detection model based on BiLSTM with multi-head attention mechanism. Electronics, 12(19), 4170. DOI: https://doi.org/10.3390/electronics12194170.