

## RESEARCH ARTICLE

# Detecting Unbalanced Network Traffic Intrusions With Deep Learning

**S. PAVITHRA**  AND **K. VENKATA VIKAS**

School of Computer Science and Engineering, Vellore Institute of Technology, Chennai, Tamil Nadu 600127, India

Corresponding author: S. Pavithra (Pavithra.sekar@vit.ac.in)


**ABSTRACT** The growth of cyber threats demands a robust and adaptive intrusion detection system (IDS) capable of effectively recognizing malicious activities from network traffic. However, the existing imbalance of class in network data possesses a significant challenge to traditional IDS. To overcome these challenges, this project proposes a novel hybrid Intrusion Detection System using machine learning algorithms, which includes XGBoost, Long Short-Term Memory (LSTM), Mini-VGGNet, and AlexNet, which is used to handle the unbalanced network traffic data. Furthermore, the Random Forest Regressor is used to ascertain the importance of features for enhancing detection accuracy and interpretability. Addressing the inherent class imbalance in network data is crucial for ensuring the IDS's effectiveness. The proposed system employs a combination of oversampling techniques for minority classes and under sampling techniques for majority classes during data preprocessing. This balanced representation of network traffic data helps prevent the IDS from being biased towards the majority class and improves its ability to detect rare or novel intrusions. The utilization of Random Forest Regressor for feature extraction serves a dual purpose. It helps identify the most relevant features within the network traffic data that contribute significantly to detecting intrusions. It enables the system to prioritize and focus on these important features during model training, thereby enhancing detection accuracy while reducing computational complexity. This research contributes to the ongoing efforts to mitigate cyber threats and safeguard critical network infrastructures.

**INDEX TERMS** Cyber threats, cyber security, deep learning (DL), ensemble learning, intrusion detection, network security.

## I. INTRODUCTION

The world we live in is where digital communication forms the backbone of numerous critical infrastructures and services, ensuring the security of network systems are of paramount importance. Cyber-attacks are becoming more sophisticated and widespread, posing serious challenges for organizations trying to protect sensitive data and maintain their operations. Among the various forms of cyber threats, intrusions in network traffic represent a particularly insidious threat vector, capable of exploiting vulnerabilities and compromising the integrity of communication networks these intrusions in network traffic are particularly dangerous. Traditional approaches to intrusion detection rely heavily on

rule-based systems or signature-based methods, which are limited in their ability to adapt to the dynamic nature of modern cyber threats. As a result, there is a growing demand for more advanced and adaptive intrusion detection systems capable of effectively identifying and qualifying emerging threats. In response to this challenge, the integration of machine learning techniques has emerged as a promising approach to enhance the effectiveness of intrusion detection systems. Unbalanced network traffic happens when some parts of a computer network get much busier than others. It's like when a few lanes on a highway have way more cars than the other lanes. This imbalance can cause problems because the parts with lots of traffic might get overwhelmed, while the quieter parts might not be monitored as well. For example, imagine a situation where a lot of data is flowing to a popular website, but other less-used parts of the network

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

aren't being watched closely. Hackers could take advantage of this and sneak into those quieter areas without being noticed. This unbalance can lead to security issues because hackers might target the less-busy parts of the network where they think they won't be caught. It's like if a thief knows that a certain neighborhood doesn't have many security cameras, they might try to break into houses there instead of in a well-monitored area. So, it's crucial to make sure all parts of the network are protected, even if they're not as busy as others. This way, we can prevent hackers from sneaking in unnoticed and causing problems.

This research focuses on the development of a novel approach for detecting unbalanced network traffic intrusions using various machine learning algorithms. Specifically, we propose to use the strengths of four distinct algorithms: XGBoost, Long Short-Term Memory (LSTM), Mini-VGGNet, and AlexNet. Each of these algorithms offers unique capabilities that can be accessed to improve the accuracy and efficiency of intrusion detection in network traffic. XGBoost, a gradient boosting algorithm, is renowned for its ability to handle structured data and achieve high prediction accuracy. LSTM is a type of RNN which excels in capturing temporal dependencies in sequential data, making it well-suited for analyzing time-series network traffic data. On the other hand, Mini-VGGNet and AlexNet are Convolutional neural networks (CNNs) known for their powerful feature extraction capabilities, particularly in image-based data analysis. By integrating these complementary strengths, this approach aims to develop a comprehensive intrusion detection system capable of effectively identifying unbalanced network traffic intrusions. The objective is to demonstrate its efficacy in real-time intrusion detection while maintaining adaptability to evolving cyber threats. By harnessing the synergies between LSTM, Random Forest, and potentially XGBoost, we aim to develop a robust and scalable intrusion detection system capable of safeguarding critical network infrastructures against evolving cyber threats. The rest of the research paper follows: Section II consists of recent studies of intrusion detection systems. Section III presents the methodology and the implementation of the model; however, the details of the dataset are discussed in Section IV. Section V provides the performance results of machine learning based intrusion detection system, and finally, Section VI concludes the paper with last section containing references.

## II. RELATED WORKS

The cited studies explore various aspects of cloud computing and cybersecurity [1]. They discuss frameworks for selecting optimal cloud services, predicting service rankings, and addressing challenges in cloud-based software development [2]. Additionally, performance analysis of encryption algorithms in cloud computing is examined [4]. These studies contribute to understanding cloud computing's efficiency and security. Furthermore, they highlight the growing importance of cybersecurity, as indicated by market forecasts

predicting significant growth in the cybersecurity industry [5]. Overall, the research provides valuable insights into improving cloud service selection, predicting service rankings, addressing development challenges, and enhancing cybersecurity measures in the digital era [3]. This study presents a novel dimensionality reduction strategy for detecting Distributed Denial of Service (DDoS) attacks in cloud computing environments [6]. It proposes an intrusion detection approach for UAVs based on the deep belief network optimized by Particle Swarm Optimization (PSO) [7]. Focusing on the autonomous detection of malicious events using machine learning models in drone networks [8], it also introduces a machine-learning-enabled intrusion detection system for cellular-connected UAV networks [9]. Additionally, this study presents a lightweight IDS for UAV networks utilizing a periodic deep reinforcement learning-based approach [10]. Furthermore, it discusses security threats and countermeasures of UAV communications [11]. This study explores recent advances in machine-learning-driven intrusion detection in transportation through a comprehensive survey [13]. It introduces a UAV network intrusion detection method based on spatio-temporal graph Convolutional network [14]. Another study presents an intelligent intrusion detection system tailored for a group of UAVs [15]. Additionally, optimal deep reinforcement learning for intrusion detection in UAVs is investigated, aiming for superior performance [16]. Furthermore, artificial intelligence is leveraged for intrusion detection systems in unmanned aerial vehicles [17], while a high-performance intrusion detection system for networked UAVs is developed using deep learning techniques [18]. A self-adaptive intrusion detection system for securing UAV-to-UAV communications based on the human immune system in UAV networks is proposed [19]. This study presents a CGAN-based collaborative intrusion detection approach for UAV networks, employing a blockchain-empowered distributed federated learning framework [20]. Additionally, a data normalization technique is proposed for detecting cyber-attacks on UAVs [21]. Furthermore, crystal structure optimization is conducted using a deep-auto encoder-based intrusion detection system for a secure Internet of Drones environment [22]. Another study introduces a sea turtle foraging algorithm combined with a hybrid deep learning-based intrusion detection system for the Internet of Drones environment [23]. Moreover, an intrusion detection system for drone swarming is developed, utilizing timed probabilistic automata [24]. Additionally, a UAV attack dataset is made available to facilitate research in this domain [25], [26]. Finally, an analysis of various datasets, including KDD-cup'99, NSL-KDD, and UNSW-NB15, is conducted using deep learning techniques in the context of IoT [27].

