iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.71013

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Enhancing Cyber Security to Predict DDOS Attack Using Deep Learning Algorithms

Dr.R.G. Suresh Kumar[1], Dr.S. Udhayashree[2], A. Sathiya[3], K. Subashshree[4], P. Abirami[5]
[1]Head of the Department, [2]Assistant Professor, [3, 4, 5]UG, Dept of CSE, Rajiv Gandhi College of Engineering and Technology, Puducherry, India

Abstract: ***Artificial Intelligence (AI) is a technology that allows machines to think andlearn like humans. It can perform tasks like human intelligence, such as understandinglanguage, recognizing images, solving problems, and making decisions. In our existing system, they proposed a new approach detect DDOS attacks by using Artificial Intelligence concepts. Such as Convolutional Neural Networks (CNNs) combined with adaptive architectures and transfer learning techniques At Particularly, they use the Convolutional Neural Networks (CNNs) to shown promise in detecting and mitigating DDOS attacks by automatically learning feature representations and identifying complex patterns in network traffic data. Even though the existing work is effective, there are still some issues such as High Computational Cost, Struggling with Large Datasets, Time Consuming Training, Limited Use for Other cyber threats. For these issues, we proposed to find solutions to develop a hybrid model that combines multiple algorithms or techniques to enhance the prediction accuracy of DDOS attack detection system. We use cloud resources that can reduce costs and improve processing speed and we will use deep learning models effectively to handle large datasets, for time consuming we planned to split the work across several computers so that the training happens faster. Using multi-task learning, combining models and focusing on common features can help the model to detect a wider range of threats.***

Keywords: Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), hybrid algorithm, Distributed Denial of Service (DDoS).

I. INTRODUCTION

A Distributed Denial of Service (DDoS) attack is a cyberattack that aims to disrupt the normal operation of a server, network, or website by overwhelming it with excessive traffic from multiple sources. Unlike a Denial of Service (DoS) attack, which originates from a single source, a DDoS attack involves a network of compromised devices, known as a botnet, that attackers control remotely. These infected devices, often unaware of their involvement, simultaneously flood the target system with a massive volume of requests, consuming its bandwidth, memory, or processing power. As a result, legitimate users experience slow responses or complete inaccessibility. DDoS attacks can be categorized into volume-based attacks, which flood bandwidth with data, protocol attacks, which exploit network vulnerabilities, and application-layer attacks, which target specific applications like web servers or databases. Common attack techniques include UDP floods, SYN floods, HTTP floods, and DNS amplification. To mitigate these threats, organizations deploy firewalls, rate limiting, Content Delivery Networks (CDNs), load balancers, and DDoS protection services like Cloudflare or AWS Shield. Since DDoS attacks can lead to severe financial losses, downtime, and reputational damage, businesses must implement robust cybersecurity strategies to detect and neutralize such threats effectively.

A. Gated Recurrent Unit (GRU):

A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) architecture designed to handle sequential data while addressing the vanishing gradient problem. Introduced as a simpler alternative to Long Short-Term Memory (LSTM) networks, GRU improves learning efficiency by using fewer parametersA GRU, or Gated Recurrent Unit, operates using two primary mechanisms: the update gate and the reset gate. The reset gate is responsible for deciding the extent to which previous information should be discarded, while the update gate regulates how much of the new input should influence the current memory. Unlike LSTM networks, which use distinct gates for input, output, and forgetting, GRUs streamline this process by merging functionalities, resulting in improved computational efficiency and simpler architecture. When processing sequential data, GRU selectively retains relevant information while discarding unnecessary details, allowing it to excel in tasks like natural language processing (NLP), speech recognition, and time-series forecasting. Due to its ability to capture long-term dependencies without excessive computational complexity, GRU is often preferred for applications requiring real-time predictions.

Its simpler structure enables faster training and reduced memory consumption compared to LSTM, making it suitable for resource-constrained environments. However, while GRU performs well in many scenarios, LSTM may still be preferred for tasks requiring finer control over long-term dependencies. In summary, GRUs strike an effective compromise between computational simplicity and predictive power, which makes them well-suited for a wide range of deep learning tasks.

B. Long- Short Term Memory(LSTM)

The Long Short-Term Memory (LSTM) network is a specialized form of recurrent neural network (RNN) developed to process sequential information and address the issue of vanishing gradients. Unlike traditional RNNs, which struggle to retain long-term dependencies, LSTM incorporates memory cells and three key gates: the input gate, forget gate, and output gate. These gates regulate the flow of information, allowing the network to selectively store, update, or discard information over long sequences. The forget gate determines which previous information should be discarded, the input gate selects the new information to be added to the memory cell, and the output gate regulates the extent to which the stored data affects the present output. This architecture enables LSTM to capture long-range dependencies, making it highly effective in tasks like natural language processing (NLP), speech recognition, time-series forecasting, and handwriting recognition. Although LSTM is powerful, it requires more computational resources compared to simpler models like Gated Recurrent Units (GRUs). While GRUs offer faster training and reduced complexity, LSTM provides greater control over memory retention, making it ideal for tasks requiring long-term context understanding. Due to its ability to process complex sequences, LSTM remains a fundamental model in deep learning applications.

