Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 9 No. 6 (2025) 1347 - 1357 e-ISSN: 2580-0760

Optimizing a Hybrid Deep Learning Model for DDoS Detection Using DBSCAN and PSO

Indrastanti Ratna Widiasari[1], Rissal Efendi[2]
[1,2]Informatics Engineering Department, Information Technology Faculty, Satya Wacana Christian University, Salatiga, Indonesia

[1]indrastanti@uksw.edu, [2]rissal.efendi@uksw.edu

Abstract

This study proposes a hybrid deep learning that combines Gated Recurrent Units (GRUs) and Convolutional Neural Networks (CNNs) for Distributed Denial of Service (DDoS) cyberattack detection. The model, called DBSCAN–GRU–CNN, uses density-based clustering (DBSCAN) to select relevant features and reduce execution time. The dataset for this research was collected from penetration testing, in which several simulated attack scenarios were executed on a monitored network. To evaluate the performance of the proposed model, several comparison models were used, including DBSCAN–GRU–CNN (Single Hidden Layer), DBSCAN–GRU–CNN (Double Hidden Layers), DBSCAN–GRU–CNN (With Regularization), DBSCAN–GRU–CNN–PSO, GRU–CNN, GRU–CNN (With Hyperparameter Tuning), and Random Forest (Tuned Model). Variations of the model tested were made by adding hidden layers, regularization, optimization with Particle Swarm Optimization (PSO), and hyperparameter tuning. The results of the experiments reveal that the DBSCAN–GRU–CNN–PSO model provided optimal performance with a 99.3% accuracy, a 99% precision, a 98.9% recall, and a 99% F1-score, while the model with hyperparameter tuning achieved a 99% accuracy. By adding PSO, the model achieved optimized weights, better generalization, and excellent accuracy in DDoS detection.

Keywords: attack detection; DBSCAN; DdoS; hybrid deep learning; PSO;

How to Cite: I. R. Widiasari, and R. Efendi, "Optimizing a Hybrid Deep Learning Model for DDoS Detection Using DBSCAN and PSO", *J. RESTI (Rekayasa Sist. Teknol. Inf.)*, vol. 9, no. 6, pp. 1347 - 1357, Dec. 2025.
Permalink/DOI: https://doi.org/10.29207/resti.v9i6.6383

Received: February 8, 2025
Accepted: August 15, 2025
Available Online: December 7, 2025

1. Introduction

In recent years, Artificial Intelligence (AI), an umbrella term encompassing several Machine Learning (ML) and Deep Learning (DL) models, has acquired increased attention in the field of malware detection. ML algorithms are developed using formatted and categorized data, enabling them to identify forms and classify recent information. These algorithms have shown significant potential in detecting malicious activities by studying large amounts of cybersecurity data.

DDoS is a type of cyberattacks where attackers try to incapacitate systems and servers, rendering them inaccessible to users [1]. It causes severe damage to a system, impacting it to a severe extent [2]. All industries, irrespective of size, continuously face a serious risk to their network security due to the growing frequency, complexity, and volume of these attacks.

As networks continue to move toward a software-centric model, they face a greater risk from malicious cyber activities such as DDoS attacks, which exploit weaknesses like inadequate authentication, performance bottlenecks, and poor handling of exception [2]. Typically, these attacks overwhelm the target—a particular network or server—with numerous packets from a vast number of hosts as it becomes compromised by malicious software [3]. A DDoS attack primarily results in a massive influx of packets directed to the target, depleting all available resources, and eventually causing malfunction of every programmable switch. Consequently, a DDoS attacker can render a network inoperable. For that reason, detection of suitable responses to cyberattacks is more vital in programmable networks than in conventional ones [4]. AI models are key in solving these challenges, with deep learning models, recognized for their ability to detect complex patterns from large datasets, being frequently applied to DDoS attack detection [5], [6].

DL, which is built on Artificial Neural Networks (ANNs), is generally more intricate than ML, which includes various algorithms such as K-Nearest Neighbor (KNN), Random Forest (RF), Support Vector Machine (SVM), Logistic Regression (LR), and Naïve Bayes (NB) algorithms. The artificial neuron serves as the basic building block of an ANN. When one neuron is combined with others, the network can perform complex classification and forecasting functions as a result of their concurrent structure [7]. An ANN fundamentally consists of one input layer, one or more hidden layers, and one output layer [8]. As DL employs multiple hidden layers, it generally needs higher data volumes and computational strength compared to ML. In supervised DL, data labelling, as in distinguishing between malicious and benign data packets, is necessary. Conversely, unsupervised DL operates without data labelling. DL architectures come in several types, such as Deep Neural Networks (DNNs) , Recurrent Neural Networks (RNNs), Deep Belief Networks (DBNs) and Convolutional Neural Networks (CNNs) [9], [10]. Data Generation AI, a fast-growing domain, is applied in Large Language Models (LLMs), which are a variety of DNNs, with several layers that are essential for extracting deep abstractions and dealing with complex challenges, including natural language understanding and generation [11].

CNNs, renowned for their prowess in feature extraction from complex data structures, offer a promising avenue for enhancing the accuracy and adaptability of DDoS detection mechanisms. This paper builds upon a rich body of literature addressing DDoS attacks, cloud security, and machine learning applications in cybersecurity. Prior research has extensively explored the vulnerabilities and countermeasures associated with DDoS attacks in various contexts. Additionally, the effectiveness of machine learning, especially CNNs, in handling cybersecurity challenges has been well-documented [12] - [15].

DL models, which are growing in prevalence, are applied to detect attacks and anomalies in computer networks. Among these, GRUs prioritize newer information by enabling the model to swiftly customize the most up-to-date data and improving its capability to identify abnormal traffic [16]. The results obtained from the application of the GRU model across various meteorological forecasting tasks show advantages in precision, predictive ability, and efficiency, making it a favorable choice for this study. A GRU has a gating mechanism that performs selective updates to its hidden states according to input data. It operates with an update gate by merging the roles of the forget gate and the input gate, thereby refining the design and decreasing parameter complexity. As a result, it leads to more efficient computation and reduced model development. The GRU model incorporates hidden cells and states, which improves the smoothness of information flow, hence its ability to maintain crucial details and discard unrelated ones, making it very suitable for sequence data analysis [17] - [20]. It has been extensively implemented in a wide range of sectors, such as forecasting the movement of landslides, estimating traffic flow [21], predicting electrical load, estimating solar radiation [22], conducting advanced agricultural practices [23], assessing wind speed and temperature conditions [24], predicting carbon dioxide levels [25], and estimating solar energy levels [26].