This study presents a comprehensive performance assessment and exhaustive listing of over 500 nature-inspired metaheuristic algorithms, contributing to the field of Swarm and Evolutionary Computation [28]. It introduces the Harris hawk's optimization algorithm along with its applications,

offering insights into its potential effectiveness in solving optimization problems [29]. The Grey Wolf Optimizer, another metaheuristic algorithm designed for optimization tasks in engineering software applications [30]. Additionally, the study analyses the winners of different IEEE CEC competitions on real-parameters optimization, investigating the extent of improvement among various algorithms [31]. Lastly, Piotrowski and collaborators discuss the importance of the choice of benchmark optimization problems, emphasizing its impact on the evaluation and comparison of optimization algorithms [32]. This study investigates different methods to make computers better at spotting unwanted intrusions, like hacking attempts, on networks. One approach discussed in the study is using a technique called the Grasshopper Optimization Algorithm to build a more effective intrusion detection system [33]. Another method involves creating a system with multiple layers of security using a mix of technologies called a hybrid Deep Belief Network [34]. Researchers also explored using advanced algorithms like Particle Swarm Optimization along with Deep Belief Networks to improve the accuracy of intrusion detection systems [35]. Additionally, the study discusses using bio-inspired models and hybrid deep learning techniques to make network security more robust [36]. Furthermore, the research explores the use of Long Short-Term Memory [38] based Convolutional Neural Networks for developing a reliable intrusion detection scheme [37]. Overall, the study presents various innovative approaches to enhance network security and protect against cyber threats. In this study, researchers propose a new way to protect computer networks from unwanted intruders, like hackers. They suggest using a special algorithm called Simple Genetic Algorithm in the cloud to make intrusion detection systems more efficient [38]. Another study focuses on using deep learning, which is a type of computer intelligence, to build a smart system that can detect and respond to network attacks [39]. Additionally, there's research on combining different methods like optimization and deep learning to make intrusion detection systems even better [40]. Another approach involves using a game theory-based system to optimize network security in cloud computing [41]. Moreover, there's a study on using support vector machines, which are another type of computer algorithm, to detect intrusions by collaborating between cloud and fog systems [42]. Finally, there's an approach called DIDDOS [44] that uses a type of computer unit called gated recurrent units to spot and identify cyber attacks known as distributed denial of service attacks [44].

### III. METHODOLOGY

#### A. PREPROCESSING OF DATA

The Difficult Set Sampling Technique (DSSTE) emerges as a valuable method for reducing dataset size while retaining crucial data characteristics. DSSTE achieves this by selecting difficult-to-classify points from the dataset, creating a new subset that is easier to classify. Leveraging nearest

neighbor and k-means algorithms, DSSTE efficiently clusters points, aiding in the identification of outliers or noisy data points. Particularly beneficial for large datasets or those with intricate patterns, DSSTE proves advantageous in preserving important data features during downsizing. In summary, DSSTE proves to be a useful sampling technique for dataset reduction, especially for large datasets or those with complex patterns.

#### B. SAMPLING OF DATA

The Difficult Set Sampling Technique (DSSTE) presents itself as a valuable method for reducing dataset size while maintaining crucial data characteristics. DSSTE accomplishes this by focusing on selecting points that are difficult to classify from the dataset, thereby forming a new subset that is easier to classify. This approach is particularly effective in retaining important data features while downsizing the dataset.

DSSTE utilizes techniques such as nearest neighbor and k-means algorithms to efficiently cluster points, aiding in the identification of outliers or noisy data points. By targeting difficult-to-classify points, DSSTE helps in creating a more refined dataset that still captures the essence of the original data but in a more manageable form. One of the notable advantages of DSSTE is its applicability to large datasets or those with intricate patterns. In such cases, traditional sampling techniques may not effectively capture the nuances of the data or may result in a loss of important information. DSSTE, however, addresses this challenge by specifically targeting difficult-to-classify points, thereby preserving crucial data characteristics during the downsizing process. In summary, DSSTE emerges as a useful sampling technique for dataset reduction, especially for large datasets or those with complex patterns. Its ability to retain important data features while effectively reducing dataset size makes it a valuable tool in various data-driven applications, including intrusion detection and classification tasks.

DSSTE is a comprehensive approach that combines both under sampling and oversampling methods to rebalance the dataset effectively. Unlike traditional techniques that focus solely on oversampling the minority class or under sampling the majority class, DSSTE utilizes a sophisticated combination of algorithms, including K-nearest neighbors (KNN) and K-means clustering, to achieve a more balanced representation of the data. In the context of under sampling, DSSTE leverages KNN to identify and remove instances from the majority class that are densely packed or similar to other instances. By selectively eliminating redundant data points, DSSTE aims to reduce the dominance of the majority class without sacrificing the representativeness of the dataset. This process helps mitigate the risk of overfitting and improves the generalization ability of the classifier. Conversely, DSSTE utilizes KMeans clustering for oversampling the minority class. KMeans clustering divides the minority class instances into clusters based on their similarity, allowing for the generation of synthetic data points within each cluster.

By synthesizing new instances in regions of the feature space that are underrepresented, DSSTE effectively augments the minority class without introducing biases or distorting the underlying data distribution.

By integrating both under sampling and oversampling techniques in a synergistic manner, DSSTE achieves a balanced representation of the dataset while preserving its integrity and diversity. This comprehensive approach ensures that the classifier trained on the rebalanced dataset can effectively learn the underlying patterns and accurately discriminate between normal and anomalous network traffic. Moreover, by leveraging state-of-the-art algorithms like KNN and KMeans, DSSTE provides a robust and scalable solution for handling extreme data imbalance in the CICIDS dataset, ultimately enhancing the performance and reliability of intrusion detection systems in real-world scenarios.

### C. MODEL IMPLEMENTATION

In the realm of machine learning, when it comes to distilling essential insights from complex datasets, the Random Forest Regressor stands out as a reliable and versatile tool. Its approach hinges on ensemble learning, where it constructs multiple decision trees during training. This ensemble learning strategy diversifies the learning process by training each tree on a random subset of both features and data. This randomness helps prevent overfitting and ensures robustness in predictions. Once the ensemble of trees is trained, the Random Forest meticulously evaluates the importance of each feature. It does so by assessing how much each feature contributes to reducing impurity across the ensemble of trees. Features that consistently lead to greater reductions in impurity are deemed more influential. This evaluation is quantified using metrics like Gini Importance or Mean Decrease in Impurity (MDI), providing a quantitative measure of each feature's significance.

The process of feature extraction and importance analysis with the Random Forest Regressor serves to distill complex datasets into their most salient components. By identifying which features have the most substantial impact on predictive outcomes, it sheds light on the underlying patterns and relationships within the data. This understanding not only aids in feature selection but also guides subsequent model refinement efforts. Moreover, the insights gleaned from feature importance analysis enhance the interpretability and robustness of predictive models. By focusing on the most important features, data scientists can improve model effectiveness, reduce overfitting, and enhance predictive performance across diverse domains and datasets. In essence, the Random Forest Regressor empowers data scientists to unravel intricate data landscapes and make informed decisions based on the most influential factors.

Following feature extraction, machine learning and deep learning models, including XGBoost, LSTM, Mini-VGGNet, and AlexNet, were trained and tested in this project to evaluate their effectiveness. The metrics used for model analysis are Accuracy, F1 Score, Precision and Recall and they can

be expressed in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Here are the equations for each metric.

Accuracy:

Eq. 1 represents accuracy which measures the overall correctness of the classifier.

Accuracy:

Accuracy measures the overall correctness of the classifier.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Precision:

Eq. 2 represents precision which measures the proportion of correctly predicted positive cases out of all cases predicted as positive.