II. LITERATURE SURVEY

1) Flexible SDN-Based Architecture For Identifying And Mitigating Low-Rate Ddos Attacks Using Machine Learning. This study presents a dynamic SDN-oriented framework aimed at detecting and countering Low-Rate Distributed Denial of Service (LR-DDoS) attacks, which continue to be a significant challenge in software-defined networking environments. The proposed architecture is modular and adaptable, focusing on the identification and suppression of LR-DDoS threats. An intrusion detection system (IDS), trained with six different machine learning algorithms, demonstrates a strong detection accuracy of 95%, highlighting its capability to handle the subtle nature of such attacks. The solution is implemented using the Open Network Operating System (ONOS) controller within a Mininet simulation environment, effectively emulating the behavior and conditions of real-world production networks. The testing topology reveals that the intrusion prevention detection system successfully mitigates all previously detected attacks, highlighting the practical utility of the proposed architecture in addressing LR-DDoS threats. The modular and flexible design of the architecture enables easy replacement of individual modules without affecting the overall functionality, enhancing adaptability and scalability in SDN environments[1].

2) Cochain-SC: An Intra- and Inter-Domain Ddos Mitigation Scheme Based on Blockchain Using SDN and Smart Contract. In response to the escalating impact of Distributed Denial-of-Service (DDoS) attacks, this paper introduces Cochain-SC, a novel blockchain-based approach for cost-effective and flexible DDoS mitigation. Operating at both intra-domain and inter-domain levels, the proposed scheme leverages software-defined networks (SDN) and smart contracts on the Ethereum blockchain. Intra-domain defense utilizes entropy-driven techniques and Bayesian approaches to detect and suppress unauthorized traffic within a single network domain. Inter-domain collaboration is facilitated through smart contracts, allowing multiple SDN-based domains to securely share attack information in a decentralized manner. Cochain-SC uniquely combines SDN, blockchain, and smart contract technologies to achieve efficient mitigation along the attack path and near the attack origin.The implementation is deployed on Ethereum's official test network Ropsten, marking a pioneering effort in addressing both intra- and inter-domain DDoS mitigation challenges in a unified framework[2].

3) Hybrid Ddos Detection Framework Using Matching Pursuit Algorithm. In addressing the persistent threat of Distributed Denial-of-Service (DDoS) attacks, this study proposes a novel detection framework utilizing the Matching Pursuit algorithm, particularly focusing on resource depletion type DDoS attacks. The method concurrently considers multiple network traffic characteristics, enhancing efficiency in detecting low-density DDoS attacks. The approach employs a dictionary generated from network traffic parameters using the K-SVD algorithm, providing adaptable models for both legitimate and attack traffic. Comparative evaluations with Matching Pursuit and Wavelet techniques, along with the introduction of a hybrid detection framework incorporating artificial neural networks, showcase the proposed method's superior performance. Achieving over 99% true positive rates and a false positive rate below 0.7% in low-density DDoS attack datasets, the study introduces the AMP method, offering a unique approach to DDoS detection by utilizing the Matching Pursuit algorithm.

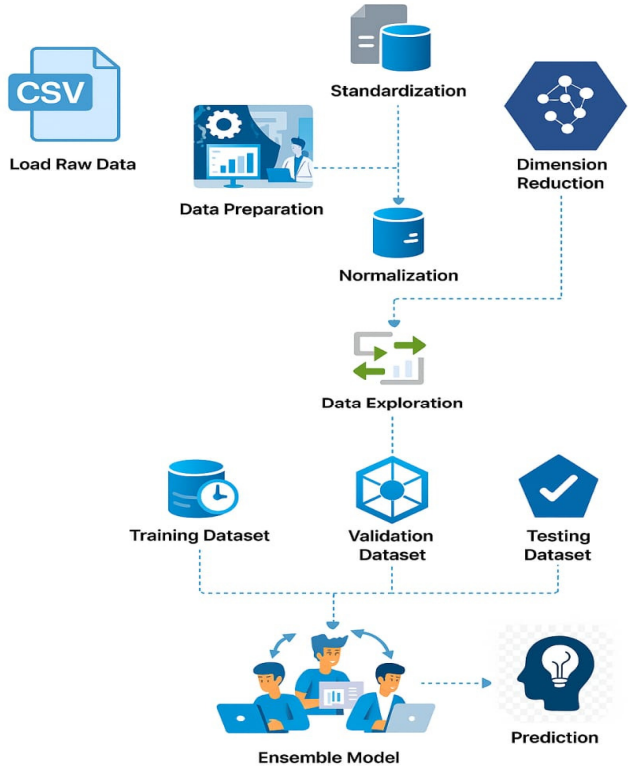
The proposed methodology is compared with existing methods, demonstrating its effectiveness in the detection of DDoS attacks, especially in low-density scenarios where resource depletion is a significant concern[3].

4) DDoS Attack Detection Method Based on Improved KNN With the Degree of DDoS Attack in Software-Defined Networks. This paper explores the potential of Software Defined Networking (SDN) as a defense mechanism against Distributed Denial of Service (DDoS) attacks. Two methods are proposed for DDoS detection in SDN: one utilizes the degree of DDoS attack, and the other employs an improved K-Nearest Neighbors (KNN) algorithm based on Machine Learning (ML). Theoretical analysis and experimental results on datasets demonstrate the effectiveness of these methods in comparison to other existing approaches. Recognizing DDoS as a significant threat in SDN networks, the paper introduces four features and a novel concept called the "degree of attack." Two detection algorithms, DDADA and DDAML, are presented and show superior detection rates in experiments compared to existing solutions. The authors acknowledge the anonymous reviewers for their valuable input and express plans to enhance and apply the proposed algorithms in real SDN environments in future work[4].