This paper proposes two integrated models of DNNs for accurate DDoS attack monitoring and analysis. The main contribution of this study is an integrated DL approach called “GRU–CNN” is designed to identify DDoS packets and categorize them into various types of DDoS attacks promptly and accurately. On the other hand, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is applied to determine the most relevant attribute from an extensive DDoS attack dataset. By detecting the density-based clusters and outliers associated with malicious payload, this method boosts detection accuracy while reducing computation time. Therefore, potential DDoS attacks can be detected and effectively identified by clustering them based on density.

Previous studies have explored the vulnerabilities of networks targeted by DDoS attacks, as well as the potential of ML and DL in addressing these challenges. However, the primary contribution in this research lies in the integration of DBSCAN for feature selection and PSO for optimization in a hybrid GRU–CNN framework.

2. Methods

This section details the research flow for the optimization and comparison of hybrid GRU–CNN models in detecting DDoS attacks using the DBSCAN algorithm. Through this study, we propose an approach to optimizing detection accuracy by utilizing a combination of deep learning models and optimization techniques. This process also integrates clustering methods for efficient data segmentation and anomaly identification. Figure 1 illustrates the steps involved in this process.

The process started from network traffic data that went through a preprocessing stage for cleaning, feature selection, and data balancing. DBSCAN was used for data segmentation and anomaly detection. DBSCAN was chosen because it is superior to K-Means in DDoS detection. It does not assume a spherical or uniform cluster shape, making it capable of handling more complex and unstructured data patterns, such as DDoS attacks. In addition, it can automatically detect outliers and noise, which is important for separating scattered attacks from normal traffic data, while K-Means is sensitive to outliers. It also avoids the need for prior determination of the number of clusters, making it more flexible in handling imbalanced datasets with an unknown number of clusters.

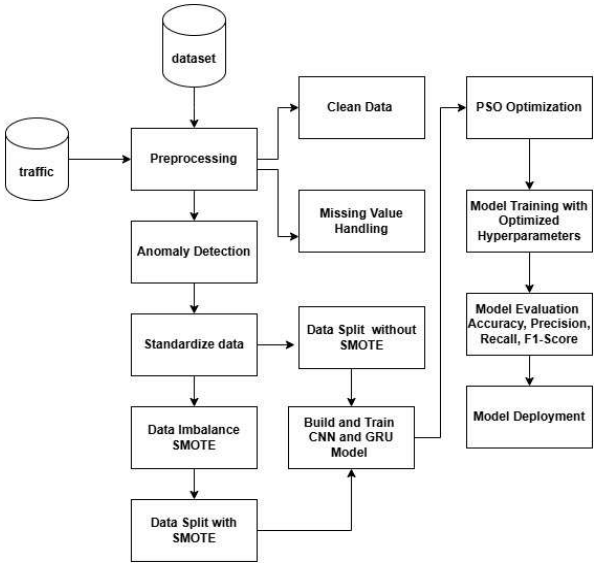

Figure 1. Flowchart of the Research

We then built a GRU model as the basis for temporal analysis. The results of the GRU were then input to build a CNN model that was optimized using the PSO algorithm to determine the best parameters, which offered an efficient way to find the optimal parameter space without the need for a grid search which could be more time-consuming and result in more dependency on the choice of pre-defined grid parameters. PSO is more flexible and effective in dealing with complex optimization problems. The final step involved evaluating the model based on accuracy, precision, recall, and F1-score metrics before the model was practically implemented.

2.1 Data Collection

Data collection included attributes such as time, sources, destinations, protocols, durations, clusters, and anomalies in a managed network infrastructure to obtain reliability and a realistic modeling of real-world attack behavior. The data included data on both benign and anomalous network traffic. The data size obtained was 1,031,892 records. From the analysis of the class distribution in the dataset, the normal class amounted to about 93%, while the DDoS class did about 7%. This shows that the dataset was not completely balanced, with a tendency for normal data to dominate. Data collection is important because it supplies the essential data needed for model training and evaluation.

In this study, the dataset was generated through controlled penetration testing in a lab environment. To simulate realistic DDoS attack scenarios, two tools were employed, namely, LOIC (Low Orbit Ion Cannon) and Hping3. LOIC was used to generate HTTP and UDP flood attacks, while Hping3 was configured to simulate SYN flood and other packet-based attacks. During the attack simulations, Wireshark was used to capture and analyze the resulting network traffic. The recorded packet data was later preprocessed and utilized to extract relevant features for training and evaluating the proposed hybrid deep learning model.

2.2 Data Preprocessing

In this phase, the raw data obtained was processed and organized to prepare it for further analysis. Data preprocessing in this study included feature normalization using Min–Max Scaling to ensure that all attributes had a uniform value range between 0 and 1 to reduce bias due to scale differences. This process included several important tasks, including addressing missing data, excluding irrelevant information, changing data formats as needed, and eliminating unnecessary or redundant data. This stage aimed to ensure that the data was consistent, accurate, and reliable, so that it could produce valid analysis results in the next stage.

In addition to basic data preprocessing, to increase data diversity and improve model performance, a data augmentation technique was performed. The technique used was SMOTE (Synthetic Minority Over-sampling Technique), which generated synthetic data to create variations of DDoS attacks that might not have been represented in the original dataset. This augmentation step aimed to balance the dataset, enrich the training data, and help the model recognize more complex and varied attack patterns.

2.3 Dataset Split

The data used in this study was split into three main subsets. The division ratio used was 80% for training data, 10% for validation data, and 10% for test data. Training data was employed to train the model, validation data was used to select optimal hyperparameters and prevent overfitting, and test data was used to test the validity of the performance that had been trained on data that had not been encountered earlier by the model. This division aimed to ensure that the model could generalize well and to avoid bias in the data used during training.

The data split with an 80:10:10 ratio was consistently applied throughout the experiments to maintain a balanced evaluation process. This approach ensured that the training phase contained sufficient data to capture underlying patterns, while the validation phase provided an independent reference for fine-tuning the model. Meanwhile, the test phase offered an unbiased benchmark for assessing the final performance. Such a division strategy was widely recognized in machine learning research as it reduced the risk of data leakage and enhanced the reliability of the experimental results.

2.4 Segmentation and Anomaly Detection Using DBSCAN

DBSCAN was used for data segmentation and anomaly detection in network data. Figure 2 represents data segmentation, which was performed to divide the data based on density patterns, allowing the identification of homogeneous clusters and striking anomalies.

The process began by selecting an unvisited data point, followed by calculating the number of points within an epsilon radius (ε). If the number of points met a

minimum threshold (minPts), the data point was marked as a core point, and the cluster was expanded by adding neighboring points. Otherwise, the data point was marked as noise or anomaly. This process continued until all data points were visited, resulting in clearly segmented clusters and anomalies. This approach is effective for understanding complex patterns in high-dimensional network data.