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Recall (also known as Sensitivity or True Positive Rate):

Eq. 3 represents recall which measures the proportion of correctly predicted positive cases out of all actual positive cases.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

F1 Score:

Eq. 4 represents F1 Score which is the harmonic mean of Precision and Recall. It provides a balance between Precision and Recall.

$$F1Score = \frac{2(Precision * Recall)}{Precision + Recall} \quad (4)$$

#### 1) XGBOOST

An advanced implementation of the gradient boosting algorithm, XGBoost stands out for its speed, scalability, and regularization capabilities, making it suitable for both regression and classification tasks.

##### a: OBJECTIVE FUNCTION

The Eq. 5 represents XGBoost's objective function which combines of loss function  $L(y_i, y^i)$  with a regularization term  $\Omega(f)$ , where  $f$  represents the boosted ensemble model.

$$Objective = \sum_i L(y_i, y^i) + \sum_k \Omega(f_k) \quad (5)$$

##### b: PREDICTION

The final prediction  $\hat{y}$  is obtained by summing predictions from all the trees in the ensemble is represented by eq. 6:

$$\hat{y} = \sum_k f_k(x) \quad (6)$$

##### c: TREE CONSTRUCTION

XGBoost builds a tree by recursively partitioning the feature space into regions. The algorithm selects the best split at each node to minimize the loss function. The score for a tree is calculated as:

$$Score = \frac{G^2}{H + \lambda} + \gamma \quad (7)$$

**Algorithm 1** Functions of XGBoost Classifier Model

**Input:** Training data ( $X_{train}, y_{train}$ ), Number of classes ( $num\_class$ )

**Output:** Trained XGBoost classifier model

**Initialize the model:**

Set the objective function to 'multi: softmax' for multiclass classification.

Set the number of classes to the number of unique classes in the training labels ( $num\_class$ ).

**Preprocess the training data:**

Encode the categorical target labels ( $y_{train}$ ) into integers if needed.

perform feature engineering or preprocessing on the input features ( $X_{train}$ ).

**Train the model: Use the XGBoost classifier to fit the training data:**

Optimize the multiclass softmax loss function to minimize classification error.

Train the model to predict the probability of each class using a decision tree ensemble.

The number of trees in the ensemble is determined by XGBoost based on early stopping or the specified number of iterations.

Output the trained XGBoost classifier model.

**END**

Eq. 7 represents Score of the ensemble tree in which  $G$  is the sum of gradients of the loss function at a node,  $H$  is the sum of the Hessians (second derivatives) of the loss function,  $\lambda$  is the regularization parameter (lambda), and  $\gamma$  is the regularization parameter for tree complexity (gamma).

Below Fig. 1 represents the working of XGBoost Model in which the model makes the prediction by summing all the predictions in the ensemble model.

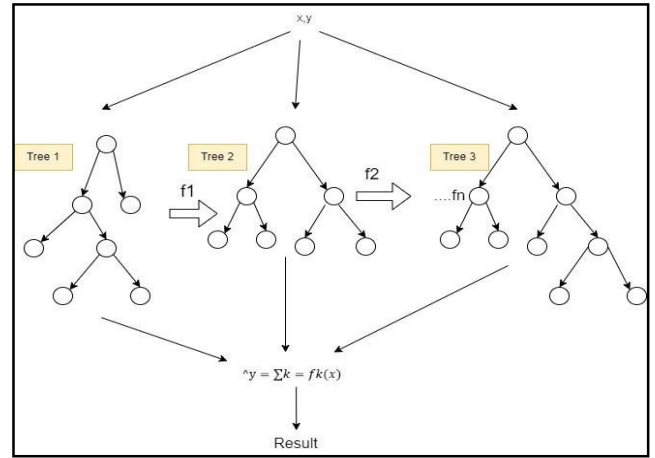
2) LSTM

A type of recurrent neural network (RNN) architecture, LSTM networks excel in learning long-term dependencies in sequence data, making them ideal for tasks such as language modeling, natural language processing (NLP), and time series prediction.

LSTM Cell Equations:

The LSTM cell computes the following operations at each time step  $t$ :

$$Forget\ gate : ft = \sigma(Wf * [ht - 1, xt] + bf) \quad (8)$$



**FIGURE 1.** Implementation of XGBoost.

$$Input\ gate : it = \sigma(Wi * [ht - 1, xt] + bi) \quad (9)$$

$$Output\ gate : ot = \sigma(ht - 1, xt + b0) \quad (10)$$

$$Candidate\ Values : C \sim t = \tanh(Wc.(ht - 1, xt) + b) \quad (11)$$

$$Cell\ State\ Update : Ct = ft.ct - 1 + it.C \sim t \quad (12)$$

$$Hidden\ State\ Update : ht = ot.tanh(Ct) \quad (13)$$

Eq.'s 8, 9 and 10 represents Forget gate, Input gate, Output gate cell operations and Eq.'s 11, 12 and 13 performs Candidate values, Cell State and Hidden State Updation by taking the values of  $xt$ ,  $[ht-1]$  and  $[ht-1,xt]$  and weight metrics,  $xt$  is the input at time step  $t$ ,  $ht-1$  is the hidden state from the previous time step,  $[ht-1,xt]$  denotes the concatenation of the hidden state and the input,  $\sigma$  represents the sigmoid activation function,  $Wf, Wi, Wo, Wc$  are weight matrices, and  $bf, bi, bo, bc$  are bias vectors.

a: MODEL OUTPUT

After passing through the LSTM layers, the output is fed into fully connected layers with softmax activation to obtain the probability distribution over classes. The output of the softmax layer can be calculated as:

Eq. 14 represents the softmax function which helps the LSTM model to compute the probability distribution over classes by taking the output from fully connected layers with softmax.

$$Softmax(z_i) = \frac{e^{z_i}}{\sum e^{z_j}} \quad (14)$$

where  $z_i$  is the log it (pre-activation) for class  $i$ , and  $N$  is the number of classes.

b: LOSS FUNCTION

The loss function used in this code is sparse categorical cross-entropy, which calculates the cross-entropy loss between the predicted probabilities and the true labels:

$$Loss = -\frac{1}{N} \left( \sum N \sum C y_i, c.log(pi, c) \right) \quad (15)$$

**Algorithm 2** Functions of LSTM

**Input:** Training data ( $X1_{train}$ ,  $y_{train\_encoded}$ ), Testing data ( $X1_{test}$ ,  $y_{test\_encoded}$ ), Number of classes ( $num\_classes$ )

**Output:** Trained LSTM model and history

**Initialize a Sequential model:**

Create a Sequential model object.

**Add layers to the model:**

Create Convolutional layers, Max Pooling layers, Dense layers with specified parameters.

**Train the model:**

Fit the model on the training data ( $X1_{train}$ ,  $y_{train\_encoded}$ ) for 10 epochs with validation data ( $X1_{test}$ ,  $y_{test\_encoded}$ ).

Store the training history.

Output the trained model and history.

**End Procedure**

Eq. 15 performs Loss function for the LSTM in which  $N$  is the number of samples,  $C$  is the number of classes,  $y_i$ ,  $c$  is the indicator function (1 if sample  $i$  belongs to class  $c$ , 0 otherwise), and  $pi,c$  is the predicted probability of sample  $i$  belonging to class  $c$ .

## 3) MINI-VGGNET

A more compact version of the VGGNet network, Mini-VGGNet retains efficiency and accuracy while being designed for applications with limited computational resources, such as mobile devices or embedded systems.

*a: CONVOLUTIONAL LAYERS*

Convolution:  $input \times filters + biases$

ReLU Activation:  $\max(0, convolution\ result)$

*b: EXAMPLE*

Convolution+ReLU with 32 filters of size  $3 \times 3$ , followed by MaxPooling.