5) Identifying Application-Layer DDoS Attacks Based on Request Rhythm Matrices. This paper addresses the growing threat of Application-layer Distributed Denial of Service (AL-DDoS) attacks on websites, which often evade traditional intrusion prevention systems. A novel statistical model, the Rhythm Matrix (RM), is proposed to detect AL-DDoS attacks by abstracting and accumulating access trajectories, including requested objects and dwell-time values. The RM efficiently compresses complex features into a simple structure, characterizing user access behavior. Abnormality degrees in the RM are utilized to detect AL-DDoS attacks, and malicious hosts are identified based on change-rate outliers. Simulated attacks using popular DDoS tools demonstrate the method's accurate and efficient detection. The proposed approach also exhibits excellent performance in distinguishing flash crowds from AL-DDoS attacks, with a True Positive Rate (TPR) exceeding 99% and a False Positive Rate (FPR) below 1%. The precision and recall of identifying malicious hosts approach 100% with parameter optimization, and simulation attacks are detected well in advance of their impact. Results on modified datasets further illustrate the method's strong performance in flash crowd scenarios[5].

A. Architectural Diagram:

An architectural diagram visually outlines how the components of a software system are physically arranged. It illustrates the overall framework of the system, including the relationships, constraints, andinteractions among its various parts.



An architectural diagram serves as a visual blueprint that outlines the physical arrangement and interactions of components within a software system. This graphical representation provides a high-level overview of the system's structure, showcasing the relationships, constraints, and boundaries between various elements. It helps stakeholders, including developers and designers, understand the overall design and organization of the software, highlighting key components, their interactions, and the overall flow of data or control. Architectural diagrams play a crucial role in communicating the system's design principles and serve as a reference for making informed decisions during the development and maintenance phases, contributing to a shared understanding among team members and facilitating effective collaboration.

III. PROPOSED SYSTEM

The proposed system integrates Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks to create a hybrid deep learning model for DDoS attack detection in Software-Defined Networking (SDN) environments. GRU is known for its computational efficiency in processing sequential data, while LSTM excels in capturing long-term dependencies within network traffic patterns. By combining these two architectures, the model enhances feature extraction, improves detection of complex attack behaviors, and reduces false alarms. Trained on the CICDDoS2019 dataset, the hybrid model demonstrates higher accuracy, better adaptability, and improved generalization in detecting diverse and evolving DDoS threats. This hybrid approach significantly enhances the detection and mitigation of DDoS attacks, ensuring network stability and security in SDN environments. By leveraging the strengths of both GRU and LSTM, the proposed model effectively handles high-dimensional network traffic data while maintaining computational efficiency. It enables real-time monitoring, allowing network administrators to detect and respond to threats more effectively. The ability of the system to analyze sequential attack patterns ensures better identification of sophisticated and multi-vector DDoS attacks, improving overall cybersecurity resilience.

A. Gated Recurrent Unit (GRU) Layers:

A Gated Recurrent Unit (GRU) consists of two key layers: the reset gate and the update gate, which work together to control the flow of information in sequential data processing. The reset gate determines how much past information should be forgotten, allowing the model to discard irrelevant details from previous time steps. If the reset gate value is low, most of the past information is ignored, whereas a high value retains important past context.

1) Reset Gate:

The reset gate in a Gated Recurrent Unit (GRU) controls how much of the past hidden state should be forgotten before computing the new candidate activation. This mechanism helps the model determine whether past information is relevant for the current state. If the reset gate value is close to zero, the past hidden state is largely ignored, allowing the model to prioritize recent input data. Conversely, if the reset gate value is close to one, more historical information is retained, enabling the model to capture dependencies over long sequences.

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

where:

- r_t is the reset gate activation at time step t.
- σ is the sigmoid activation function, which outputs values between 0 and 1.
- W_r and U_r are weight matrices for the input x_t and the previous hidden state h_{t-1}, respectively.
- b_r is the bias term.

By modulating the influence of past states, the reset gate ensures that GRUs adapt dynamically to changing input sequences, making them efficient for tasks like speech recognition, machine translation, and time-series analysis.

2) Update Gate:

The update gate in a Gated Recurrent Unit (GRU) is responsible for determining how much of the previous hidden state should be retained and how much of the new candidate state should be incorporated. This gate plays a crucial role in controlling the information flow through the network, ensuring that relevant past knowledge is preserved while allowing the model to adapt to new inputs. If the update gate value is close to 1, the past hidden state is retained, enabling the model to remember long-term dependencies.

Conversely, if the update gate value is close to 0, the model prioritizes the newly computed state, allowing for quick adaptation to changing patterns in sequential data. This dynamic gating mechanism helps GRUs overcome the vanishing gradient problem while maintaining computational efficiency.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

where:

- z_t is the update gate activation at time step t.
- σ is the sigmoid activation function, which outputs values between 0 and 1.
- W_z and U_z are weight matrices for the input x_t and the previous hidden state h_{t-1}, respectively.
- b_z is the bias term.

By controlling the balance between past and new information, the update gate allows GRUs to selectively retain context over long sequences, making them particularly effective for applications like natural language processing, speech recognition, and time-series forecasting. This mechanism ensures that GRUs maintain memory without requiring the complex memory cell structure of Long Short-Term Memory (LSTM) networks.

