Efficient Cloud IDS Framework Using Automated Feature Encoding and Multi-Model Evaluation

Himanshu Shukla[1], Dr. Ajay Partap[2], Dr. Harsh Dev[3]

[1,2]AIIT, Amity University Uttar Pradesh, Lucknow – 226012

[3] Babu Banasri Das University, Uttar Pradesh, Lucknow, India

Corresponding author: Himanshu Shukla. (e-mail: himanshushukla@csjmu.ac.in),

Author(s) Email: Dr. Ajay Partap (e-mail: apratap@lko.amity.edu, Dr. Harsh Dev (e-mail: drharshdev@gmail.com)

HOW TO CITE:

Himanshu Shukla, Ajay Partap, Harsh Dev (2026). Efficient Cloud IDS Framework Using Automated Feature Encoding and Multi-Model Evaluation. International Journal of Special Education, 41(4s), 211-223.

COPYRIGHT STATEMENT:

Copyright: © 2026 Authors. Open access publication underthe terms and conditions of the Creative CommonsAttribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).

ABSTRACT:

This work provides an explanation for an efficient method for the detection of Distributed Denial-of-Service (DDoS) traffic on cloud networks, achieved through the development of an elucidated machine learning process. This process begins with an exploration and methodically thorough evaluation of the selected dataset, followed by extensive cleaning, encoding, scaling, and partitioning. Three respective classifications achieved using the Random Forest, Support Vector Machine (SVM), and the development of Logistic Regression algorithms, are performed on the partitioned data. The preliminary testing provides an accurate reading of 99.83%, along with the presence of high precision, recall, and F1-score values, substantiating the efficient potential of classical algorithms coupled with expert-level attention to detail, as well as complete and effective data preparation, for optimal functionality on cloud networks. In comparing the model to that of other contemporary research concerning efficient cloud network intrusion detection, it becomes immediately clear that the integration of the proposed efforts maintains accordance and, in some instances, beats the functionality of hybrid and Bayesian-based CNN-RNN architectures. This finding correlates well to the fact that, despite the crucial functionality and integral involvement of advanced classical and AI-driven architectures, expert-level attention to detail serves as the determining factor for advanced functionality and performance. This work illustrates the high degree of requisite attention necessary during the development and design process for an efficient cloud network intrusion technique, maintaining the inherent import and potential that offers the involvement and development of high-performing classical architectures.

Keywords: Network; Cloud; BCC cPacket; Machine Learning; DDoS

1. Introduction

Cloud computing services are now deemed essential in day-to-day life in both India and other parts of the world. Cloud computing underpins many areas including banking, education, healthcare, governments, and different business transactions. With the increase in the number of system transactions moving towards cloud computing platforms, cases of Distributed Denial-of-Service (DDoS) attacks have consequently increased over time. Recent occurrences highlight an increase in widespread attacks launched against cloud networks [1]. DDoS disrupts server functionality by flooding networks with unnecessary traffic. This slows or renders different internet services unavailable.

A considerable amount of research has been carried out in attempts to understand the nature of DDoS attacks in order to achieve earlier detection. Various research endeavors have contributed toward creating relevant datasets and classifications that define the nature of contemporary threats [2]. The appearance of fresh cloud-oriented datasets and traffic analysis research further illustrates the adaptive model of DDoS threat behaviors in attempts to thwart detection by conventional methods based solely upon predefined DDoS threat rules [3]. Consequently, research has shifted toward implementing machine learning algorithms in attempts to increase detection accuracy. The result indicates that supervised machine learning algorithms possess the capability of detecting traffic behaviors and DDoS threats more accurately than conventional methods [4, 5]. Further research has also optimized conventional Naïve Bayes approaches with minor improvements in feature processing in cloud-oriented platforms [6]. Further research indicates successful outcomes from classification-oriented approaches in predicting DDoS threats with exceptionally high accuracy levels [7]. Increasingly, attention mechanisms implemented in CNN-BiLSTM models have proved adept in mastering sophisticated patterns in network traffic data [8,9]. The earlier works conducted in feed-forward neural networks [10] and in LSTM networks [11] indicated the efficacy of deep nets in dealing with voluminous traffic data. Certain research works consider multiple cloud configurations and cooperative schematics towards improving reaction times [12]. Moreover, more recent research has utilized adversarial training in improving detection efficacy against evasive methods [13].

A. Need for a Simple and Effective Detection Pipeline

Despite the existence of various advanced approaches, effective cloud platforms still need detection mechanisms that are both resource-lean and simple to use. Organizations would like solutions that support fast deployment with little preprocessing and the capability to perform well under realistic conditions. This causes the need for the development of effective machine learning workflows that involve exploration of the dataset, removal of inaccuracies and irrelevant data, feature encoding, and generation of interpretable models that are able to identify traffic type under the attack category or not under either attack or suspicious category. This would allow early detection and enhance the security of the cloud platform with little computational effort.