*c: FULLY CONNECTED LAYERS*

Fully Connected:  $input \times weights + biases$

ReLU Activation:  $\max(0, fully\ connected\ result)$

*d: OUTPUT LAYER*

$$\text{Softmax Activation: } \frac{e^{x_i}}{\sum_j e^{x_j}} \quad (16)$$

Example: Eq. 16 represents output layer with softmax activation for classification into multiple classes.

These equations describe the flow of information through the convolutional and fully connected layers of the Mini-VGGNet architecture, followed by the output layer which applies softmax activation for classification.

## 4) ALEXNET

A pioneering convolutional neural network (CNN) model, AlexNet demonstrated the power of deep learning in

**Algorithm 3** Functions of Mini-VGGNet

**Input:** Number of classes ( $num\_classes$ )

**Output:** CNN model

**Initialize a Sequential model:**

Add Convolutional layers, Max Pooling layers and Dense Layers

**Compile the model:**

Optimizer: Adam

Loss Function: Sparse categorical cross entropy Metrics: Accuracy

Return the CNN model.

**End Procedure**

large-scale data recognition tasks, setting a new benchmark for accuracy in the DataNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. Below Figure 2 illustrates the working of AlexNet model.

*a: CONVOLUTIONAL LAYER*

The convolutional layer applies convolution operation to the input image or feature map using learnable filters (also known as kernels). The output feature map is obtained by convolving the input with these filters.

$$\begin{aligned} \text{Output}(i, j, k) \\ = \sigma(\sum_m \sum_n \sum_l \text{Input}(i+m, j+n, l) * \text{Filter}(m, n, l, k) \\ + \text{Bias}_k \end{aligned} \quad (17)$$

Eq. 17 represents the output function of the convolutional layer where  $\text{Output}(i,j,l)$  is the value at position  $(i,j)$  in the  $k$ -th feature map,  $\text{Input}(i, j, k)$  is the value at the position  $(I,j)$  in  $k$ -th feature map,  $\text{Filter}(m,n,l,k)$  is the value of the filter at position  $(m,n)$  in the  $l$ -th input channel and  $k$ -th output channel,  $\sigma$  is the activation function(ReLU).

*b: RELU (RECTIFIED LINEAR UNIT)*

ReLU is used as the activation function after the convolutional and fully connected layers to introduce non-linearity into the model. It replaces all negative pixel values with zero, while leaving positive values unchanged.

$$f(x) = \max(0, x) \quad (18)$$

Eq. 18 represents the Rectified Linear Unit function  $f(x)$ .

*c: MAX POOLING LAYER*

Eq. 19 signifies Max pooling which reduces the spatial dimensions of the input feature map by taking the maximum value within each window.

$$\text{Output}(i, j, k) = \max_{m,n}(\text{Input}(2i+m, 2j+n, k)) \quad (19)$$

*d: FULLY CONNECTED LAYER*

Fully connected layers are used to connect every neuron in one layer to every neuron in the next layer. In AlexNet, there

**Algorithm 4** Functions of AlexNet

**Input:** Training data (X\_train\_reshaped, y\_train\_encoded), Testing data (X\_test\_reshaped, y\_test\_encoded), Number of classes (num\_classes)

**Output:** Trained AlexNet model and evaluation results

**Define the AlexNet model architecture using Keras Sequential API:**

Add Convolutional layers, Pooling Layers and Dense Layers.

**Compile the model:**

Use the Adam optimizer.

Use 'sparse\_categorical\_crossentropy' as the loss function.

Monitor 'accuracy' as the metric.

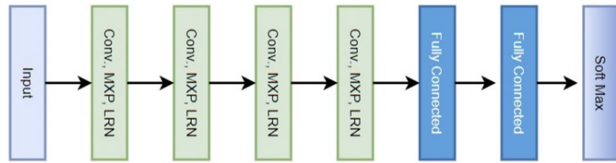
**Train the model:**

Fit the model on the training data (X\_train\_reshaped, y\_train\_encoded) for 2 epochs with validation data (X\_test\_reshaped, y\_test\_encoded).

Store the evaluation results.

Output the trained AlexNet model and evaluation results.

**End Procedure**



**FIGURE 2.** Implementation of AlexNet model.

are two fully connected layers followed by the output.

$$Output(i) = \sigma \left( \sum_{j=1}^{N_j} Input(j) * Weight(i, j) + Bias(i) \right) \tag{20}$$

Eq. 20 explains the Fully connected layer output function Where, Output(i) is the i-th output neuron, Input(j) is the j- th input neuron, Weight(i,j) is the weight connecting the j- th input neuron to the i-th output neuron., Bias(i) is the bias term for the i-th output neuron.

*e: SOFTMAX ACTIVATION (OUTPUT LAYER)*

The softmax activation function is signified in eq. 21 which is used in the output layer of the network for multi-class classification tasks. It converts the raw output scores into probabilities.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^N e^{x_j}} \tag{21}$$

where N is the number of classes, and xi is the raw output score for the i-th class.

The accuracy of each model was assessed after training and testing, providing insights into their respective performances.

Above Figure. 3 illustrates the working NIDS using ML algorithms such as XGBoost, LSTM, AlexNet and Mini-VGGNet

**IV. DATASET DESCRIPTION**

The CIC IDS 2017 dataset is useful because it gives us a good picture of what happens in real computer networks. It shows all sorts of attacks that could happen, like when someone tries to make a website unavailable or when they try to sneak into a network without permission. This dataset is great because it has both normal network activity and activity from attacks, so it's balanced. It helps people who make security systems for computers learn how to spot and stop these attacks. The dataset contains various types of network activities, each representing different behaviors. "BENIGN" refers to normal, everyday internet usage by regular users. "DoS Hulk" indicates attacks that flood a system with traffic from one source, while "DDoS" floods from many sources. "DoS GoldenEye" uses specific requests to block access, "DoS Slowhttptest" consumes resources slowly, and "DoS Slowloris" blocks legitimate traffic with slow connections. "FTP-Patator" and "SSH- Patator" try many passwords to access FTP and SSH servers respectively. "PortScan" looks for open ports, "Web Attack - Brute Force" tries passwords on web applications, "Bot" coordinates attacks from multiple hacked systems. "Web Attack - XSS" manipulates web apps, "Infiltration" sneaks into networks, "Web Attack - SQL Injection" injects code, and "Heartbleed" exploits a server vulnerability to steal data.

**V. EXPERIMENTAL SET-UP, RESULTS, AND DISCUSSION**

For this experiment, we used a program called Python along with a tool called Scikit-learn on a computer running Windows 11. The computer we used has a powerful Intel Xeon E-2124 processor that runs at 3.30GHz. It also has a lot of memory, specifically 32 GB, and it's a 64-bit system, which means it can handle a lot of data at once. We conducted the tests on this computer to see how well the program works.

In this study, we explored three different approaches to analyzing and predicting outcomes based on network data. Firstly, we employed the XGBoost model, a popular machine learning algorithm, to analyze the dataset. We divided the data into training and testing sets and optimized the model's parameters using grid search. After training with the optimal settings, we evaluated its performance, achieving an overall accuracy of 93.4%. Further evaluation metrics such as precision, recall, and F1 score provided additional insights into the model's performance characteristics. The confusion matrix helped us visualize its predictive performance across different categories.