3) Final Hidden State:

The final hidden state in a Gated Recurrent Unit (GRU) is the combination of the previous hidden state and the newly computed candidate state, regulated by the update gate. This mechanism ensures that the network selectively retains useful past information while incorporating new input. The update gate determines how much of the previous hidden state should be carried forward, allowing the GRU to capture long-term dependencies efficiently. If the update gate value is close to 1, the past hidden state is largely preserved, maintaining historical information.

$$h_t = (1 - z_t) \odot \tilde{h}_t + z_t \odot h_{t-1}$$

where:

- h_t is the final hidden state at time step t.
- z_t is the update gate value that controls memory retention.
- $\tilde{h}_t$ is the candidate hidden state, computed using the reset gate and new input.
- h_{t-1} is the previous hidden state.
- $\odot$ represents the element-wise multiplication.

This equation effectively blends past and new information based on the update gate's decision. When, z_t is high, the hidden state remains similar to h_{t-1}, preserving long-term dependencies. When zt is low, the model prioritizes the newly computed state $h\sim t$, allowing quick adaptation to changes. This gating mechanism makes GRUs efficient in handling sequential data while reducing computational complexity compared to LSTMs.

B. Long Short-Term Memory (LSTM) Layers:

Long Short-Term Memory (LSTM) layers are a type of recurrent neural network (RNN) architecture designed to overcome the vanishing gradient problem. They achieve this by using a specialized memory cell and three gating mechanisms: forget gate, input gate, and output gate. These gates regulate the flow of information, ensuring that relevant data is retained while irrelevant data is discarded. LSTMs are highly effective for processing sequential data, making them widely used in natural language processing, speech recognition, and time-series forecasting. Their ability to capture long-term dependencies makes them more powerful than traditional RNNs.

1) Forget gate:

The forget gate in a Long Short-Term Memory (LSTM) network is responsible for deciding which information from the previous cell state should be discarded.

This mechanism is crucial for preventing unnecessary accumulation of outdated information, allowing the model to retain only relevant context over long sequences. The forget gate examines both the previous hidden state and the current input, applying a sigmoid activation function to produce values between 0 and 1. A value close to 0 means forgetting the information, while a value close to 1 means retaining it. This adaptive approach helps LSTMs manage long-term dependencies effectively, making them suitable for sequential tasks like speech processing and time-series prediction.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

where:

- f_t is the forget gate activation at time step t.
- σ is the sigmoid activation function.
- W_f and U_f are weight matrices for the input x_t and previous hidden state h_{t-1}, respectively.
- b_f is the bias term.

The forget gate plays a crucial role in controlling memory updates within LSTM cells. If the forget gate outputs a value near 0 for a particular piece of information, that data is effectively removed from the memory cell. Conversely, if the output is close to 1, the information is retained for future time steps. This selective forgetting process enables LSTMs to maintain relevant long-term context while efficiently managing computational resources. By dynamically adjusting memory retention, the forget gate ensures the stability and efficiency of LSTMs in complex sequence modeling tasks.

2) final hidden state:

The final hidden state in an LSTM represents the processed information at the last time step, combining the current input and past dependencies.

$$h_t = o_t \odot \tanh(c_t)$$

where:

- h_t is the final hidden state.
- o_t is the output gate activation.
- c_t is the current cell state.
- $\odot$ denotes element-wise multiplication.

It is determined by the output gate, which decides how much of the updated cell state should be exposed. The hidden state enables the model to pass relevant information through time while filtering unnecessary details.

C. Data Collection

Data collection is the first step in building a machine learning model, where relevant data is gathered from various sources such as databases, APIs, sensors, or publicly available datasets. The quality and quantity of data significantly impact model performance. Collected data should be diverse and representative to avoid biases. In real-world scenarios, data can be structured (e.g., tables), semi-structured (e.g., JSON, XML), or unstructured (e.g., images, audio, text). The goal is to obtain sufficient, high-quality data that accurately represents the problem domain. Proper documentation and ethical considerations, such as data privacy, must also be ensured.

D. Pre-processing

Pre-processing involves cleaning and transforming raw data into a format suitable for analysis. This step includes handling missing values, removing duplicates, normalizing numerical features, encoding categorical variables, and filtering noise. In text data, pre-processing includes tokenization, stemming, and stopword removal, while in images, it may involve resizing and noise reduction. Standardization and normalization help maintain consistency across different data points, ensuring that the model learns efficiently. Proper pre-processing enhances model accuracy by eliminating irrelevant or inconsistent information.

E. Feature Extraction

Feature extraction involves selecting and transforming raw data into meaningful representations that improve model performance. It helps reduce dimensionality while preserving important information. In natural language processing (NLP), feature extraction includes word embeddings like Word2Vec or TF-IDF. For images, convolutional filters extract patterns, and in audio, Mel-frequency cepstral coefficients (MFCCs) are used. Selecting relevant features improves learning efficiency, reducing overfitting and computational costs. Techniques like Principal Component Analysis (PCA) further help in dimensionality reduction while preserving essential data characteristics.

F. Model Creation

Model creation involves selecting an appropriate machine learning algorithm and training it on pre-processed data. This step includes choosing between supervised, unsupervised, and reinforcement learning models based on the problem type. Algorithms such as decision trees, neural networks, or support vector machines (SVMs) are commonly used. The model learns patterns from training data by adjusting internal parameters to minimize errors. Hyperparameter tuning techniques like grid search or Bayesian optimization enhance model performance. The trained model is then validated using techniques such as cross-validation to assess generalizability.