2. Related Work

The use of DDoS detection based on the system in paper [14] involves the use of only a CNN in detection. The network traffic data is converted to feature inputs, and the CNN model detects signatures. The system works well in detecting flash bursts common in DDoS attacks. The system also tests the stability of the model in detecting DDoS attacks using various datasets. The outcomes from the system testing indicate that deep feature extraction works well in real-time DDoS detection. The system concludes that CNN detection works well in the cloud.

In research paper [15], the research proposes a hybrid approach to detecting intrusions in networks using a methodology known as DCNN-BiLSTM. The methodology involves the use of a deep CNN block combined with a BiLSTM layer. The deep CNN block is able to decode local traffic patterns while the BiLSTM layer captures the long-term dynamics of network traffic. This combination proves effective in detecting both sudden and gradual traffic patterns commonly seen in network attacks. The experimental outcomes reveal high levels of accuracy and F1-measure values in

detecting intrusions in network traffic. The study indicates that the use of both spatial and temporal information provides more accurate detection of intrusions.

In paper [16], the proposal involves implementing DDoS detection using a deep neural network with autoencoders acting as preprocessing modules. The autoencoders eliminate unwanted noise in traffic data by compressing high-dimensional data before the classification process. The feed-forward network classifies traffic patterns as either normal or DDoS attacks. The autoencoders generally increase precision and recall in testing using publicly available datasets and help overcome overfitting in the network by providing a clean representation of network traffic. The system demonstrates enhanced detection accuracy compared to conventional approaches in detecting DDoS traffic.

The paper in [17] describes the detection and defense methodology developed using LSTMs in the context of cloud networks. The system examines sequences of traffic patterns instead of individual packets in attempt to follow the process of the attack over time. A simple defense system has been added and implemented after detection. The results obtained in the context of the cloud datasets depict improved accuracy and fast detection rates. The approach proves effective in detecting Low Rate and Dynamic DDoS Attacks, and the results depict the efficiency of LSTM networks in treating the necessary traffic correlations.

in paper [18], the authors put forth a multi-scale CNN architecture for identifying DDoS attacks. The architecture makes use of multiple-scale filters in the convolutional layer to identify both detailed and overall traffic information. The experimental results validate the effectiveness of the proposed multi-scale feature extraction in improving the accuracy of classifications compared with single-scale CNN approaches. The proposed scheme is also efficient in terms of computational complexity and traffic data processing latency. The authors further conclude that feature extraction through multiple scales proves effective in detecting different types of attacks.

A combined IDS system based on CNN and BiLSTM networks is proposed in paper [19]. The CNN network focuses on detecting the spatial features, and the BiLSTM network focuses on the bidirectional temporal features of the sequences. The results obtained in this research clearly indicate that the combined system outperforms individual CNN and simple LSTM networks in terms of accuracy and can handle both binary and multiclass classifications of the intrusion datasets effectively. The research proposes that the combined architectures improve the general detection capacity of networks. The proposed model in research paper [20] utilizes both self-attention and inter-sample attention in the transformer structure to identify and prevent DDoS attacks based on the TCP protocol in the cloud platform. This further helps the model not only highlight key traffic features but also grasp the connections among different samples. The experimental results demonstrate high accuracy and fast detection rate in high cloud traffic scenarios. The proposed model outperforms conventional LSTM and CNN models in most parameters. The results further confirm significant enhancement in the detection of fine-grained attacking behaviors. The conclusion draws attention towards the flexibility and accuracy of attention models in ensuring cloud security.

Table 1. Review of Existing Models

Ref.	Dataset / Methods	Results	Conclusion
[21]	Deep-learning IDS using traffic features and neural network layers.	Accuracy and F1-score are above 99%.	Shows deep learning can detect DDoS with very high reliability.
[22]	Text-based transformer model (BERT variant) for DDoS traffic messages.	Accuracy above 98% on multiple datasets.	Proves that text-based features and transformers help in early DDoS detection.