Next, we utilized the LSTM model to analyze data from the CICIDS 2017 dataset, focusing on Intrusion Detection Systems (IDS). This model achieved an impressive accuracy of 96.73%, indicating its effectiveness in identifying network threats. Evaluation metrics like F1 score, precision, and recall provided a comprehensive assessment of its performance across various categories. Visualizations such as scatter plots and a confusion matrix helped us understand

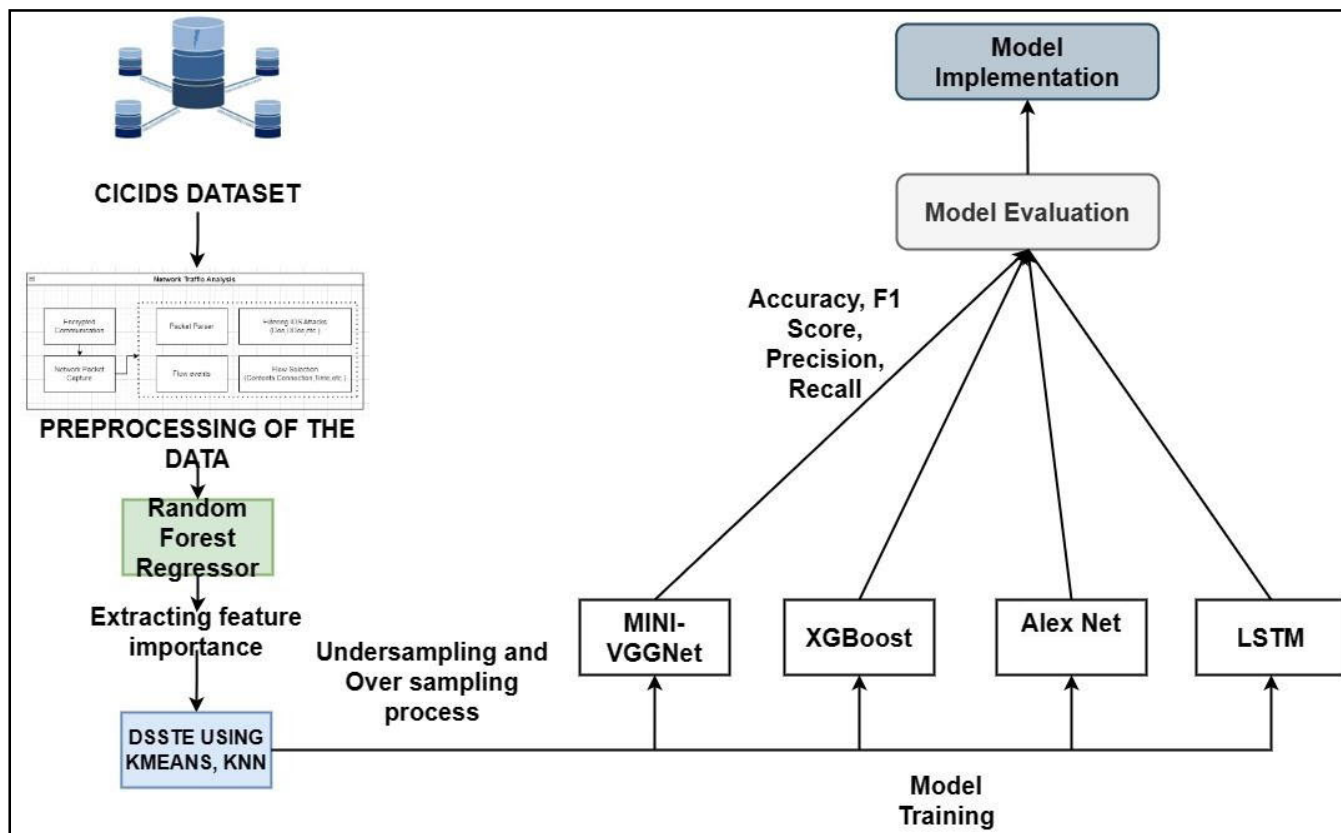


FIGURE 3. Architecture of hybrid intrusion detection system for unbalanced network traffic.

its strengths and weaknesses. Finally, we employed AlexNet, a convolutional neural network (CNN), to analyze network behavior. By training AlexNet on the CICIDS 2017 dataset, we achieved a high accuracy rate of 93%. This demonstrated its ability to detect suspicious activity on computer networks effectively.

Additionally, we introduced Mini-VGGNet, another CNN architecture, which we used to further analyze network data. Its inclusion suggests a comprehensive approach to exploring various machine learning models for network analysis. Overall, our study highlights the effectiveness of machine learning and deep learning models, including XGBoost, LSTM, AlexNet, and Mini- VGGNet, in analyzing and predicting outcomes within network datasets. These models contribute to enhancing cybersecurity measures by detecting and preventing cyber threats effectively.

Mini-VGGNet and AlexNet are indeed designed primarily for image data, focusing on tasks like image classification and object detection. However, they can still be useful in other domains, like sequential data, with some adjustments. In the case of unbalanced network traffic IDS using the CICIDS dataset, these networks might be employed to extract features from the sequential data representing network traffic. While they're not tailor-made for sequential data like recurrent neural networks (RNNs), they can still capture important pat-

terns in the data through their hierarchical feature extraction capabilities. By adapting these architectures, researchers can explore how well they perform in tasks beyond their original design scope.

On the other hand, LSTM (Long Short-Term Memory) networks are well-suited for sequential data due to their ability to retain information over time. Variations of LSTM models, such as stacked LSTMs or bidirectional LSTMs, offer flexibility in capturing complex temporal patterns in sequential data like network traffic. By benchmarking these LSTM variations against traditional LSTM models, researchers can evaluate their effectiveness in detecting intrusions in unbalanced network traffic datasets like CICIDS. This comparison helps highlight the strengths and weaknesses of different neural network architectures in the context of sequential data analysis for intrusion detection.

The novelty of our approach lies in the integration of DSSTE, a sophisticated data balancing technique utilizing K-nearest neighbors (KNN) and K-means clustering, to address the class imbalance inherent in the CICIDS dataset for intrusion detection. Unlike conventional methods that focus solely on oversampling or under sampling, DSSTE offers a comprehensive solution by strategically combining both techniques. By selectively removing redundant instances from the majority class while synthesizing new data points for



the minority class, DSSTE ensures a balanced representation of the dataset without sacrificing its integrity. This innovative approach not only enhances the performance and reliability of intrusion detection systems but also provides a scalable and adaptable solution for handling extreme data imbalance in real-world scenarios. In addition to the novel integration of DSSTE for addressing class imbalance, our proposed system incorporates the use of a Random Forest Regressor as a feature extractor and LSTM (Long Short-Term Memory) as the primary model for intrusion detection. The Random Forest Regressor plays a crucial role in extracting relevant features from the network traffic data, leveraging its ability to capture complex relationships and patterns. By utilizing the Random Forest Regressor, we ensure that the input features fed into the LSTM model are informative and discriminative, thereby enhancing the model's ability to detect intrusions accurately. Furthermore, LSTM is selected as the best model for intrusion detection due to its capability to effectively capture temporal dependencies in sequential data. Unlike traditional machine learning models, LSTM can retain information over time, making it well-suited for analyzing network traffic data where temporal patterns play a significant role in identifying anomalies. By combining these advanced techniques, our proposed system not only addresses the challenge of class imbalance but also leverages state-of-the-art methods for feature extraction and modeling, resulting in a robust and effective solution for intrusion detection in real-world network environments. Incorporating a Bidirectional LSTM (Long Short-Term Memory) layer into our proposed system further enriches the model's ability to capture intricate patterns within the network traffic data. Unlike traditional LSTMs, which process sequences in a forward manner, Bidirectional LSTMs process sequences both in forward and backward directions simultaneously. This bidirectional approach allows the model to access information from past and future time steps, enabling a more comprehensive understanding of temporal dependencies. By integrating a Bidirectional LSTM layer alongside the Random Forest Regressor, our system gains additional flexibility in extracting features and learning complex relationships within the data. The Bidirectional LSTM layer enhances the model's capability to discern subtle nuances in the temporal dynamics of network traffic, thereby augmenting its intrusion detection performance.