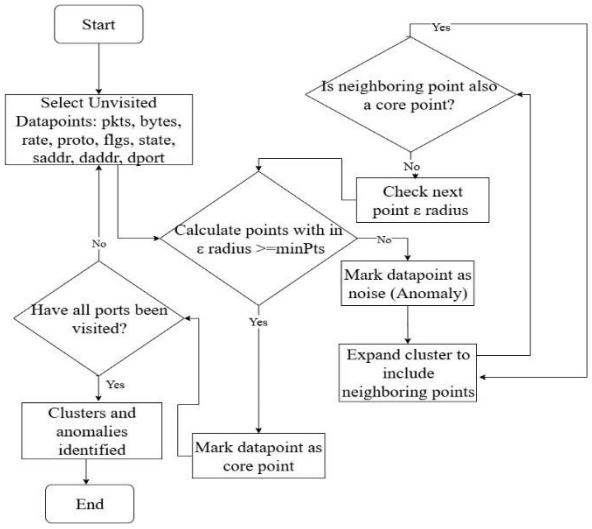

Figure 2. DBSCAN Clustering Process with Anomaly Detection Utilizing Network Traffic Variables

2.5 Building the GRU Model

In this study, the GRU model was developed with hyperparameter adjustments to ensure that the model would generate the best performance in capturing temporal dependencies in network traffic data. For the learning rate, the value used was 0.001, which is a value commonly used in optimization with Particle Swarm Optimization (PSO), because it provides stable convergence without causing too large parameter updates [27]. Meanwhile, the number of hidden units on each GRU layer was set at 128 units, which allowed the model to capture complex patterns in sequential data. The batch size used in model training was 64, which provided a good balance between memory efficiency and model accuracy. The model was trained for 50 epochs, with monitoring of convergence and avoidance of overfitting through validation techniques, using a dropout rate of 0.2 to reduce the risk of overfitting. The selection of these hyperparameters was carried out based on experiments with the model, as well as adjustments during training, to find the best configuration that could improve accuracy in detecting DDoS attacks. In the GRU architecture, the reset gate, update gate, candidate hidden state, and final hidden state.

The reset gate regulated the hidden state information which must be forgotten so that the GRU only retained information that was relevant to detecting DDoS attack patterns based on current network traffic conditions as shown in Equation 1. This mechanism ensured that irrelevant past information did not affect the model's current prediction. The mathematical formulation of the reset gate computes the gate value based on the current input and previous hidden state.

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

The update gate regulated the amount of old hidden state information that should have been retained with new information (from ongoing network traffic). This allowed the model to remember relevant patterns and ignore useless information, which is important in detecting DDoS attacks. It balanced past and current information to optimize the GRU's prediction performance. This process follows Equation 2. This process follows Equation 2.

$$Z_t = \sigma(W_z \,.\, x_t + U_z . h_{t-1} + b_z) \quad (2)$$

Candidate hidden states combined current input and reset hidden state information, calculated using Equation 3. This resulted in a new hidden state that took into account the current network traffic conditions, which is important for detecting attack patterns or anomalies in traffic data. This process allowed the GRU to generate more accurate representations by integrating relevant past and current information.

$$\check{h}_t = tanh\,(W_h \,.\, x_t + U_h \,.\, (r_t \odot h_{t-1}) + b_h \quad (3)$$

The final hidden state combined the old hidden state $\check{h}_t$ and the new candidate hidden state. The GRU model could decide whether to retain the old hidden state or combine it with new information based on the current network traffic input. This mechanism enabled the model to adaptively update its memory, improving its ability to detect DDoS attack patterns. Final hidden gate is represented in Equation 4.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \check{h}_t \quad (4)$$

To explain how the GRU model was used in DDoS detection, the process began with input in the form of network traffic data (such as the number of packets, connection duration, and traffic levels) and output in the form of attack detection (attack or normal traffic). The input data was converted into a time sequence, where the GRU processed the information by retaining relevant data from the previous sequence using the reset gate and update gate. The GRU then integrated the new input and the previous hidden state to produce the final hidden state that would be processed by the dense layer to predict the DDoS detection state. Thus, the GRU could capture temporal patterns in traffic data and enhance the effectiveness of DDoS attack detection.

2.6 Building the CNN Model using GRU Outputs

After the results from the GRU were processed, the CNN model was used for DDoS attack classification. For the CNN model, several hyperparameters were also set to get the best performance in detecting attack patterns. On the convolutional layer, the numbers of filters used were 32, 64, and 128 on the first, second, and third convolutional layers, respectively. The filter (kernel) size used on each Conv1D layer was 3 (equivalent to 3 x 1 in 1D convolution), as it is adept at

identifying temporal or local patterns in sequential data. To avoid overfitting, the dropout rate on the fully connected layer was set to 0.5. In addition, the batch size for the CNN model was set to 32 to speed up the training process, while the activation function used on the convolutional layer was ReLU, which helped improve non-linearity in the model. On the output layer, the The model utilized a sigmoid activation function for binary classification between DDoS or non-DDoS.

Table 1 shows the CNN architecture for DDoS detection. The model consisted of three convolutional layers (Conv1D) with increasing numbers of filters, followed by a fully connected layer. The output layer used a Sigmoid activation function since this was a binary classification problem (DDoS or non-DDoS). Table 2 shows regularization techniques in the CNN architecture for DDoS detection. MaxPooling1D was applied after each convolutional layer to reduce the output dimensionality and help the model to enhance generalization capability. A dropout rate of 0.5 was applied to the dense layers to help minimize overfitting. Batch normalization was applied after each convolutional layer to speed up the training and optimize the model's performance.

Table 1. CNN Architecture for DDoS Detection

Layer	Type	Output Shape	Filters/Units	Kernel Size	Activation
Input	-	(None, timesteps, 1)	-	-	-
Convolutional Layer 1	Conv1D	(None, timesteps-2, 32)	32	3	ReLU
Convolutional Layer 2	Conv1D	(None, timesteps-4, 64)	64	3	ReLU
Convolutional Layer 3	Conv1D	(None, timesteps-6, 128)	128	3	ReLU
Fully Connected	Dense	(None, units)	-	-	ReLU
Output	Dense	(None, 1)	-	-	Sigmoid

Table 2. CNN Architecture for DDoS Detection

Layer	Pooling	Dropout	Batch Normalization
Input	-	-	-
Convolutional Layer 1	MaxPooling1D	-	Yes
Convolutional Layer 2	MaxPooling1D	-	Yes
Convolutional Layer 3	MaxPooling1D	-	Yes
Fully Connected	-	0.5	No
Output	-	-	No

Mathematically, the convolution process can be described by the Equation 5:

$$F_t = \sum_{i=0}^{L-1} X_{t+1} \ . K_i \quad (5)$$

F_t is the the result of a convolution operation that shows the effectiveness of the kernel or filter K, which fits a particular portion of the input X at time (or position) t; X_{t+1} is the input (output of the GRU) at position $t+1$, which contains network packet traffic flow in DDoS detection, or time values in time-series data that has been analyzed by the GRU; $\sum_{i=0}^{L-1} X_{t+1}$ is the sum of all the products of the input values and the kernel values for each position in the filter; and *K*: is a CNN filter or kernel, which is used to detect patterns or anomalies in network traffic.