[23]	Hybrid deep model for IoT DDoS detection using stacked networks.	Achieved around 98–99% precision and recall.	Confirms hybrid deep learning performs well in IoT attack detection.
[24]	CNN-BiLSTM method with hybrid feature selection for SDN networks.	Accuracy above 99% with reduced false alarms.	Shows combining convolution and temporal learning improves intrusion detection.
[25]	Deep learning + hybrid structure to identify DoS/DDoS in SDN.	Performance above 98% for attack classification.	Hybrid deep learning gives stable and accurate detection in SDN.
[26]	Machine-learning model with feature selection and tuned parameters.	Accuracy around 97–98%.	Feature selection and tuning improve traditional ML model performance.
[27]	Hybrid deep model with optimization (spotted hyena algorithm).	Achieved 99% accuracy and balanced metrics.	Optimization techniques can further boost deep-learning IDS performance.
[28]	CNN with feature reduction techniques for intrusion detection.	Accuracy reported above 98%.	Feature reduction plus CNN leads to efficient and accurate IDS.
[29]	Multi-model approach for SCADA DDoS detection.	Accuracy around 97–99% depending on classifier.	Multi-model systems improve security in SCADA environments.
[30]	Entropy-based early detection for SDN-IoT using stochastic technique.	Significant reduction in attack impact and detection delay.	Entropy analysis helps early detection before heavy damage.
[31]	Near real-time CNN-based IDS for SDN-IoT networks.	Accuracy above 99% with low latency.	CNN models can be deployed in near real-time environments.
[32]	Deep learning with Capuchin Search Algorithm for feature optimization.	Accuracy around 98–99%.	Search optimization improves model stability and reduces errors.
[33]	Work explains importance of normalization for ML preprocessing.	Shows improved consistency and faster convergence.	Normalization is essential for training stable ML models.
[34]	SMOTE oversampling technique for imbalanced datasets.	Improves recall of minority classes without harming accuracy.	Oversampling solves imbalance and helps in intrusion detection datasets.
[35]	Introduction to CNN architectures explaining convolution theory.	Demonstrates strong ability to extract spatial features.	CNN is useful for traffic pattern learning in IDS tasks.
[36]	LSTM architecture explaining sequence learning.	Useful for long pattern dependencies.	LSTM helps detect time-based behaviour of attacks.
[37]	Encoder-decoder model concepts for sequence tasks.	Explains structured mapping useful for anomaly detection.	Encoder-decoder helps compress and analyse complex traffic.
[38]	Multi-head attention mechanism for better feature learning.	Improves feature weighting and detection accuracy.	Attention improves IDS performance in complex cloud traffic.

[39]	Review of modern deep-learning architectures and applications.	Summarizes strengths of CNN, LSTM, hybrids, and transformers.	Confirms deep learning is now central in intrusion detection research.
[40]	CNN-GRU hybrid for time-series prediction (GNSS deformation).	High prediction accuracy using combined spatial-temporal features.	Shows CNN-GRU combination is effective for sequence-based tasks like IDS.

2. Method

The research applies to a systematic and effective procedure in designing the cloud-based system for intrusions. The methodology has four stages as shown in fig.1. Stage One deals with the in-depth analysis of the dataset. Stage Two handles the preprocessing of data. Stage Three comprises the division of the dataset performed either through stratified or simple sampling based on the research requirements. Stage Four deals with the generation of machine-learning models. The four stages altogether address the integrity of the dataset and the extraction of relevant patterns that define both the attacking and harmless traffic. Moreover, the discussion regarding the performance parameters would be dealt with in the Results section.

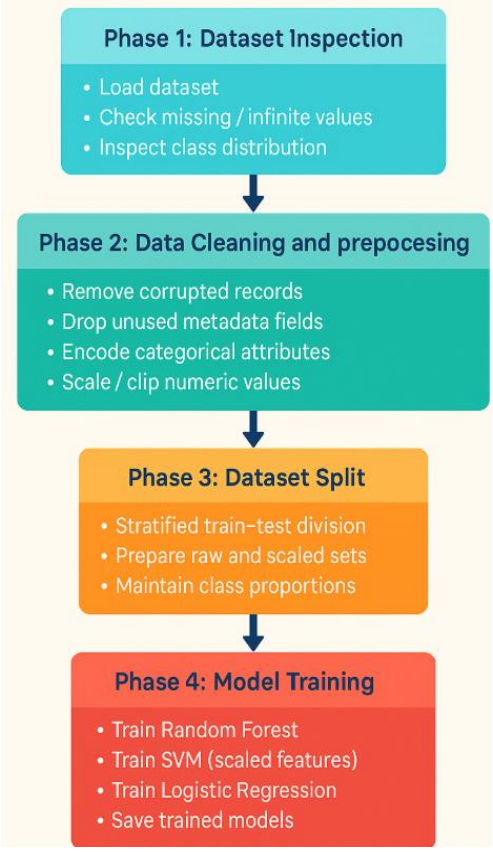

Figure 1 Proposed Model

A. Phase 1: Dataset Inspection

(Arial 10, Normal) The first step is to check the dataset carefully before any processing. This ensures that the data is consistent, complete, and suitable for model training. The dataset is loaded and scanned for its total size, number of features, presence of missing values, and any infinite values that can disturb model behaviour. The label column is also inspected to understand how many records belong to each class. The basic structure of the dataset is described using simple statistical measures. To support the inspection process, the equations from (1) to (5) help summarize the dataset:

Total number of records

$$N = \sum_{i=1}^{n} 1 \qquad (1)$$

Missing value counts for a feature

$$M_j = \sum_{i=1}^{N} I\left(x_{ij} = \text{NaN}\right) \qquad (2)$$

Proportion of missing values

$$P_j = \frac{M_j}{N} \qquad (3)$$

Class count for a label

$$C_k = \sum_{i=1}^{N} I\left(y_i = k\right) \qquad (4)$$

Class proportion

$$R_k = \frac{C_k}{N} \qquad (5)$$

Here,

- $N$is the total number of samples,
- $M_j$is missing values in feature j,

- $P_j$is the missing proportion,
- $C_k$is number of samples in class k,
- $R_k$is the class ratio,
- $I(\cdot)$is an indicator that returns 1 when the condition is true and 0 when false.