G. Test Data

Test data is a separate dataset used to evaluate the trained model's performance on unseen samples. This step ensures that the model generalizes well rather than memorizing patterns from training data. A typical data split involves 70% for training, 20% for validation, and 10% for testing. Performance metrics such as accuracy, precision, recall, F1-score, and mean squared error (MSE) assess model effectiveness. Overfitting detection ensures that the model does not perform well only on training data but also on real-world scenarios.

H. Prediction

Prediction is the final step, where the trained model is deployed to make real-time predictions on new data. The model takes input features and outputs predictions based on learned patterns. In classification problems, it assigns probabilities to different classes, while in regression, it outputs continuous values. The accuracy of predictions is monitored over time, and retraining may be necessary as new data becomes available. In practical applications, prediction models are integrated into real-world systems such as recommendation engines, fraud detection systems, or autonomous vehicles.

IV. RESULT AND DISCUSSION

The results of the proposed hybrid GRU-LSTM model demonstrate its effectiveness in DDoS attack detection within SDN environments. By leveraging GRU's efficiency in processing sequential data and LSTM's strength in capturing long-term dependencies, the system achieves high accuracy in distinguishing normal traffic from malicious activity. Compared to traditional ML-based models, which often struggle with identifying evolving attack patterns, this deep learning approach provides better adaptability and generalization to dynamic threats. The model also maintains a low false positive rate, reducing unnecessary security alerts that could overwhelm network administrators.

Performance evaluation metrics, including accuracy, precision, recall, and F1-score, indicate a significant improvement in attack detection capabilities. The confusion matrix analysis further confirms that the model effectively minimizes misclassification errors, ensuring reliable intrusion detection. Additionally, its ability to process high-dimensional network traffic data in real time makes it highly suitable for deployment in large-scale SDN infrastructures. The system not only detects known attack signatures but also identifies anomalous traffic behaviors that indicate emerging threats, enhancing its proactive defense capabilities.

Future enhancements could incorporate reinforcement learning for adaptive threat response and attention mechanisms to further refine feature extraction. These improvements would enhance the model's scalability and robustness, making it a future-proof cybersecurity solution. Overall, the proposed GRU-LSTM model presents a highly efficient, scalable, and accurate approach for real-time DDoS attack detection, strengthening network security against increasingly sophisticated cyber threats.

A. *Accuracy*

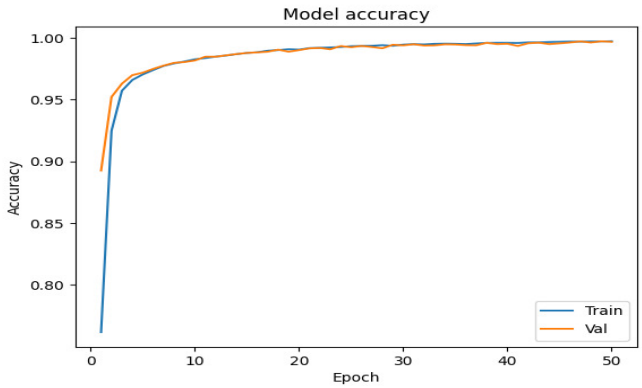
Accuracy is a key performance metric used to evaluate the effectiveness of a classification model, particularly in DDoS attack detection.It calculates the ratio of accurately identified samples—whether legitimate or harmful—compared to the overall number of samples in the dataset.

A high accuracy score indicates that the model effectively differentiates between benign and attack traffic, making it a reliable cybersecurity solution. However, accuracy alone may not always reflect the model's true performance, especially in imbalanced datasets where one class significantly outweighs the other.

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

where:

- **TP (True Positives):** Correctly classified attack instances
- **TN (True Negatives):** Correctly classified normal traffic
- **FP (False Positives):** Normal traffic misclassified as attacks
- **FN (False Negatives):** Attacks misclassified as normal traffic



While accuracy is a useful metric, it is crucial to complement it with precision, recall, and F1-score to get a more comprehensive evaluation, especially in cybersecurity applications. For example, a model with high accuracy but many false negatives may still fail to detect critical attacks, making the system vulnerable. Therefore, when assessing DDoS detection models, security researchers often analyze the confusion matrix and balance accuracy with recall and precision to ensure the model performs well across different traffic scenarios.

B. *Loss*

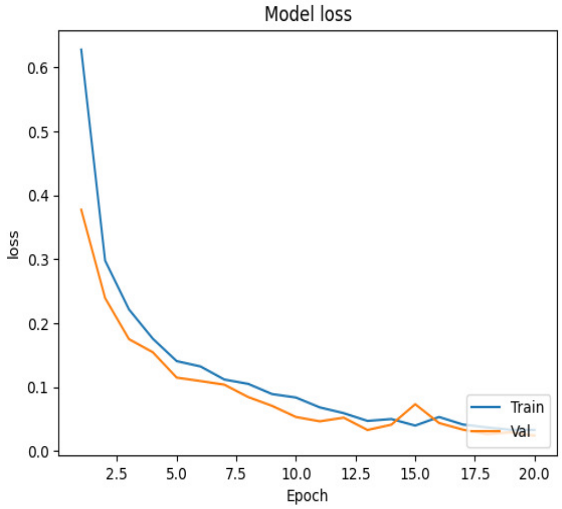
Loss function is a critical component in training deep learning models, including the GRU-LSTM hybrid model for DDoS attack detection. It quantifies how far the model's predictions deviate from the actual labels, guiding the optimization process to improve accuracy. A lower loss value indicates a better-fitting model, while a high loss suggests poor generalization to unseen data. Loss functions are essential for updating model weights during training, ensuring the model learns meaningful patterns from network traffic data.