This convolution operation was used to produce features highlighting patterns that could help identify whether the network traffic was part of a DDoS attack or normal traffic.

2.7 PSO Optimization

In this phase, PSO was used to optimize the model built using GRU–CNN. Each particle in the PSO model represented a potential solution consisting of GRU–CNN parameters, such as weights and biases. This process began by initializing the particles randomly in the specified search space. This random initialization allowed PSO to explore various possible GRU–CNN parameter configurations in an attempt to find optimal convergence to the expected solution. There were some steps to perform optimization using PSO. First, the velocity of each particle is updated by considering its previous velocity, its best-known position, and the global best position found by the swarm as represented in Equation 6. Then, the position of each particle was updated according to the new velocity, enabling the swarm to gradually move toward the optimal solution in the search space.

$$vi(t+1) = w.v_i(t) + c_1.r_i\left(p_{i,best} - x_i(t)\right) + c_2.r_2 (g_{best} - x_i(t)) \quad (6)$$

$vi(t)$ is the velocity of particle *i* at time *t*; w is the inertia weight; c_1 and c_2 are the cognitive and social coefficients, respectively; r_i and r_2 are the random numbers between 0 and 1, respectively; $p_{i,best}$ is the best location/position identified by particle *i*; g_{best} is the best position found by the entire swarm; and $x_i(t)$ is the current position of particle *i*.

For example, suppose a particle's velocity at time *t* for parameter TCP packets is 0.5, w = 0.7, c_1 = 1.5, c_2 = 1.5, r_i and r_2 are random values (e.g., 0.4 and 0.6), $p_{i,best}$ = 0.8, and g_{best} = 1.0. Then, the updated velocity might be:

$$v_{tcp_{packet}}(t+1) = 0.7 \cdot 0.5 + 1.5 \cdot 0.4 \cdot (0.8 - 6) + 1.5 \cdot (1.0 - 6)$$

$$v_{tcp_{packet}}(t+1) \approx 0.35 + (-2.7) + (-4.5) = -6.85 .$$

Position $x_i(t+1)$ is updated using new velocity $x_i(t+1) = x_i(t) + v_i(t+1)$. For example, if the current position of the particle for the TCP packets is 6, then: $x_{tcp_{packets}}(t+1) = 6 + (-6.85) = -0.85$. Each new position of the particle was evaluated using an objective

function (e.g., malware detection accuracy). The best position (solution) was updated accordingly. Then, the steps were repeated for a set number of iterations.

These features would form part of the input vector x_t for GRU–CNN. PSO optimized GRU–CNN weights and biases to increase accuracy in detecting types (benign or malicious).

2.8 Model Evaluation

In this stage, we applied a confusion matrix to determine the necessary requirement for the model. Some of the used features were false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN). The confusion matrix was provided to focus on the correctness of the predictions. We also evaluated our proposed model with metrics commonly applied in DDoS.

Accuracy describes the precision of prediction of the model. It was used to calculate abnormal or malicious packets generated by the CNN–GRU model. Equation 7 represents the accuracy percentage of any DDoS packets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

Precision is the percentage of misidentified intrusion to the total incidents of attacks that occurred. It is based on Equation 8, which shows the number of correctly predicted positive DDoS detections:

$$Precision\ (P) = \frac{TP}{FP + TP} \quad (8)$$

Recall is the outcome ratio of detecting adverse incidence compared to the overall number of adverse vectors. Equation 9 demonstrates the number of true positives which are precisely identified:

$$Recall\ (r) = \frac{TP}{FN + TP} \quad (9)$$

F-score or F1-score provides the following information about network functionality which reflects on both false positives and false negatives. It is useful especially in situations where class labels are imbalanced. Equation 10 demonstrates the consistency of recall and precision:

$$F_{score} = 2 \times \frac{P \times R}{P + R} \quad (10)$$

3. Results and Discussions

In this research, we aimed to optimize the effectiveness of a hybrid deep learning model that combines DBSCAN, GRU, and CNN techniques in detecting DDoS attacks. This approach was chosen to address the increasingly complex challenges of attack detection by utilizing DBSCAN's ability to handle imbalanced data and GRUs' ability to recognize temporal patterns in large and dynamic data. This hybrid model was then optimized with various techniques, including regulation and hyperparameter settings, to improve detection accuracy.

3.1 Results

During model development, various configuration variations, such as varying numbers of hidden layers and neurons per layer and dropout rates, were tested to find the best combination that could produce optimal performance. This process aimed to explore the potential of each technique used, as well as to understand how each element of the model could result in improved accuracy and efficiency for DDoS detection. The results of the model development are expected to provide further insights into the strengths and weaknesses of each method in the context of network attack detection.

Figure 3 shows performance comparison metrics of the all models used in the study. The metrics compared included accuracy, precision, recall, and F1-score, expressed as percentages. The models tested included methods such as LSTM, DBSCAN, SMOTE, PSO, and Random Forest, with various configurations and optimizations. The findings indicate that the DBSCAN–LSTM model with PSO showed the highest performance compared to other models, especially in terms of accuracy and F1-score.