These calculations help identify any imbalance, inconsistency, or error in the dataset. After completing this inspection, the dataset is ready for the next phase where cleaning and preprocessing take place.

B. Phase 2: Data Cleaning and Pre-processing

(Arial 10, Normal) This stage focuses on data preparation to allow models to learn from clean and meaningful traffic patterns. After the inspection step, the data may include irrelevant data in rows, unused metadata columns, data of different types, and data values requiring scaling. The data preprocessing involves removing the placeholder data in rows where either the label or the activity columns contain text instead of valid data values. Metadata columns in datasets are not essential in data learning; hence, this step involves removing such columns to minimize data dimensionality while preserving predictiveness. After data preprocessing, there would be categorical attributes requiring numerical representation. The simple label-encoding function converts each unique string into an integer. This transformation is expressed through a mapping function $E(\cdot)$in equation (6), where an input category is replaced by its encoded representation:

$$x_{ij}^{(enc)} = E(x_{ij}) \qquad (6)$$

After encoding, the dataset is separated into features and labels. This step helps prepare the data for model training. The separation process can be written through equation (7), where Xcontains only feature values and $y$stores the class labels extracted from the dataset:

$$X = x_{i1}, x_{i2}, \dots, x_{im}, y = y_i \qquad (7)$$

Some features may have very large or very small numeric scales, which can affect the performance of models like SVM and logistic regression. Therefore, a scaling transformation is applied. A standard scaling method normalizes each feature by subtracting the mean and dividing by the standard deviation as shown in equation (8):

$$x_{ij}^{(scaled)} = \frac{x_{ij} - \mu_j}{\sigma_j} \qquad (8)$$

During cleaning, duplicate records must be removed so that the training phase does not learn repeated values that give false confidence. The removal of duplicate rows can be represented mathematically in equation (9), where the cleaned dataset D'contains only unique rows:

$$D' = \{x_i \mid x_i \neq x_k \quad \text{for all } i \neq k\} \qquad (9)$$

Another important part of preprocessing is handling infinite or very large abnormal values detected during inspection. Such values are replaced using a clipping transformation. The clipping rule is written in equation (10), where values exceeding a chosen bound B are replaced by the boundary value:

$$x_{ij}^{(clip)} = \begin{cases} B, & x_{ij} > B \\ -B, & x_{ij} < -B \\ x_{ij}, & \text{otherwise} \end{cases} \qquad (10)$$

Finally, once all transformations are applied, the processed feature matrix and label array are stored for later phases. The final cleaned feature matrix can be formalized as in equation (11), representing the complete set of encoded, scaled, and validated traffic features:

$$X^{(final)} = [\, x_{ij}^{(clip,scaled,enc)} \,]_{N \times m} \qquad (11)$$

These operations ensure that the dataset is consistent, free from formatting errors, and suitable for learning. The dataset becomes ready for splitting into training and testing partitions in the next phase, without carrying any noise or structural issues that may affect the performance of intrusion detection models.

C. Phase 3: Dataset Splitting

After cleaning and preparing the dataset, the next step is to divide it into training and testing sets in a controlled and balanced manner. This phase ensures that the system learns from one part of the data and

is evaluated in another part that it has never seen before. A simple random split is not enough for intrusion detection because the number of attacks, benign, and suspicious samples may not be equal. If the split is not balanced, the model may learn biased patterns. Therefore, a stratified splitting strategy is used so that each class keeps the same proportion in both subsets.

The dataset is first represented as a complete pair of features and labels, shown in equation (12), where every feature vector $x_i$corresponds to a class label y_i:

$$D = \{(x_i, y_i)\}_{i=1}^{N} \quad (12)$$

To maintain the original class proportions during splitting, the ratio of each class is preserved. This requirement is expressed in equation (13), where the class proportion in the training set must match the overall dataset:

$$|D_train| \,/\, C_k\hat{}(train) \;\approx\; N_\{C_k\} \quad (13)$$

The dataset is then divided into two disjoint sets: the training set $D_{train}$and the testing set D_{test}. This is represented mathematically in equation (14), ensuring no record appears in both sets:

$$D = D_{train} \cup D_{test}, D_{train} \cap D_{test} = \emptyset \quad (14)$$

Since some models require scaled data while others do not, two separate feature sets are created for training. Models such as SVM and logistic regression use scaled features, while Random Forest operates on raw values. This transformation is defined in equation (15), where a scaling function $S(\cdot)$is applied only to selected models:

$$X_{train}^{(scaled)} = S(X_{train}), X_{test}^{(scaled)} = S(X_{test}) \quad (15)$$

To maintain reproducibility, a fixed random state is used. This ensures that every run of the split produces the same division of training and testing data. The final representation of all sets used in this phase is given in equation (16), capturing raw and scaled versions:

$$\left(D_{train}, D_{test}, X_{train}^{(scaled)}, X_{test}^{(scaled)}\right) \quad (16)$$

These steps guarantee that the dataset is divided fairly and consistently. The training set contains balanced samples from all classes, and the testing set gives an unbiased estimate of how well the intrusion detection system performs on unseen traffic. This controlled division forms the foundation for building reliable models in the next phase.