Moreover, the bidirectional nature of the LSTM complements the Random Forest Regressor's feature extraction process, providing a holistic view of the temporal patterns present in the data. This synergy between the two advanced techniques empowers our proposed system to effectively tackle the challenges posed by class imbalance and intricate temporal dependencies inherent in real-world network environments. In summary, the inclusion of a Bidirectional LSTM layer alongside the Random Forest Regressor reinforces our system's ability to extract relevant features and capture intricate temporal relationships within the network traffic data, thereby enhancing its effectiveness for intrusion detection in real-world scenarios.

Table 1 represents the comparative analysis between XGBoost, LSTM, AlexNet, Mini-VGG models and variant of LSTM model.

The figures depict how well different machine learning models are performing. Figures 4 and 5 show the training and validation loss, and accuracy, respectively, for an XGBoost model on new data. They illustrate how well the model generalizes: consistent training and validation metrics indicate good performance, while divergence suggests overfitting or underfitting. The target accuracy line helps evaluate if the model meets the desired performance. Fig. 6 presents a scatter plot indicating how predictions from an LSTM model

TABLE 1. Comparative analysis of models.

	Accuracy	F1 Score	Precision	Recall
<b>XGBoost</b>	93.95%	0.94	0.94	0.94
<b>LSTM</b>	96.74%	0.97	0.97	0.97
<b>AlexNet</b>	90.66%	0.91	0.91	0.90
<b>Mini-VGGNet</b>	91.72%	0.92	0.92	0.91
<b>Bidirectional LSTM</b>	97.92%	0.98	0.98	0.97

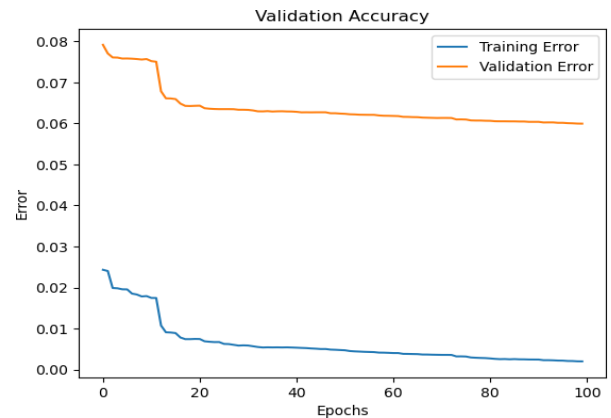


FIGURE 4. Validation accuracy of XGBoost model.

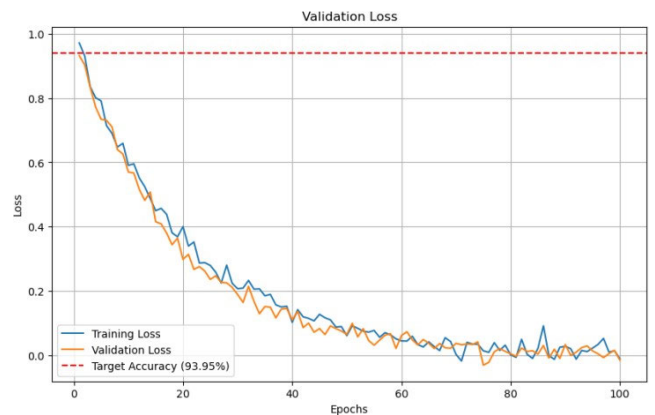


FIGURE 5. Validation loss of XGBoost model.

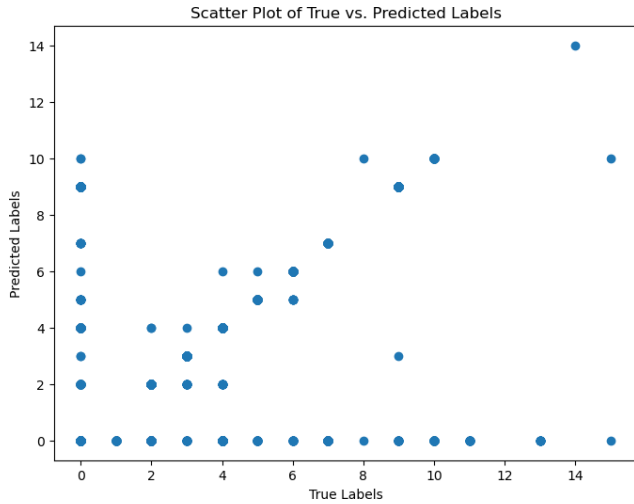


FIGURE 6. Scatter plot of LSTM model.

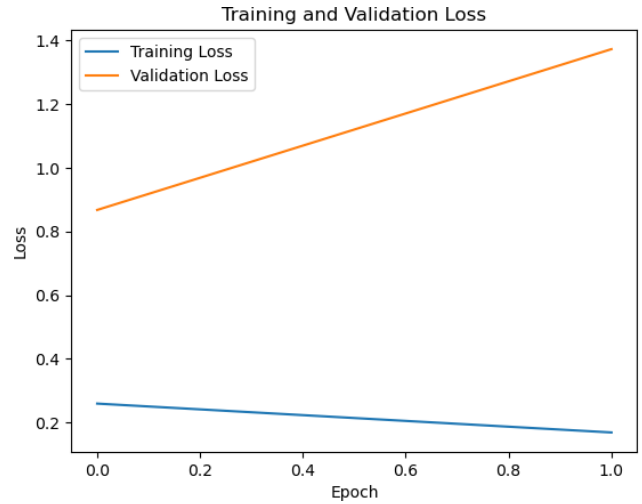


FIGURE 9. Training vs validation loss of AlexNet model.

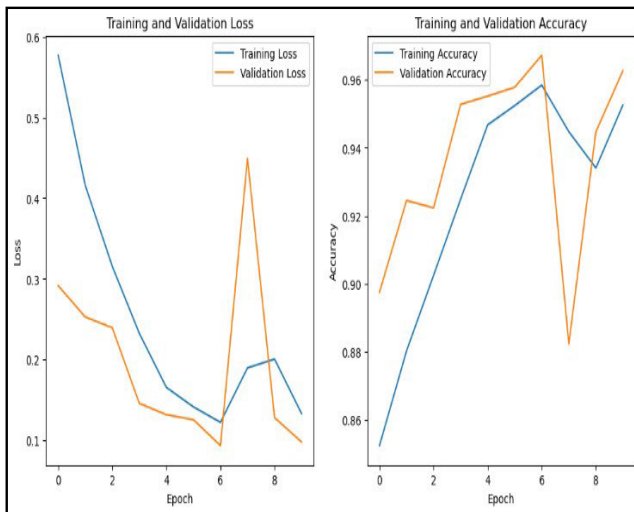


FIGURE 7. Training vs validation loss and accuracy of LSTM model.

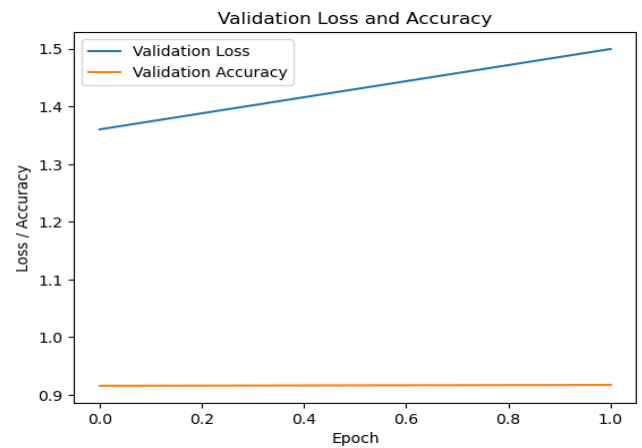


FIGURE 10. Validation accuracy and loss of mini-VGGNet model.