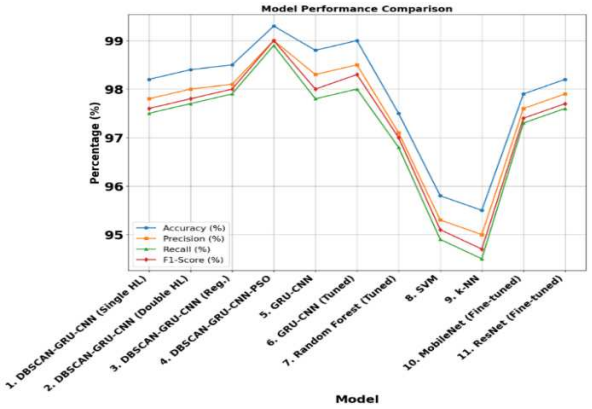

Figure 3. Performance Metrics of Various Models

Table 3 presents the outcomes of the performance evaluation of various models in terms of accuracy, precision, recall, and F1-score. From this table, it can be seen that the DBSCAN–GRU–CNN–PSO model produced the best performance compared to all other models, with an accuracy of 99.3%, a precision of 99%, a recall of 98.9%, and an F1-score of 99%. This shows a significant performance improvement compared to baseline models, such as SVM and k-NNs, which only achieved accuracies of 95.8% and 95.5% and F1-scores of 95.1% and 94.7%, respectively. In addition, popular pre-trained deep learning models such as MobileNet and ResNet also showed good performance but were still below the proposed model. MobileNet produced an accuracy of 97.9% and an F1-score of 97.4%, while ResNet achieved an accuracy of 98.2% and an F1-score of 97.7%. Compared to the baseline CNN model (GRU–CNN), the use of techniques such as regularization, hyperparameter tuning, and optimization using PSO was shown to consistently improve performance. This strengthens the argument

that the CNN model proposed in this research is not only effective, but also outperforms simpler methods and other deep learning alternatives.

Table 3. Comparison of metrics

No	Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	DBSCAN–GRU–CNN (Single Hidden Layer)	98.2	97.8	97.5	97.6
2	DBSCAN–GRU–CNN (Double Hidden Layers)	98.4	98	97.7	97.8
3	DBSCAN–GRU–CNN (With Regularization)	98.5	98.1	97.9	98
4	DBSCAN–GRU–CNN–PSO	99.3	99	98.9	99
5	GRU–CNN	98.8	98.3	97.8	98
6	GRU–CNN (With Hyperparameter Tuning)	99	98.5	98	98.3
7	Random Forest (Tuned Model)	97.5	97.1	96.8	97
8	Support Vector Machine (SVM)	95.8	95.3	94.9	95.1
9	k-Nearest Neighbors (k-NNs)	95.5	95	94.5	94.7
10	Pre-trained MobileNet (Fine-tuned)	97.9	97.6	97.3	97.4
11	Pre-trained ResNet (Fine-tuned)	98.2	97.9	97.6	97.7

Table 4 and Figure 4 show the training and validation evaluation results of various models based on training loss, validation loss, training accuracy, and validation accuracy. The table provides a deeper overview of the stability and generalization ability of each model tested.

The DBSCAN–GRU–CNN–PSO model again showed the best performance, with the lowest training loss (0.05) and a very small validation loss (0.08). In addition, this model recorded a training accuracy of 99.3% and a validation accuracy of 98.9%, indicating that this model is not only very accurate on the training and validation data. The small difference between training and validation accuracies indicates a very good generalization ability and minimal overfitting.

The GRU–CNN model and the GRU–CNN model with hyperparameter tuning also showed strong performance, with validation accuracies of 97.8% and 98.3% and relatively low validation losses (0.14 and 0.10), respectively. Meanwhile, baseline models, such as SVM and k-NNs, had much higher validation losses (0.32 and 0.34) and lower validation accuracies (95.5% and 95.2%). This indicates that these traditional models are less able to handle data complexity optimally compared to the CNN-based deep learning architecture used in this study.

Popular pre-trained deep learning models, such as MobileNet and ResNet, showed competitive results with validation accuracies of 97.4% and 97.7% and validation losses of 0.17 and 0.15, respectively. Although their performance was quite good, these results were still below the proposed models, especially DBSCAN–GRU–CNN–PSO, both in terms of accuracy and training efficiency.

Overall, Table 4 supports the findings from Table 3 that the proposed CNN model does not only excel in accuracy, but is also stable, efficient, and has an excellent generalization ability compared to other benchmark models.

Table 4. Training and Validation Outcomes

No	Model	Training Loss	Validation Loss	Training Accuracy (%)	Validation Accuracy (%)
1	DBSCAN–GRU–CNN (Single Hidden Layer)	0.15	0.2	98.2	97.5
2	DBSCAN–GRU–CNN (Double Hidden Layers)	0.12	0.18	98.4	97.7
3	DBSCAN–GRU–CNN (With Regularization)	0.1	0.15	98.5	97.9
4	DBSCAN–GRU–CNN–PSO	0.05	0.08	99.3	98.9
5	GRU–CNN	0.1	0.14	98.8	97.8
6	GRU–CNN (With Hyperparameter Tuning)	0.08	0.1	99	98.3
7	Random Forest (Tuned Model)	0.25	0.3	97.5	96.8
8	Support Vector Machine (SVM)	0.28	0.32	96.7	95.5
9	k-Nearest Neighbors (k-NNs)	0.3	0.34	96.4	95.2
10	Pre-trained MobileNet (Fine-tuned)	0.14	0.17	98.1	97.4
11	Pre-trained ResNet (Fine-tuned)	0.11	0.15	98.5	97.7

The results in Table 3 and Table 4 as a whole show that the developed CNN-based approach not only provides accurate prediction results, but also stability during the training and validation process. The combination of evaluation metrics (accuracy, precision, recall, and F1-score) with training metrics (loss and validation accuracy) allowed for a more comprehensive analysis of model performance. In particular, CNN-based models combined with optimization and regularization methods showed high performance consistency when tested on both training and validation data. This performance shows that the model is not only superior in terms of evaluation numbers, but also robust in dealing with data variations, making it worthy of consideration as a more effective approach compared to traditional methods or other common deep learning

architectures.

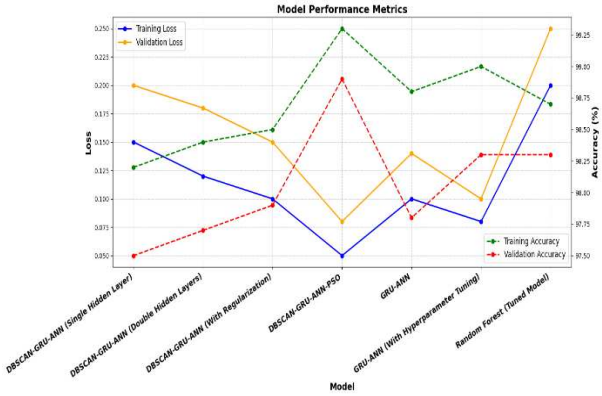

Figure 4. Model Performance Metrics

3.2 Discussion

The results of this experiment show that DBSCAN–GRU–CNN significantly improves accuracy in detecting DDoS attacks compared to other simpler models. The DBSCAN–GRU–CNN–PSO model, which utilizes Particle Swarm Optimization (PSO) for hyperparameter adjustment, successfully achieved the highest accuracy of 99.3%, with a precision of 99% and a recall of 98.9%. These results indicate that hyperparameter optimization through methods such as PSO greatly contributes to the achievement of the best performance, where PSO is able to optimize parameter values configuration to improve model performance. In comparison to other models, DBSCAN–GRU–CNN (With Regularization) showed very good performance, with an accuracy of 98.5%. The addition of regularization to the model helped the model address overfitting, which is a common difficulty in optimizing models, especially on large and imbalanced datasets, such as those often encountered in DDoS detection. The use of DBSCAN to handle imbalanced data proved to be very effective, which is consistent with findings from previous studies that showed that DBSCAN can help in identifying hidden attack patterns in irregular data [28], [29]. DBSCAN also helps in reducing the impact of noise that is often present in DDoS datasets, increasing the model's effectiveness to generalize and detect unusual attacks.