D. Model Development

Once the dataset is divided into training and testing sets, the next step is to develop the intrusion detection models. This phase focuses on training three different machine-learning models: Random Forest, Support Vector Machine, and Logistic Regression. Each model learns patterns from the training data and builds a decision function that can classify network traffic into attack, benign, or suspicious categories. The training process uses the cleaned features prepared in earlier phases and applies model-specific learning rules to form decision boundaries.

The model receives the training set as input, which is represented in equation (17). Each feature vector $x_i$in the training data is paired with its correct class label y_i, forming the core learning material:

$$D_{train} = \{(x_i, y_i)\}_{i=1}^{T} \quad (17)$$

The Random Forest model creates multiple decision trees. Each tree learns patterns independently, and the final decision is taken by majority voting. This ensemble mechanism is expressed in equation (18), where the predicted class is the most common output among all trees:

$$\hat{y} = \text{mode}(h_1(x), h_2(x), \ldots, h_M(x)) \quad (18)$$

For the Support Vector Machine, the model learns a separating boundary between classes. The decision function is built using a weight vector and a bias term. The core SVM decision rule is shown in equation (19), where the sign of the linear score determines the predicted class:

$$f(x) = w^{\top}x + b \quad (19)$$

Logistic Regression is also used to classify traffic by learning weights that map input features to class probabilities. The logistic transformation is applied to convert the raw score into a probability. This behaviour is captured in equation (20), which defines the logistic output without using any fractional form:

$$p = \frac{1}{1 + e^{-z}} \text{ where } z = w^{\top}x + b \qquad (20)$$

Training these models requires adjusting their internal parameters so that the error is minimized on the training set. A general optimization rule is used across all models, represented in equation (21), where parameters θ are updated to reduce an error function $L(\theta)$:

$$\theta_{\text{new}} = \theta_{\text{old}} - \alpha \nabla L(\theta) \qquad (21)$$

Through these learning steps, each model develops its own way of recognizing patterns in the traffic data. After training, the models are saved and prepared for evaluation using the testing set.

3. Results

In this section, the performance of the three trained models, namely Random Forest, Support Vector Machine (SVM), and Logistic Regression, on the testing dataset would be discussed. Results would be interpreted based on the confusion matrix and performance metrics such as accuracy, macro-precision, macro-recall, macro-F1, and macro-ROC AUC. The confusion matrix for the Logistics Regression classifier appears in Figure 2, showing that it classifications correctly the vast majority of both attacking and benign examples, and comparatively well the suspicious network traffic. In Figure 3, the confusion matrix for the Random Forest classifier plots the high accuracy of the classifier for all three classes, and the low corresponding misclassifications. In the SVM classifier, the confusion matrix of Figure 4 illustrates high accuracy, especially for the benign and the suspicious class, and low misclassifications.

In the various models, the model that returns the highest performance values for accuracy, macro-precision, macro-recall, macro-F1, and macro-ROC-AUC is the Random Forest model, which returns 0.9983, 0.9989, 0.9973, 0.9981, and 0.9999, respectively. The accuracy, macro-precision, macro-recall, and macro-F1 for the SVM model are 0.9968, 0.9980, 0.9925, and 0.9952, respectively. Accuracy, macro-precision, macro-recall, and

The comparative study reveals that the most stable and best-performing model for all the classes of traffic is the Random Forest model, followed closely by the SVM model, which performs well for high-throughput cloud traffic. The performance of the Logistic Regression model is good but less effective for the suspicious traffic compared to the other two models. The use of the cleaned features, stratified data split, and scaling helps to achieve good performance for all the categories.