One commonly used loss function in binary classification problems like DDoS detection is Binary Cross-Entropy (BCE), calculated as:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where:

- y_i is the actual class label (0 for normal traffic, 1 for attack)
- $\hat{y}_i$ is the predicted probability of the instance being an attack
- N is the total number of samples



This function penalizes incorrect predictions by assigning higher loss values when the predicted probability deviates significantly from the true class. Minimizing loss is crucial to improving model performance, and this is done using optimization algorithms like Adam or SGD (Stochastic Gradient Descent). However, solely focusing on reducing loss can sometimes lead to overfitting, where the model performs well on training data but poorly on new, unseen traffic. To address this, techniques such as dropout regularization, batch normalization, and early stopping can be used to balance loss minimization and model generalization, ensuring robust DDoS attack detection.

C. Precision

Precision is a key metric that measures the accuracy of positive (attack) predictions made by a model. It evaluates how many of the instances predicted as attacks were actually attacks, helping to reduce false alarms in cybersecurity applications. High precision indicates that the model minimizes false positives, which is critical in network security to prevent unnecessary alerts.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

where:

- **True Positives (TP):** Correctly predicted attack instances.
- **False Positives (FP):** Normal traffic incorrectly classified as an attack.

While high precision is important, it must be balanced with recall to avoid missing real attacks. If precision is increased by reducing false positives, it may also reduce recall (missing true attacks). The F1-score, which combines precision and recall, helps achieve this balance in DDoS detection systems.

D. Recall

Recall is a key metric in evaluating the effectiveness of a DDoS attack detection model. It measures how well the system identifies actual attack instances from network traffic. A high recall score indicates that most of the attacks are correctly detected, reducing the chances of malicious traffic being overlooked. In cybersecurity, missing an attack can have serious consequences, such as network downtime, data breaches, or service disruption. Therefore, recall is particularly important in high-risk environments, where detecting every possible threat is a priority.

A model with high recall ensures that very few attacks go undetected. However, this often comes at the cost of false positives, where normal traffic is mistakenly classified as an attack. Although false positives can be bothersome, they are often more acceptable than false negatives, which can result in threats going undetected.. The challenge in designing an effective DDoS detection model is to strike a balance between high recall and minimizing false alarms. This balance is essential for ensuring that security teams can focus on real threats without being overwhelmed by excessive alerts.

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

where:

- **True Positives (TP):** Correctly identified attacks.
- **False Negatives (FN):** Actual attacks that were misclassified as normal traffic.

E. F1 SCORE

The F1-score is a crucial metric for evaluating the performance of a DDoS attack detection model, as it balances both precision and recall. Precision measures how many detected attacks are actually malicious, while recall measures how many actual attacks were correctly identified. In cybersecurity, relying on just one of these metrics can lead to an incomplete assessment. A model with high precision but low recall may miss many attacks, while a model with high recall but low precision may generate too many false alarms. The F1-score provides a harmonized measure that ensures a balance between these two factors.

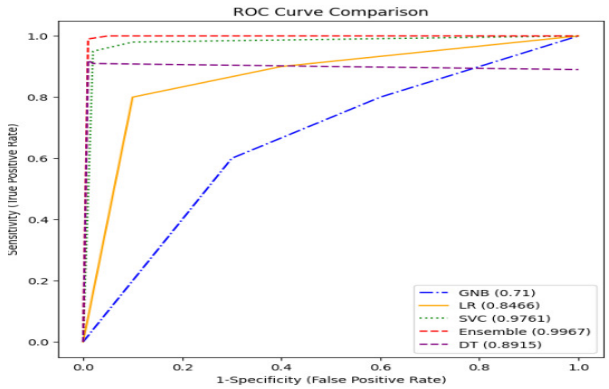
$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

- **Precision** = $\frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$
  (Measures how many predicted attacks are actual attacks)
- **Recall** = $\frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$
  (Measures how many actual attacks were correctly identified)

A high F1-score indicates that the model is not only detecting most attack instances (high recall) but also correctly identifying malicious traffic without excessive false positives (high precision). This is particularly important in real-time DDoS detection, where both false negatives (missed attacks) and false positives (unnecessary alerts) can impact network security and operational efficiency. A well-optimized GRU-LSTM model ensures an improved F1-score by effectively distinguishing between normal and attack traffic, minimizing both misclassifications and false alarms.

F. Comparison Graph



The ROC (Receiver Operating Characteristic) curve comparison graph visualizes the performance of different machine learning models in distinguishing between positive and negative classes. The x-axis illustrates the rate of false positives (calculated as 1 minus specificity), while the y-axis reflects the rate of true positives (also known as sensitivity). A model performs more effectively when its curve is positioned nearer to the upper-left corner of the graph.. In this case, the ensemble model (red dashed line) performs the best, achieving an Area Under the Curve (AUC) of 0.9967, indicating almost perfect classification. The SVC (green dotted line) also performs well with an AUC of 0.9761, followed by the decision tree (purple dashed line) at 0.8915 and logistic regression (orange solid line) at 0.8466. The Gaussian Naïve Bayes (blue dash-dot line) has the lowest AUC of 0.71, indicating it is the least effective among the models compared.