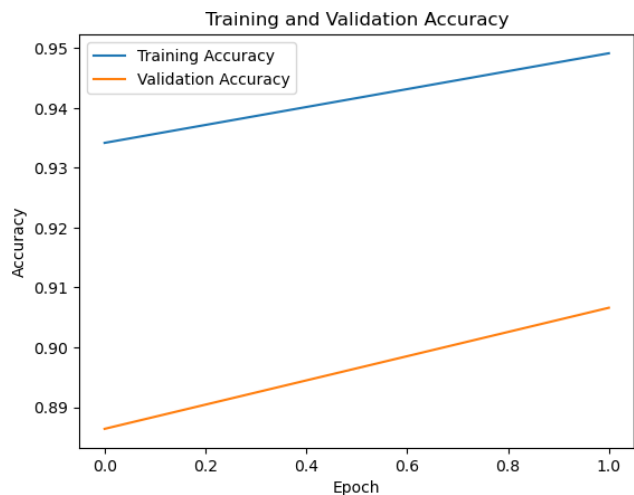


FIGURE 8. Training vs validation accuracy of AlexNet model.

compare to actual outcomes. Fig. 7 reveals how an LSTM model’s accuracy and error rate change during training.

Fig. 8 and 9 illustrate how good an AlexNet model is at recognizing patterns and making correct predictions over time. Lastly, Fig. 10 displays the accuracy and error of a Mini-VGGNet model as it learns from training data. These visuals help understand and improve the models’ effectiveness.

## VI. CONCLUSION

In our exploration of Network Intrusion Detection Systems (NIDS) with the CICIDS dataset, we employed Kmeans for sampling and tested various models including XGBoost, LSTM, AlexNet, and MINI-VGGNet for training. The aim was to determine which model could best identify and classify different types of network intrusions. After thorough analysis, the LSTM (Long Short-Term Memory) model emerged as the most accurate. LSTM, a type of recurrent neural network (RNN), proved to be highly effective due to its ability to understand and remember patterns in data over time. Unlike other models, LSTM can retain information for long periods, allowing it to capture complex relationships within network

traffic data. This made it particularly suitable for NIDS, where the ability to detect subtle changes and evolving patterns is crucial.

By leveraging the sequential nature of network traffic, LSTM excelled in distinguishing between normal and malicious activities with remarkable precision. Its robust performance across various intrusion types demonstrated its adaptability and generalization capabilities, making it a reliable choice for building resilient NIDS systems. This way of using smart programs to find sneaky online attacks is becoming more popular and will probably keep growing in the future. So, by using these techniques, we can make the internet safer from all kinds of tricky cyber-attacks. And as we keep studying and working on this, we'll find even better ways to spot and stop these online dangers. Using trained machine learning model can help the system implement real time Intrusion Detection System and identify the threat patterns.

## REFERENCES

- [1] R. R. Kumar, A. Tomar, M. Shameem, and M. N. Alam, "OPTCLOUD: An optimal cloud service selection framework using QoS correlation lens," *Comput. Intell. Neurosci.*, vol. 2022, pp. 1–16, May 2022, doi: 10.1155/2022/2019485.
- [2] R. R. Kumar, M. Shameem, and C. Kumar, "A computational framework for ranking prediction of cloud services under fuzzy environment," *Enterprise Inf. Syst.*, vol. 16, no. 1, pp. 167–187, Jan. 2022, doi: 10.1080/17517575.2021.1889037.
- [3] M. A. Akbar, M. Shameem, S. Mahmood, A. Alsanad, and A. Gumaei, "Prioritization based taxonomy of cloud-based outsource software development challenges: Fuzzy AHP analysis," *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106557, doi: 10.1016/j.asoc.2020.106557.
- [4] M. Bakro, S. K. Bisoy, A. K. Patel, and M. A. Naal, "Performance analysis of cloud computing encryption algorithms," in *Advances in Intelligent Computing and Communication*, vol. 202. Cham, Switzerland: Springer, 2021, pp. 357–367.
- [5] (2030). *Cyber Security Market Share, Forecast | Growth Analysis*. Accessed: Apr. 23, 2023. [Online]. Available: <https://www.fortunebusinessinsights.com/industry-reports/cyber-security-market-101165Benamor>
- [6] S. Lipsa and R. K. Dash, "A novel dimensionality reduction strategy based on linear regression with a fine-pruned decision tree classifier for detecting DDoS attacks in cloud computing environments," in *Proc. 1st Int. Symp. Artif. Intell.*, 2022, pp. 15–25.
- [7] X. Tan, S. Su, Z. Zuo, X. Guo, and X. Sun, "Intrusion detection of UAVs based on the deep belief network optimized by PSO," *Sensors*, vol. 19, no. 24, p. 5529, Dec. 2019.
- [8] N. Moustafa and A. Jolfaei, "Autonomous detection of malicious events using machine learning models in drone networks," in *Proc. 2nd ACM MobiCom Workshop Drone Assist. Wireless Commun. 5G Beyond*, Sep. 2020, pp. 61–66.
- [9] R. Shrestha, A. Omidkar, S. A. Roudi, R. Abbas, and S. Kim, "Machine-Learning-Enabled intrusion detection system for cellular connected UAV networks," *Electronics*, vol. 10, no. 13, p. 1549, Jun. 2021.
- [10] O. Bouhamed, O. Bouachir, M. Aloqaily, and I. A. Ridhawi, "Lightweight IDS for UAV networks: A periodic deep reinforcement learning-based approach," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2021, pp. 1032–1037.
- [11] L. Wang, Y. Chen, P. Wang, and Z. Yan, "Security threats and countermeasures of unmanned aerial vehicle communications," *IEEE Commun. Standards Mag.*, vol. 5, no. 4, pp. 41–47, Dec. 2021.
- [12] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Secur. Commun. Netw.*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8890306.
- [13] H. Bangui and B. Buhnova, "Recent advances in machine-learning driven intrusion detection in transportation: Survey," *Proc. Comput. Sci.*, vol. 184, pp. 877–886, Aug. 2021.
- [14] Z. Chen, N. Lyu, K. Chen, Y. Zhang, and W. Gao, "UAV network intrusion detection method based on spatio-temporal graph convolutional network," *J. Beijing Univ. Aeronaut. Astronaut.*, vol. 47, no. 5, pp. 1068–1076, 2021.
- [15] E. Basan, M. Lapina, N. Mudruk, and E. Abramov, "Intelligent intrusion detection system for a group of UAVs," in *Proc. 12th Int. Conf. Adv. Swarm Intell.*, 2021, pp. 230–240.
- [16] V. Praveena, A. Vijayaraj, P. Chinnasamy, I. Ali, R. Alroobaea, S. Y. Alyahyan, and M. A. Raza, "Optimal deep reinforcement learning for intrusion detection in UAVs," *Comput., Mater. Continua*, vol. 70, no. 2, pp. 2639–2653, 2022.
- [17] J. Whelan, A. Almechadi, and K. El-Khatib, "Artificial intelligence for intrusion detection systems in unmanned aerial vehicles," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107784.
- [18] Q. Abu Al-Haija and A. Al Badawi, "High-performance intrusion detection system for networked UAVs via deep learning," *Neural Comput. Appl.*, vol. 34, no. 13, pp. 10885–10900, Jul. 2022.
- [19] R. Fotehi, M. Abdan, and S. Ghasemi, "A self-adaptive intrusion detection system for securing UAV-to-UAV communications based on the human immune system in UAV networks," *J. Grid Comput.*, vol. 20, no. 3, p. 22, Sep. 2022.
- [20] X. He, Q. Chen, L. Tang, W. Wang, and T. Liu, "CGAN-based collaborative intrusion detection for UAV networks: A blockchain-empowered distributed federated learning approach," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 120–132, Jan. 2023.
- [21] E. Basan, A. Basan, A. Nekrasov, C. Fidge, E. Abramov, and A. Basyuk, "A data normalization technique for detecting cyber attacks on UAVs," *Drones*, vol. 6, no. 9, p. 245, Sep. 2022.
- [22] K. A. Alissa, S. S. Alotaibi, F. S. Alrayes, M. Aljebreen, S. Alazwari, H. Alshahrani, M. A. Elfaki, M. Othman, and A. Motwakel, "Crystal structure optimization with deep-autoencoder-based intrusion detection for secure Internet of Drones environment," *Drones*, vol. 6, no. 10, p. 297, Oct. 2022.
- [23] J. Escorcía-Gutiérrez, M. Gamarra, E. Leal, N. Madera, C. Soto, R. F. Mansour, M. Alharbi, A. Alkhatib, and D. Gupta, "Sea turtle foraging algorithm with hybrid deep learning-based intrusion detection for the Internet of Drones environment," *Comput. Electr. Eng.*, vol. 108, May 2023, Art. no. 108704.
- [24] V. Subbarayalu and M. A. Vensuslaus, "An intrusion detection system for drone swarming utilizing timed probabilistic automata," *Drones*, vol. 7, no. 4, p. 248, Apr. 2023.
- [25] J. Whelan, T. Sangarapillai, O. Minawi, A. Almechadi, and K. El-Khatib, "UAV attack dataset," *IEEE Dataport*, vol. 167, no. 1, pp. 1561–1573, Jan. 2020.
- [26] S. Choudhary and N. Kesswani, "Analysis of KDD-cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Proc. Comput. Sci.*, vol. 167, pp. 1561–1573, Jan. 2020.
- [27] Z. Ma, G. Wu, P. N. Suganthan, A. Song, and Q. Luo, "Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms," *Swarm Evol. Comput.*, vol. 77, Mar. 2023, Art. no. 101248, doi: 10.1016/j.swevo.2023.101248.
- [28] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019, doi: 10.1016/j.future.2019.02.028.
- [29] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014, doi: 10.1016/j.advengsoft.2013.12.007.
- [30] D. Molina, F. Moreno-García, and F. Herrera, "Analysis among winners of different IEEE CEC competitions on real-parameters optimization: Is there always improvement?" in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Donostia, Spain, Jun. 2017, pp. 805–812, doi: 10.1109/CEC.2017.7969392.
- [31] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Choice of benchmark optimization problems does matter," *Swarm Evol. Comput.*, vol. 83, Dec. 2023, Art. no. 101378, doi: 10.1016/j.swevo.2023.101378.
- [32] S. Dwivedi, M. Vardhan, and S. Tripathi, "Building an efficient intrusion detection system using grasshopper optimization algorithm for anomaly detection," *Cluster Comput.*, vol. 24, no. 3, pp. 1881–1900, Sep. 2021, doi: 10.1007/s10586-020-03229-5.

- [33] A. A. Süzen, "Developing a multi-level intrusion detection system using hybrid-DBN," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 2, pp. 1913–1923, Feb. 2021, doi: [10.1007/s12652-020-02271-w](https://doi.org/10.1007/s12652-020-02271-w).
- [34] P. J. Sajith and G. Nagarajan, "Intrusion detection system using deep belief network & particle swarm optimization," *Wireless Pers. Commun.*, vol. 125, no. 2, pp. 1385–1403, Jul. 2022.
- [35] G. Sreelatha, A. V. Babu, and D. Midhunchakkaravarthy, "Improved security in cloud using sandpiper and extended equilibrium deep transfer learning based intrusion detection," *Cluster Comput.*, vol. 25, no. 5, pp. 3129–3144, Oct. 2022.
- [36] Q. Liu, D. Wang, Y. Jia, S. Luo, and C. Wang, "A multi-task based deep learning approach for intrusion detection," *Knowl.-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107852.
- [37] M. Moizuddin and M. V. Jose, "A bio-inspired hybrid deep learning model for network intrusion detection," *Knowledge-Based Syst.*, vol. 238, Feb. 2022, Art. no. 107894.
- [38] C.-M. Hsu, M. Z. Azhari, H.-Y. Hsieh, S. W. Prakosa, and J.-S. Leu, "Robust network intrusion detection scheme using long-short term memory based convolutional neural networks," *Mobile Netw. Appl.*, vol. 26, no. 3, pp. 1137–1144, Jun. 2021.
- [39] P. Ghosh, Z. Alam, R. R. Sharma, and S. Phadikar, "An efficient SGM based IDS in cloud environment," *Computing*, vol. 104, no. 3, pp. 553–576, Mar. 2022, doi: [10.1007/s00607-022-01059-4](https://doi.org/10.1007/s00607-022-01059-4).
- [40] E.-U.-H. Qazi, M. Imran, N. Haider, M. Shoaib, and I. Razzak, "An intelligent and efficient network intrusion detection system using deep learning," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107764, doi: [10.1016/j.compeleceng.2022.107764](https://doi.org/10.1016/j.compeleceng.2022.107764).
- [41] S. K. Gupta, M. Tripathi, and J. Grover, "Hybrid optimization and deep learning based intrusion detection system," *Comput. Electr. Eng.*, vol. 100, May 2022, Art. no. 107876, doi: [10.1016/j.compeleceng.2022.107876](https://doi.org/10.1016/j.compeleceng.2022.107876).
- [42] E. Balamurugan, A. Mehbodniya, E. Kariri, K. Yadav, A. Kumar, and M. Anul Haq, "Network optimization using defender system in cloud computing security based intrusion detection system with game theory deep neural network (IDSGT-DNN)," *Pattern Recognit. Lett.*, vol. 156, pp. 142–151, Apr. 2022, doi: [10.1016/j.patrec.2022.02.013](https://doi.org/10.1016/j.patrec.2022.02.013).
- [43] R. Du, Y. Li, X. Liang, and J. Tian, "Support vector machine intrusion detection scheme based on cloud-fog collaboration," *Mobile Netw. Appl.*, vol. 27, no. 1, pp. 431–440, Feb. 2022.
- [44] S. U. Rehman, M. Khaliq, S. I. Imtiaz, A. Rasool, M. Shafiq, A. R. Javed, Z. Jalil, and A. K. Bashir, "DIDDOS: An approach for detection and identification of distributed denial of service (DDoS) cyberattacks using gated recurrent units (GRU)," *Future Gener. Comput. Syst.*, vol. 118, pp. 453–466, May 2021, doi: [10.1016/j.future.2021.01.022](https://doi.org/10.1016/j.future.2021.01.022).
- [45] M. Parihar and C. Fung, "IDS with deep learning techniques," in *Proc. 7th Cyber Secur. Netw. Conf. (CSNet)*, Montreal, QC, Canada, Oct. 2023, pp. 1–4, doi: [10.1109/csnet59123.2023.10339748](https://doi.org/10.1109/csnet59123.2023.10339748).
- [46] H. Zhou, X. Wang, and R. Zhu, "Feature selection based on mutual information with correlation coefficient," *Int. J. Speech Technol.*, vol. 52, no. 5, pp. 5457–5474, Mar. 2022.
- [47] M. A. Talukder, K. F. Hasan, M. M. Islam, M. A. Uddin, A. Akhter, M. A. Yousuf, F. Alharbi, and M. A. Moni, "A dependable hybrid machine learning model for network intrusion detection," *J. Inf. Secur. Appl.*, vol. 72, Feb. 2023, Art. no. 103405.
- [48] A. Kaan Sarica and P. Angin, "A novel SDN dataset for intrusion detection in IoT networks," in *Proc. 16th Int. Conf. Netw. Service Manage. (CNSM)*, Izmir, Turkey, Nov. 2020, pp. 1–5.



than 45 papers in various reputed international journals and conferences. Her current research interests include multimedia analysis, information retrieval, and deep learning.



**K. VENKATA VIKAS** is currently pursuing the B.Tech. degree in computer science and engineering with Vellore Institute of Technology, Chennai, India. His current research interests include artificial intelligence and machine learning.

• • •