However, although the simpler GRU–CNN model without the DBSCAN component achieved an accuracy of 98.8%, the result was slightly lower compared to that of the hybrid DBSCAN–GRU–CNN model. This suggests that GRUs are indeed effective in handling sequence-based or temporal data, such as that often encountered in DDoS attacks, where attack patterns can evolve over time. However, without the help of techniques such as DBSCAN to handle the problem of data imbalance, GRU–CNN's ability to identify more complex or less frequent attacks is somewhat limited.

The use of Random Forest in this study also showed quite competitive results, with an accuracy of 98.7%, indicating that ensemble models such as Random Forest remain a solid choice in many classification tasks. However, compared to the hybrid model based on DBSCAN–GRU–CNN, Random Forest is less effective in handling DDoS attacks with more dynamic and varied patterns. This is in line with the study conducted by [30], which showed that although Random Forest can provide good results in some cases, deep-learning-based models tend to be superior in tasks involving more complex and large data.

Overall, the results of this study show that the combination of DBSCAN and GRUs in an CNN architecture is very effective for DDoS attack detection, especially when the model is optimized with techniques such as PSO. The main advantage of this hybrid approach is its ability to handle imbalanced data and identify temporal patterns that are sometimes missed by other models. Although models such as GRU–CNN and Random Forest also perform well, the combination of techniques such as DBSCAN provides advantages in detecting more complex and rare attacks. Therefore, this study contributes to the development of a more robust and accurate DDoS detection model, which can be implemented in network security systems to identify attacks more effectively.

In this discussion, we will compare the results of several recent studies with a focus on intrusion detection using various methods [31] - [40]. Table 5 reveals the various detection techniques applied in these studies, ranging from hybrid methods and signatures, to machine learning and deep learning techniques. Each study showed varying results in terms of accuracy, precision, recall, and the advantages and limitations of each approach.

The table presents a comparison of various intrusion detection systems (IDS) and their effectiveness in detecting different types of attacks, employing different methodologies. A clear trend toward hybrid approaches emerged, with models like those proposed by Raza et al. [31], Midi et al. [34], and Xu et al. [36] demonstrating exceptional results for detecting DoS and routing attacks. Hybrid models appear to be effective in balancing the strengths of different techniques, achieving high accuracy and low false alarm rates. Moreover, deep-learning-based methods, such as those used by Vanitha et al. [37], show a growing interest in leveraging advanced machine learning techniques like BiLSTM, ELM, GRU, and transfer learning to improve IDS performance across diverse datasets. These approaches contribute to more robust systems capable of adapting to a wide range of attack scenarios, highlighting the growing role of deep learning and ensemble learning in cybersecurity.

The results presented in the table highlight significant advancements in IDS performance, with models such as that proposed by Xu et al. [36] achieving perfect scores in accuracy, precision, recall, and F1-score.

Table 5. Performance Comparison of IDS Methods

Reference	Publish Year	Detection Method	Encountered Attacks	Experiment-al Results
Raza et al. [31]	2013	Hybrid	Sinkhole, spoofing, altered info	100% TPR (lossless), 90% TPR (lossy)
Kasinathan et al. [32]	2013	Signature	DoS	Reduced FPR
Cervantes et al. [33]	2015	Hybrid	Sinkhole	92% DR, 8% FNR (static); 70% DR, 28% FNR (dynamic)
Midi et al. [34]	2017	Hybrid	DoS and routing attacks	91% DR, 100% accuracy, 0.19% CPU usage
Sharma et al. [35]	2019	Specification-based	Zero-day attacks	FNR = 0.137, FPR = 0.045, TPR = 0.863
Xu et al. [36]	2023	Hybrid	DoS	Achieved perfect scores: accuracy, precision, recall, and F1 = 1.0.
Vanitha et al. [37]	2023	BiLSTM, ELM, GRU	Improved ant colony optimization and machine-learning-based ensemble Intrusion Detection model	The new MLEID classifier's overall findings are 98.34%, with precision rates for classifiers like DT, SVM, and Ensemble at 77.67%, 89.67%, and 94.34%, respectively.
Rehman et al. [38]	2024	Signature-based	Anomaly detection for enhanced cybersecurity	90.37% accuracy, 91.4% precision, 90.2% recall, and 90.1% F1-score,
Hussain et al. [39]	2024	Rule-based	DDoS attack in Cyber-Physical Production Systems (CPPS)	90.37% accuracy, 98.5% precision, 98.3% recall, and 98.4% F1-score,
Ahuja et al. [40]	2021	SVC, Random Forest	Hybrid model combining SVC and Random Forest	Accuracy: 98.8%, and a very low false alarm rate
Our Model	2025	DBSCAN–GRU–CNN–PSO	An intrusion detection system for hybrid DoS attacks	Accuracy: 99.3%, Precision: 99%, Recall: 98.9, F1-Score: 99%

The focus on reducing false positive rates (FPR), as seen in the work of Kasinathan et al. [31], further emphasizes the importance of minimizing unnecessary alarms in practical deployment. Furthermore, the latest models, like the DBSCAN–GRU–CNN–PSO model proposed in 2025, achieved impressive results (99.3% accuracy, 99% precision, and 98.9% recall), showcasing the power of combining clustering algorithms with neural networks and optimization techniques. These findings demonstrate that ongoing innovation in hybrid models and machine learning techniques is crucial for addressing increasingly sophisticated and dynamic cybersecurity threats, providing more efficient and adaptive solutions for intrusion detection. The proposed DBSCAN–GRU–CNN–PSO model showed better performance in DDoS attack detection, with a 99.3% accuracy, a 99% precision, and a 98.9% recall. Compared with other state-of-the-art models, such as the hybrid models of Xu et al. (2023) [36] and Vanitha et al. (2023) [37], DBSCAN–GRU–CNN–PSO showed superiority in terms of precision and recall, and was more effective in handling imbalanced data problems using DBSCAN. Although Xu et al. (2023) achieved a perfect score, the DBSCAN–GRU–CNN–PSO model outperformed the others in detecting more complex and rare attack patterns due to the combination of clustering and optimization techniques. This makes the model more adaptable to be applied in real-world scenarios, where attack patterns may continue to evolve over time.

Although the DBSCAN–GRU–CNN–PSO hybrid model is effective in detecting DDoS attacks, there are some limitations related to the dataset used. The dataset, although large, is limited to a specific type of attacks and may not cover the latest variations or attack techniques found in the real world. This may affect the model's ability to generalize to more complex and dynamic attack scenarios.

For real-world deployment, several practical considerations must be addressed, such as inference time, computational cost, and resource usage in live network environments. Inference time directly impacts the system's responsiveness, while efficient resource utilization is crucial to ensuring that the approach can scale and operate effectively under real-time constraints. This becomes particularly critical in edge computing environments, where available memory and processing power are limited. Therefore, deployment strategies should include lightweight model optimization techniques such as pruning, quantization, and distributed inference to support scalability and responsiveness in operational settings such as IoT networks or ISP-level monitoring systems.

In addition, the historical data used may not reflect the rapid evolution of DDoS attacks, so the model needs to be tested with more recent and diverse datasets. Although DBSCAN is effective in dealing with imbalanced data, it may face challenges in highly dynamic and noisy environments, potentially decreasing detection accuracy.

In the future, it is important to use more diverse real-time datasets and develop hybrid approaches that incorporate anomaly detection, real-time learning, and

deployment-aware optimizations to improve performance in evolving, high-volume environments.

4. Conclusions

The DBSCAN–GRU–CNN–PSO model proposed in this study showed high efficiency in DDoS attack detection, achieving a 99.3% accuracy, a 99% precision, a 98.9% recall, and a 99% F1-score. The hybrid approach combining DBSCAN for clustering, GRUs and CNNs for feature extraction, and PSO for optimization sas shown efficacy in improving the performance of DDoS attack detection, even with an imbalanced dataset (93% normal class, 7% DDoS class). The data used in this research was collected through real-time penetration testing, in which simulated attacks were carried out on a managed network. This data provided a realistic and valid picture of real-world attacks, with attributes such as time, sources, destinations, protocols, durations, clusters, and anomalies used for model training and evaluation, providing a strong basis for intrusion detection models.

Although the model showed excellent results based on the data collected through live penetration testing, greater challenges arise when the model is implemented in a real operational network. One challenge is The model's adaptability to the highly dynamic and diverse attacks that often occur in real-world environments, which may not always be fully represented in simulated data. Furthermore, although the model showed high performance in terms of accuracy and detection, implementation in larger and more complex systems requires considerations regarding scalability and efficiency of computing resource usage, especially in real-time detection. Future research should focus on testing the model in more dynamic and complex operational environments and consider further optimizations so that the model can be effectively integrated into the network infrastructure of enterprises or Internet service providers (ISPs) to deal with the ever-evolving DDoS threats.

Acknowledgements

This research received funding from the Vice Rector for Research, Innovation, and Entrepreneurship (Contract No. 102/SPK-PF/RIK/07/2025) and supported by the Infrastructure and Digitalization Directorate (DID) of Satya Wacana Christian University.

References

[1] R. Efendi, T. Wahyono, and I. R. Widiasari, "DBSCAN SMOTE LSTM: Effective Strategies for Distributed Denial of Service Detection in Imbalanced Network Environments," *Big Data and Cognitive Computing*, vol. 8, no. 9, p. 118, Sep. 2024, doi: 10.3390/bdcc8090118.

[2] F. Salatino, M. G. Spina, M. Tropea, and F. De Rango, "Detecting DDoS Attacks Through AI driven SDN Intrusion Detection System," in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, IEEE, Jan. 2024, pp. 990–993. doi: 10.1109/CCNC51664.2024.10454798.

[3] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, Apr. 2004, doi: 10.1145/997150.997156.

[4] S. Wang, K. Gomez, K. Sithamparanathan, M. R. Asghar, G. Russello, and P. Zanna, "Mitigating DDoS Attacks in SDN-Based IoT Networks Leveraging Secure Control and Data Plane Algorithm," *Applied Sciences*, vol. 11, no. 3, p. 929, Jan. 2021, doi: 10.3390/app11030929.

[5] Y. Cui *et al.*, "Towards DDoS detection mechanisms in Software-Defined Networking," *Journal of Network and Computer Applications*, vol. 190, p. 103156, Sep. 2021, doi: 10.1016/j.jnca.2021.103156.

[6] T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review," *Applied Sciences*, vol. 13, no. 5, p. 3183, Mar. 2023, doi: 10.3390/app13053183.

[7] J. Zupan, "Introduction to Artificial Neural Network (ANN) Methods: What They Are and How to Use Them," *Acta Chim Slov*, vol. 41, Jan. 1994.

[8] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware Detection with Artificial Intelligence: A Systematic Literature Review," *ACM Comput Surv*, vol. 56, no. 6, pp. 1–33, Jun. 2024, doi: 10.1145/3638552.

[9] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[10] O. Aslan and A. A. Yilmaz, "A New Malware Classification Framework Based on Deep Learning Algorithms," *IEEE Access*, vol. 9, pp. 87936–87951, 2021, doi: 10.1109/ACCESS.2021.3089586.

[11] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, "From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy," *IEEE Access*, vol. 11, pp. 80218–80245, 2023, doi: 10.1109/ACCESS.2023.3300381.

[12] R. Verma, M. Jailia, M. Kumar, and B. Kaliraman, "CNN-Based Detection of DDoS Attacks in Multi-Cloud Environments," in *2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE)*, IEEE, May 2024, pp. 1011–1016. doi: 10.1109/IC3SE62002.2024.10593351.

13] E. Aydin and S. Bahtiyar, "OCIDS: An Online CNN-Based Network Intrusion Detection System for DDoS Attacks with IoT Botnets," in *2021 14th International Conference on Security of Information and Networks (SIN)*, IEEE, Dec. 2021, pp. 1–8. doi: 10.1109/SIN54109.2021.9699288.

[14] T. Dhirta and A. Sharma, "Unveiling the Effectiveness of CNN-based Models for Multiclass DDoS Attack Detection and Classification: A Comparative Analysis," in *2023 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*, IEEE, Dec. 2023, pp. 1–8. doi: 10.1109/ICDSAAI59313.2023.10452429.

[15] P. C *et al.*, "1D CNN Based Model for Detection of DDoS Attack," in *2024 2nd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, IEEE, Mar. 2024, pp. 1–6. doi: 10.1109/DICCT61038.2024.10533005.