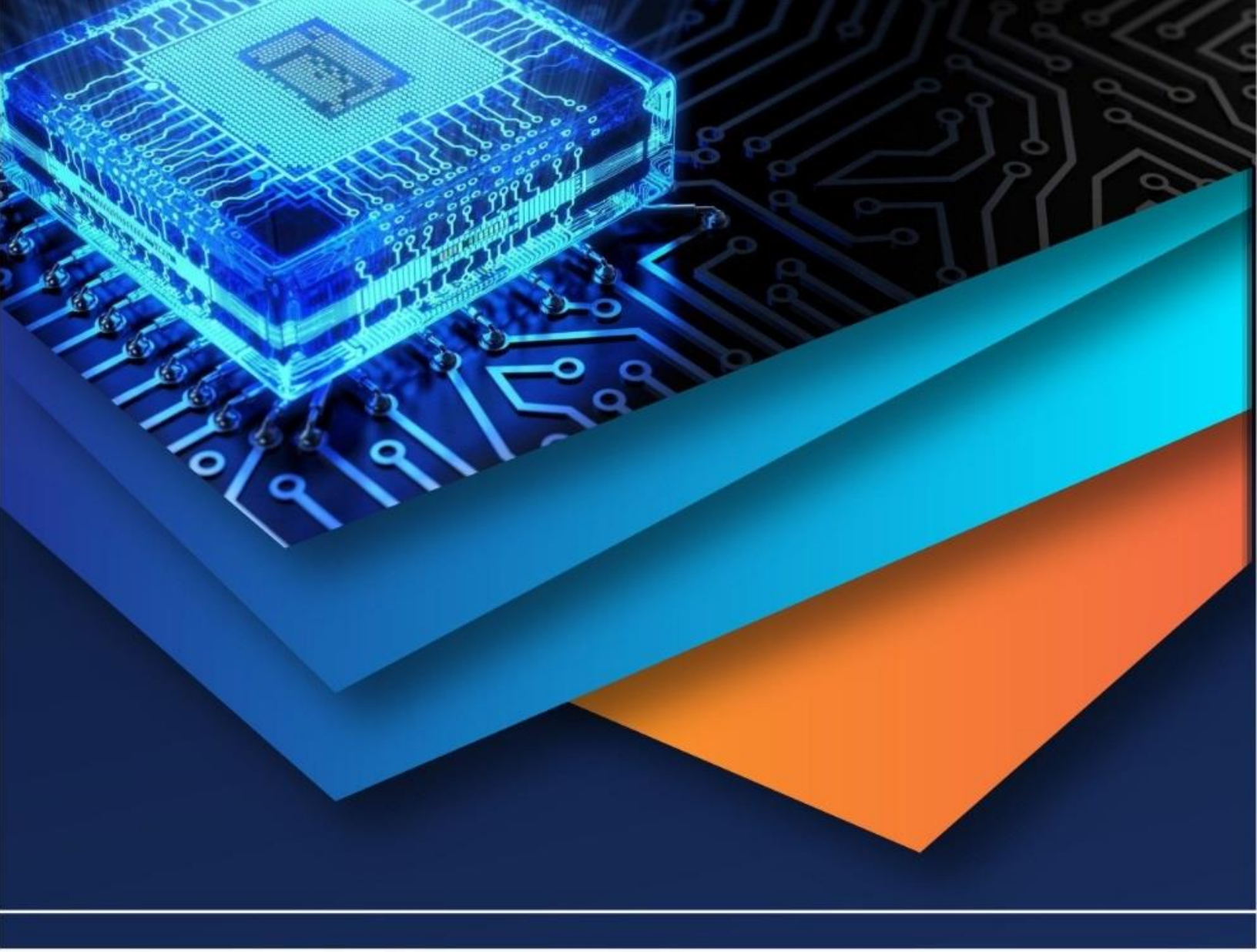
V. CONCLUSION

The rise in Distributed Denial of Service (DDoS) attacks poses a severe threat to Software-Defined Networking (SDN), making traditional Machine Learning (ML) models ineffective due to their inability to adapt to evolving attack patterns. These models often struggle with outdated training data, high false-positive rates, and limited scalability, making zero-day attack detection challenging. To overcome these limitations, a hybrid deep learning approach integrating Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) networks is proposed. By combining GRU's efficiency in handling sequential data with LSTM's ability to capture long-term dependencies, the model enhances feature extraction and effectively detects complex attack patterns. Training the model on the CICDDoS2019 dataset ensures higher accuracy, improved adaptability, and reduced false alarms, making it more suitable for SDN environments. This hybrid approach significantly strengthens intrusion detection systems (IDS) by ensuring network stability and security. Future improvements may include reinforcement learning for adaptive attack prevention, better model scalability, and enhanced real-time mitigation mechanisms. By leveraging GRU and LSTM's strengths, the system provides a robust and computationally efficient solution to counter DDoS attacks, ensuring a resilient cybersecurity framework against evolving threats. Further research is needed to enhance the model's adaptability to emerging attack patterns, ensuring its effectiveness against evolving cybersecurity threats. Exploring the integration of advanced real-time monitoring techniques and adaptive learning mechanisms can contribute to the continuous improvement of the algorithm's responsiveness. Additionally, efforts should be directed toward scalability, enabling the algorithm to handle large-scale network environments efficiently. Collaboration with cybersecurity experts and industry practitioners can provide valuable insights for practical implementation and validation of the hybrid model in diverse network settings.