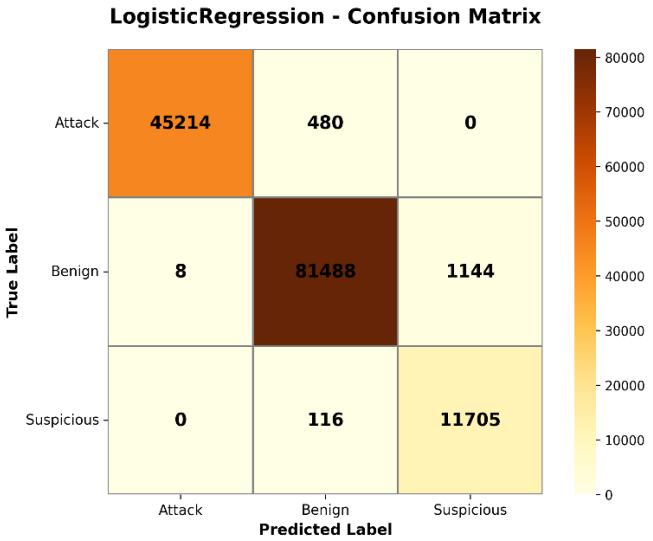

Fig. 2. Confusion Matrix for Logistic Regression

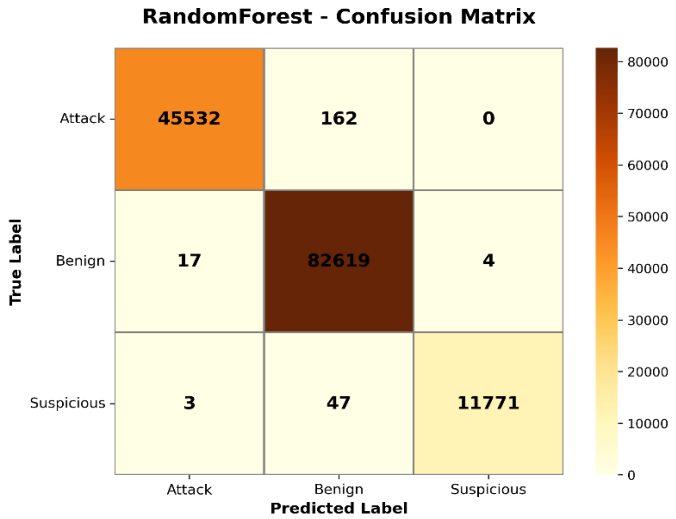

Fig. 3. Confusion Matrix for Random Forest

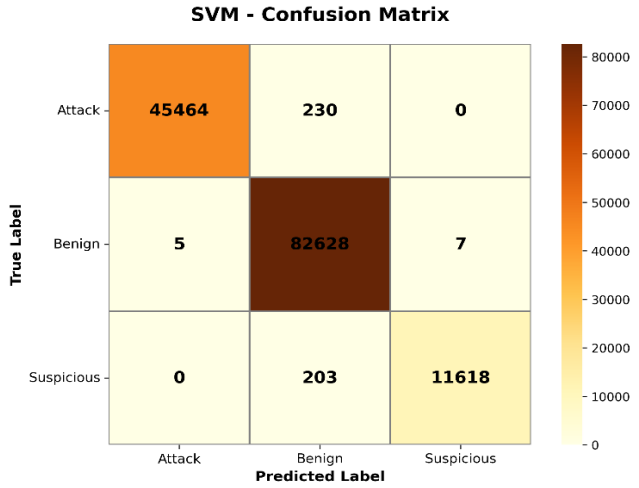

Fig. 4. Confusion Matrix for SVM

4. Discussion

Results of the intrusion detection system show that the performance of the three algorithms on the cloud traffic dataset is robust, and that the accuracy of the model, as achieved by the Random Forest algorithm, is the highest. The confusion matrices show that the accuracy of the model, as achieved by the Random Forest algorithm, is the highest, and that it maintains the lowest levels of errors for the attack, benign, and suspicious traffic. The SVM algorithm also maintains low levels of errors. The purpose of comparing our system with existing works as shown in Table 2, the results achieved are compared to two state-of-the-art cloud-centered works on DDoS detection. In Paper [41], the authors utilized the hybrid CNN-RNN model, which incorporated the multi-head attention mechanism, and achieved an accuracy of 99.32%, precision of 99.28%, recall of 99.30%, and F1-score of 99.29%. In Paper [42], the authors put forward the Bayesian-based CNN and achieved 98.65% accuracy, 98.72% precision, 98.61%, and 98.66% F1-score.

In comparison to these approaches, the accuracy of 99.83% for our proposed Random Forest model far exceeds the accuracy of both [41] and [42]. The macro-precision value of 99.89% and macro-recall of 99.73% for the proposed model are also higher compared to the performance of the hybrid deep learning approaches. This indicates that even without the use of deep learning approaches, it is conceivable that an effective classical approach that

maintains proper features and proper distribution of the data can provide significantly superior performance for cloud intrusion detection. In our experiment, the accuracy of the SVM model compared to the deep learning strategy of the approach in study [41] can be regarded as considerable. These experiments prove that, if preprocessing, data partitioning, and the management of varying features are well handled, classical models can compete and even beat deep networks. This becomes especially important for the cloud, where the simplicity and fast processing of the model are priority factors.

Table 2. Performance Evaluation with Existing Models

Model	Accuracy	Precision	F1-Score
Proposed RF	0.9983	0.9989	0.9981
Proposed SVM	0.9968	0.9980	0.9952
Proposed LR	0.9875	0.9678	0.9776
[41] CNN–RNN + Multi-Head Attention	0.9932	0.9928	0.9929
[42] Bayesian-CNN + Data Fusion	0.9865	0.9872	0.9866

5. Conclusion

(Arial 10, Normal) The paper offers an easily understandable and repeatable method for the use of cloud-based intrusion detection involving proven machine learning algorithms, supported by effective data processing. Accuracy and conformity were first ascertained and confirmed on the initial evaluation of the data, followed by extensive cleaning and encoding to make the features significant and error-free. An even split for training and testing supported effective learning and generalization. On comparing the three algorithms, the best performance was achieved by the Random Forest algorithm, which outperformed the performance achieved by two other state-of-the-art deep learning-based algorithms. The performance achieved of accuracy or more than 99% here confirms that superior performance cannot come from sophisticated and resource-intensive deep learning networks, but the performance depends on the efficacy of data processing and the robust nature of the learning procedure. SVM performed satisfactorily, and the performance of the Logistic Regression algorithm, though simple, remained reliable. This result provides an effective validation that an expertly preprocessed data source, coupled with classical algorithms, can be well applied to the practical aspects of the cloud environment that emphasize rapid processing, understandability, and resource management.

Acknowledgment

The authors are thankful to Amity University Uttar Pradesh, Lucknow Campus, and Babu Banarasi Das University, Lucknow, for their support and facilities that were necessary for the conduct of the research. The authors are grateful for the support and motivations that came from the faculty members and research guides, which played an important role in the successful accomplishment of the research. Acknowledgments are also extended for the support and help that were obtained during the process of data collection, preprocessing, and experiments. Productive discussions that took place among the authors were helpful for the improvement of the research and for making the research efficient. Open-source software and libraries that were available played an important role in the development and evaluation of the proposed intrusion detection systems.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data Availability

The dataset used in this research is publicly accessible and can be downloaded from the official source of the BCCC–cPacket Cloud DDoS 2024 dataset at: https://www.kaggle.com/datasets/dhoogla/bccc-cpacket-cloud-ddos-2024

All preprocessing scripts, trained models, confusion matrices, and evaluation files generated during this study are available from the corresponding author upon reasonable request. The processed feature matrices and label files created during analysis are stored securely and may also be shared for academic and non-commercial research use. No confidential or restricted data has been used, and all resources required to reproduce the findings remain openly accessible.

Author Contribution

All authors contributed equally to the development and completion of this research. The first author led the dataset preparation, preprocessing, and model implementation. The second author guided the experimental design, evaluation strategy, and interpretation of results. The third author supported the literature review, methodological refinement, and manuscript improvement. All authors reviewed the final version of the paper and approved it for submission.

Declarations

Ethical Approval

This study did not involve human participants, personal data, or any form of intervention requiring ethical clearance. All experiments were conducted using publicly available datasets and open-source tools. As no identifiable or sensitive information was used, formal approval from an Institutional Review Board (IRB) was not required. The research followed standard ethical practices for data handling, privacy, and responsible use of computational resources.

Consent for Publication Participants.

Consent for publication was given by all participants

Competing Interests

The authors declare no competing interests.

References

StormWall, "DDoS Attack Report 2024," StormWall Blog, 2024.

I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy," in Proc. Int. Carnahan Conf. Security Technol., Chennai, India, 2019.

M. Shafi, A. H. Lashkari, V. Rodriguez, and R. Nevo, "Toward generating a new cloud-based distributed denial of service (DDoS) dataset and cloud intrusion traffic characterization," Information, vol. 15, p. 195, 2024.

A. Fathima, G. S. Devi, and M. Faizaanuddin, "Improving distributed denial of service attack detection using supervised machine learning," Measurement: Sensors, vol. 30, p. 100911, 2023.

C. Liu and S. Zhong, "DDoS attack detection method based on machine learning," in Proc. 15th Int. Conf. Softw. Eng. Service Sci., Changsha, China, 2024.

S. Naiem, A. E. Khedr, A. M. Idrees, and M. I. Marie, "Enhancing the efficiency of Gaussian Naïve Bayes classifier in detecting DDoS in cloud computing," IEEE Access, vol. 11, pp. 124597–124608, 2023.

Ismail, I. Mohand, H. Hussain, A. A. Khan, U. Ullah, and M. Zakarya, "Machine learning-based classification and prediction technique for DDoS attacks," IEEE Access, vol. 10, pp. 21443–21454, 2022.

J. Zhao, Y. Liu, Q. Zhang, and X. Zheng, "CNN-AttBiLSTM mechanism: A DDoS attack detection method," IEEE Access, vol. 11, pp. 136308–136317, 2022.

A. A. Najar and S. M. Naik, "DDoS attack detection using CNN-BiLSTM with attention," Telemat. Inform. Rep., vol. 18, p. 100211, 2025.

A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed-forward neural network model," Expert Syst. Appl., vol. 169, p. 114520, 2021.

C. M. Hsu, M. Z. Azhari, H. Y. Hsieh, S. W. Prakosa, and J. S. Leu, "Robust network intrusion detection using LSTM-based CNN," Mobile Netw. Appl., vol. 26, pp. 1137–1144, 2021.

A. Abusitta, M. Bellaiche, M. Dagenais, and T. Halabi, "Proactive multi-cloud cooperative intrusion detection system," Future Gener. Comput. Syst., vol. 98, pp. 308–318, 2019.

M. Abdelaty, S. S. Hayward, R. Doriguzzi-Corin, and D. Siracusa, “GADoT: GAN-based adversarial training for robust DDoS detection,” in Proc. IEEE CNS, 2021.

A. A. Najar and M. Naik, “A robust DDoS intrusion detection system using CNN,” Comput. Electr. Eng., vol. 117, p. 109277, 2024.

V. Hnamte and J. Hussain, “DCNN-BiLSTM intrusion detection system,” Telemat. Inform. Rep., vol. 10, p. 100053, 2023.

A. L. Yaser, H. M. Mousa, and M. Hussein, “Improved DDoS detection using DNN and autoencoder,” Future Internet, vol. 14, p. 240, 2022.

H. Aydin, Z. Orman, and M. A. Aydin, “LSTM-based DDoS detection and defence in cloud,” Comput. Secur., vol. 118, p. 102725, 2022.

J. Cheng, Y. Liu, X. Tang, V. Sheng, M. Li, and J. Li, “DDoS attack detection via multi-scale CNN,” Comput. Mater. Contin., vol. 62, pp. 1317–1333, 2020.

A. A. Jihado and A. S. Girsang, “Hybrid CNN–BiLSTM intrusion detection,” J. Adv. Inf. Technol., vol. 15, pp. 219–232, 2024.

G. Kirubavathi, I. R. Sumathi, J. Mahalakshmi, and D. Srivastava, “TCP-based DDoS mitigation using transformer attention model,” J. Supercomput., vol. 81, p. 474, 2025.

D. Akgun, S. Hizal, and U. Cavusoglu, “Deep-learning DDoS intrusion detection model for cybersecurity,” Comput. Secur., vol. 118, p. 102748, 2022.

T. T. H. Le, S. Heo, J. Cho, and H. Kim, “DDoSBERT for DDoS detection,” Comput. Netw., vol. 262, p. 111150, 2025.

A. Ahmim, F. Maazouzi, M. Ahmim, S. Namane, and I. B. Dhaou, “Hybrid deep learning for IoT DDoS detection,” IEEE Access, vol. 11, pp. 119862–119875, 2023.

R. B. Said, Z. Sabir, and I. Askerzade, “CNN-BiLSTM intrusion detection for SDN,” IEEE Access, vol. 11, pp. 138732–138747, 2023.

H. Elubeyd and D. Y. Kaplan, “Hybrid deep learning for automatic DoS/DDoS detection in SDN,” Appl. Sci., vol. 13, p. 3828, 2023.

R. K. Batchu and H. Seetha, “Generalized ML model for DDoS detection with hybrid feature selection,” Comput. Netw., vol. 200, p. 108498, 2021.

F. Alhayan et al., “Hybrid deep learning with spotted hyena optimisation for cloud IDS,” J. Radiat. Res., vol. 18, p. 101523, 2025.

T. Xiao, C. Xing, T. Zhang, and Z. Zhao, “IDS based on feature reduction and CNN,” IEEE Access, vol. 7, pp. 42210–42219, 2019.

E. Sogüt and O. A. Erdem, “Multi-model proposal for DDoS detection on SCADA,” Appl. Sci., vol. 13, p. 5993, 2023.

I. Varalakshmi and M. Thenmozhi, “Entropy-based early DDoS detection in SDN-IoT,” Measurement: Sensors, vol. 39, p. 101873, 2025.

M. V. O. D. Assis, L. F. Carvalho, J. J. P. C. Rodrigues, J. Lloret, and M. L. Proença Jr., “Near real-time IDS for SDN-IoT using CNN,” Comput. Electr. Eng., vol. 86, p. 106783, 2020.

M. A. Elaziz, M. A. A. Al-Qaness, A. Dahou, R. A. Ibrahim, and A. A. A. El-Latif, “Deep learning with Capuchin search for cloud and IoT IDS,” Adv. Eng. Softw., vol. 176, p. 103402, 2023.

S. G. K. Patro and K. K. Sahu, “Normalization: A preprocessing stage,” Int. Adv. Res. J. Sci. Eng. Technol., vol. 2, pp. 20–22, 2015.

N. V. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, “SMOTE: Synthetic minority oversampling technique,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

K. O’Shea and R. Nash, “Introduction to convolutional neural networks,” arXiv:1511.08458, 2015.

S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Comput., vol. 9, pp. 1735–1780, 1997.

K. Cho et al., “Encoder–decoder properties in neural machine translation,” arXiv:1409.1259, 2014.

J.-B. Cordonnier and A. L. Martin, “Multi-head attention: Collaborate instead of concatenate,” in Proc. ICLR, Vienna, Austria, 2021.

I. D. Mienye and T. G. Swart, “Review of deep learning architectures and applications,” Information, vol. 15, p. 755, 2024.

Y. Xie et al., “Deep learning CNN-GRU for deformation monitoring prediction,” Appl. Sci., vol. 14, p. 4004, 2024.

Sathaporn et al., “DDoS detection using hybrid CNN–RNN with multi-head attention,” Appl. Sci., vol. 15, p. 11567, 2025.

I. AlSaleh, A. Al-Samawi, and L. Nissirat, “Bayesian-based CNN with data fusion for cloud DDoS detection,” Sensors, vol. 24, p. 1418, 2024.