[16] I. Ahmad, M. Imran, A. Qayyum, M. S. Ramzan, and M. O. Alassafi, "An Optimized Hybrid Deep Intrusion Detection Model (HD-IDM) for Enhancing Network Security," *Mathematics*, vol. 11, no. 21, p. 4501, Oct. 2023, doi: 10.3390/math11214501.

[17] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep Learning for Time Series Forecasting: Advances and Open Problems," *Information*, vol. 14, no. 11, p. 598, Nov. 2023, doi: 10.3390/info14110598.

[18] N. Li, H. Sheng, P. Wang, Y. Jia, Z. Yang, and Z. Jin, "Modeling Categorized Truck Arrivals at Ports: Big Data for Traffic Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 3, pp. 2772–2788, Mar. 2023, doi: 10.1109/TITS.2022.3219882.

[19] A. Shewalkar, D. Nyavanandi, and S. A. Ludwig, "Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU," *Journal of*

Artificial Intelligence and Soft Computing Research, vol. 9, no. 4, pp. 235–245, Oct. 2019, doi: 10.2478/jaiscr-2019-0006.

[20] A. Casolaro, V. Capone, G. Iannuzzo, and F. Camastra, "Deep Learning for Time Series Forecasting: Advances and Open Problems," *Information*, vol. 14, no. 11, p. 598, Nov. 2023, doi: 10.3390/info14110598.

[21] W. Shu, K. Cai, and N. N. Xiong, "A Short-Term Traffic Flow Prediction Model Based on an Improved Gate Recurrent Unit Neural Network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16654–16665, Sep. 2022, doi: 10.1109/TITS.2021.3094659.

[22] J. Wojtkiewicz, M. Hosseini, R. Gottumukkala, and T. L. Chambers, "Hour-Ahead Solar Irradiance Forecasting Using Multivariate Gated Recurrent Units," *Energies (Basel)*, vol. 12, no. 21, p. 4055, Oct. 2019, doi: 10.3390/en12214055.

[23] X.-B. Jin, X.-H. Yu, X.-Y. Wang, Y.-T. Bai, T.-L. Su, and J.-L. Kong, "Deep Learning Predictor for Sustainable Precision Agriculture Based on Internet of Things System," *Sustainability*, vol. 12, no. 4, p. 1433, Feb. 2020, doi: 10.3390/su12041433.

[24] F. R. Alharbi and D. Csala, "Short-Term Wind Speed and Temperature Forecasting Model Based on Gated Recurrent Unit Neural Networks," in *2021 3rd Global Power, Energy and Communication Conference (GPECOM)*, IEEE, Oct. 2021, pp. 142–147. doi: 10.1109/GPECOM52585.2021.9587479.

[25] J. Zang *et al.*, "Prediction Model of Carbon Dioxide Concentration in Pig House Based on Deep Learning," *Atmosphere (Basel)*, vol. 13, no. 7, p. 1130, Jul. 2022, doi: 10.3390/atmos13071130.

[26] A. Yildirim, M. Bilgili, and A. Ozbek, "One-hour-ahead solar radiation forecasting by MLP, LSTM, and ANFIS approaches," *Meteorology and Atmospheric Physics*, vol. 135, no. 1, p. 10, Feb. 2023, doi: 10.1007/s00703-022-00946-x.

[27] N. H. Saadoon Al-Sarray, A. Demirhan, and J. Rahebi, "Detection of distributed denial-of-service (DDoS) attacks with crow search optimization algorithm and artificial neural network," *IET Conference Proceedings*, vol. 2024, no. 23, pp. 193–198, Jan. 2025, doi: 10.1049/icp.2024.4423.

[28] S. O. Al-mamory and Z. M. Algelal, "A modified DBSCAN clustering algorithm for proactive detection of DDoS attacks," in *2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)*, IEEE, Mar. 2017, pp. 304–309. doi: 10.1109/NTICT.2017.7976107.

[29] A. Girma, M. Garuba, and R. Goel, "Advanced Machine Language Approach to Detect DDoS Attack Using DBSCAN Clustering Technology with Entropy," 2018, pp. 125–131. doi: 10.1007/978-3-319-54978-1_17.

[30] M. Alduailij, Q. W. Khan, M. Tahir, M. Sardaraz, M. Alduailij, and F. Malik, "Machine-Learning-Based DDoS Attack Detection Using Mutual Information and Random Forest Feature Importance Method," *Symmetry (Basel)*, vol. 14, no. 6, p. 1095, May 2022, doi: 10.3390/sym14061095.

[31] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013, doi: 10.1016/j.adhoc.2013.04.014.

[32] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-Service detection in 6LoWPAN based Internet of Things," in *2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, Oct. 2013, pp. 600–607. doi: 10.1109/WiMOB.2013.6673419.

[33] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, May 2015, pp. 606–611. doi: 10.1109/INM.2015.7140344.

[34] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalis — A System for Knowledge-Driven Adaptable Intrusion Detection for the Internet of Things," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, IEEE, Jun. 2017, pp. 656–666. doi: 10.1109/ICDCS.2017.104.

[35] V. Sharma, I. You, K. Yim, I.-R. Chen, and J.-H. Cho, "BRIoT: Behavior Rule Specification-Based Misbehavior Detection for IoT-Embedded Cyber-Physical Systems," *IEEE Access*, vol. 7, pp. 118556–118580, 2019, doi: 10.1109/ACCESS.2019.2917135.

[36] B. Xu, L. Sun, X. Mao, R. Ding, and C. Liu, "IoT Intrusion Detection System Based on Machine Learning," *Electronics (Basel)*, vol. 12, no. 20, p. 4289, Oct. 2023, doi: 10.3390/electronics12204289.

[37] S. Vanitha and P. Balasubramanie, "Improved Ant Colony Optimization and Machine Learning Based Ensemble Intrusion Detection Model," *Intelligent Automation & Soft Computing*, vol. 36, no. 1, pp. 849–864, 2023, doi: 10.32604/iasc.2023.032324.

[38] F. Rehman, F. Mushtaq, and H. Zaman, "A Host-based Intrusion Detection: Using Signature-based and AI-driven Anomaly Detection for Enhanced Cybersecurity*," in *2024 4th International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, IEEE, Oct. 2024, pp. 1–7. doi: 10.1109/ICoDT262145.2024.10740248.

[39] A. Hussain, E. Marín Tordera, X. Masip-Bruin, and H. C. Leligou, "Rule-Based With Machine Learning IDS for DDoS Attack Detection in Cyber-Physical Production Systems (CPPS)," *IEEE Access*, vol. 12, pp. 114894–114911, 2024, doi: 10.1109/ACCESS.2024.3445261.

[40] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, p. 103108, Aug. 2021, doi: 10.1016/j.jnca.2021.103108.