REFERENCES

[1] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, ''Adversarial machine learning applied to intrusion and malware scenarios: A systematic review,'' IEEE Access, vol. 8, pp. 35403–35419, 2020.
[2] G. Karatas, O. Demir, and O. K. Sahingoz, ''Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset,'' IEEE Access, vol. 8, pp. 32150–32162, 2020.
[3] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, ''BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset,'' IEEE Access, vol. 8, pp. 29575–29585, 2020.
[4] H. Jiang, Z. He, G. Ye, and H. Zhang, ''Network intrusion detection based on PSO-xgboost model,'' IEEE Access, vol. 8, pp. 58392–58401, 2020.
[5] A. Nagaraja, U. Boregowda, K. Khatatneh, R. Vangipuram, R. Nuvvusetty, and V. S. Kiran, ''Similarity based feature transformation for network anomaly detection,'' IEEE Access, vol. 8, pp. 39184–39196, 2020.

[6] L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, ''Classification hardness for supervised learners on 20 years of intrusion detection data,'' IEEE Access, vol. 7, pp. 167455–167469, 2019
[7] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, ''An adaptive ensemble machine learning model for intrusion detection,'' IEEE Access, vol. 7, pp. 82512–82521, 2019.
[8] Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, ''Network intrusion detection based on supervised adversarial variational auto-encoder with regularization,'' IEEE Access, vol. 8, pp. 42169–42184, 2020.
[9] C. Liu, Y. Liu, Y. Yan, and J. Wang, ''An intrusion detection model with hierarchical attention mechanism,'' IEEE Access, vol. 8, pp. 67542–67554, 2020.
[10] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, ''Toward a lightweight intrusion detection system for the Internet of Things,'' IEEE Access, vol. 7, pp. 42450–42471, 2019.
[11] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, ''Machine learning-based network vulnerability analysis of industrial Internet of Things,'' IEEE Internet Things J., vol. 6, no. 4, pp. 6822–6834, Aug. 2019.
[12] Y. Chen, B. Pang, G. Shao, G. Wen, and X. Chen, ''DGA-based botnet detection toward imbalanced multiclass learning,'' Tsinghua Sci. Technol., vol. 26, no. 4, pp. 387–402, Aug. 2021.
[13] X. Larriva-Novo, V. A. Villagrá, M. Vega-Barbas, D. Rivera, and M. S. Rodrigo, ''An IoT-focused intrusion detection system approach based on preprocessing characterization for cybersecurity datasets,'' Sensors, vol. 21, no. 2, p. 656, Jan. 2021.
[14] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, ''Network intrusion detection system: A systematic study of machine learning and deep learning approaches,'' Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, p. e4150, Jan. 2021.
[15] M. Aamir, S. S. H. Rizvi, M. A. Hashmani, M. Zubair, and J. A. Usman, ''Machine learning classification of port scanning and DDoS attacks: A comparative analysis,'' Mehran Univ. Res. J. Eng. Technol., vol. 40, no. 1, pp. 215–229, Jan. 2021.
[16] SHARAFALDIN, Iman, LASHKARI, Arash Habibi, HAKAK, Saqib, et al., "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy", International Carnahan Conference on Security Technology (ICCST), IEEE, p. 1-8, 2019.
[17] JAVEED, Danish, GAO, Tianhan, et KHAN, Muhammad Taimoor, "SDN-enabled hybrid DLdriven framework for the detection of emerging cyber threats in IoT", vol. 10, no 8, p. 918, 2021.
[18] ALAMRI, Hassan A. et THAYANANTHAN, Vijey, "Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks", IEEE Access, vol. 8, p. 194269-194288, 2020.
[19] ASSIS, Marcos VO, CARVALHO, Luiz F., LLORET, Jaime, et al.A GRU, "Deep learning system against attacks in software defined networks", Journal of Network and Computer Applications, vol. 177, p. 102942, 2021.
[20] Anley, M. B., Genovese, A., Agostinello, D., & Piuri, V. , "Robust DDoS attack detection with adaptive transfer learning", Computers & Security, 144, 103962. doi:10.1016/j.cose.2024.103962, 2024.
[21] Shaaban, A. R., Abd-Elwanis, E., & Hussein, M., " DDoS attack detection and classification via convolutional neural network (CNN)", In Proceedings of the IEEE Ninth International Conference on Intelligent Computing and Information Systems (ICICIS) (pp. 233–238), IEEE, 2019.
[22] Sharif, D. M., Beitollahi, H., & Fazeli, M., "Detection of application-layer DDoS attacks produced by various freely accessible toolkits using machine learning", IEEE Access, 11, 51810– 51819, 2023.
[23] Patel, M., Amritha, P. P., Sudheer, V. B., & Sethumadhavan, M., "DDoS attack detection model using machine learning algorithm in next generation firewall", Procedia Computer Science, 233, 175– 183, 2024.
[24] Baldini, G., & Amerini, I., "Online distributed denial of service (DDoS) intrusion detection based on adaptive sliding window and morphological fractal dimension", Computer Networks, 210, Article 108923, 2022.
[25] Kumar, D., Pateriya, R. K., Gupta, R. K., Dehalwar, V., & Sharma, A., "DDoS detection using deep learning", Procedia Computer Science, 218, 2420–2429, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY