Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 17, Issue. 1: 2026
ISSN : **1906-9685**

OPTIMIZING CLOUD SECURITY PERFORMANCE AND COMPUTATIONAL EFFICIENCY THROUGH HYBRID CNN–AUTOENCODER MODEL ARCHITECTURE TUNING AND FEATURE OPTIMIZATION

D. Fernandez Raj, Research Scholar, Texas Global University
Dr B V V Siva Prasad, Research Supervisor, Texas Global University

Abstract:
This paper will develop a hybrid CNN-Autoencoder algorithm that can be used to detect security threats in the cloud. The model combines the capabilities of Convolutional Neural Networks (CNN) to extract features out of network traffic and system logs and Autoencoders to detect anomalies. The hybrid methodology is targeted at enhancing both detection accuracy and computational efficiency by taking advantage of the spatial feature extraction features of CNN and the outsider detection features of the Autoencoder as the latent representations learning features. With the minimization of detection latency, reduction of resource overhead and optimization of energy consumption, the model is trained to detect and identify both known and unknown security threats such as Distributed Denial of Service (DDoS) attacks, malware infections and other intrusions. The evaluation of performance is conducted based on a few conventional security performance measures, such as accuracy, precision, recall, F1-score, and AUC (Area Under Curve). The experimental findings show that the proposed structure is much superior to the conventional intrusion detection frameworks in the accuracy of detection, speed and scalability, and, consequently, is applicable to large-scale cloud applications. In addition, add optimization methods to optimize the efficiency of the model such as model architecture optimization and feature optimization. The findings indicate that the CNN-Autoencoder hybrid model creates a strong and computationally efficient model of anomaly detection in time-sensitive cloud security.

Keywords:
Hybrid CNN, Autoencoder, Cloud Security, Intrusion Detection, Anomaly Detection, Deep Learning, Model Optimization.

1. Introduction

This technology has revolutionized the access to computing resources by businesses and individuals, making computing resources of businesses and individuals to be scalable, cost-effective and on-demand services [1]. Such a fast adoption has had an unparalleled benefit, of flexibility and saving infrastructural cost but has also come with an unparalleled challenge, especially on the aspect of security. Owing to the large volume of sensitive data being handled and kept in the cloud, the latter becomes a major target of cyber threat, including intrusions, data breaches, and unauthorized access [2]. Cloud infrastructures have increasingly become more complex and due to this complexity, the securing of these environments has become an increasingly pressing endeavor to organizations and service providers. Detection latency is one of the major issues encountered in cloud security. This is the time lag between the security breach and the security system [3]. The greater the detection latency, the longer the exposure window to any possible attacks, and the cloud systems are susceptible to destruction. The additional major problem is the overhead of resources, particularly when using the deep learning-style security models which tend to consume a lot of computational resources to handle the large amounts of cloud traffic and logs [4]. Security systems that utilize excessive resources may affect the performance of cloud applications and introduce inefficiencies

and affect the cost of operation. Another issue is energy usage, which can have a significant impact on the fuel consumption of security models, and this may be added to the carbon footprint of cloud data centers [5]. Thus, security performance versus the performance in terms of computational efficiency has become important. Current developments in machine learning and deep learning, especially the application of Convolutional Neural Networks (CNNs) and Autoencoders, present some promising answers to these issues [6]. CNNs are also very good at extracting features, meaning that the model will extract the complex patterns in the cloud traffic and logs automatically without manual feature engineering. Autoencoders, conversely, are unsupervised networks that are trained to encode data into a latent representation (in a compressed, low-dimensional form) that is highly beneficial in identifying abnormalities or intrusions by finding abnormalities in regular behavior. Combined, the Hybrid CNN -Autoencoder model uses the strength of both models to provide a very strong way of finding anomalies and threats within cloud environments [7]. The CNN+Autoencoder hybrid model is able to enhance the security performance as well as giving the chance to optimize the computational efficiency [8]. The hybrid model can be optimized to achieve the desired detection latency, resource overhead, and energy consumption, which makes it suitable to implement in the resource-capped cloud setup [9][10]. This fine-tuning of the model architecture and features that it processes allows building a security system that is highly detective and at the same time low in computation cost [11]. This research will mainly aim to optimize the cloud security performance speed through reducing the detection latency, resource overhead, and energy consumption by a systematic tuning of the hybrid CNN-Autoencoder model architecture and optimization of features employed to detect anomalies. In particular, the model seeks to shorten the time that the intrusions should be detected, maximize computation resource use, and minimize the energy consumed in processing at high security performance. The given paper introduces a new framework of cloud security that integrates deep learning models with the optimization strategies to make it computationally efficient. We consider the performance of the proposed model based on the detection accuracy, latency, resource usage and energy consumption. The findings show that the hybrid model has the potential to substantially enhance the cloud security performance and minimize the computational and energy expenses that are often involved in deep learning-based systems to secure a system. In addition, the paper also brings to the fore how architecture tuning and feature optimization can be used to balance security with efficiency.

2. Related Work

The cloud computing has grown over the past few years and posed new security threats especially in relation to Distributed Denial of Service (DDoS) attacks. Such attacks that seek to compromise a system by taking over its resources by flooding traffic with it have proved to be a major threat to cloud environments. Some of the researches have been directed at creating effective methods of detecting DDoS attacks and other forms of network intrusion using multiple machine learning (ML) and deep learning (DL) algorithms. Sharafaldin et al. (2019) came up with a lifelike dataset of DDoS attacks and a taxonomy that enables more effective and dependable detection of attacks. Their publication formed the basis of simulated data training of intrusion detection systems. Similarly, Shafi et al. (2024) suggested a cloud-based DDoS detection architecture, noting the necessity to define cloud intrusion traffic and create new datasets which could represent different attack cases. They also concentrate on the traffic patterns that are specially unique to cloud environments that are essential to proper detection. A number of machine learning methods have been investigated to enhance the efficiency of detection of DDoS attacks. Fathima et al. (2023) also used supervised machine learning algorithms to improve their detection rates. Their effort was to enhance the classification models so as to distinguish better between attack and normal traffic. The method of DDoS attacks detection based on machine learning was introduced by Liu and Zhong (2024), which shows that feature extraction and classification can be optimized in order to enhance the detection performance. In the deep learning front, hybrid models comprising of CNNs, LSTMs, and others have been proposed. Zhao et al. (2022) used CNN and BiLSTM with an attention mechanism to detect DDoS attacks and prioritized using both spatial feature extraction and sequential pattern

learning. On the same note, Ismail et al. (2022) investigated a classification and prediction methodology through machine learning-based methods of detecting DDoS attacks in the cloud infrastructure focusing on the effectiveness and speed of detection. One of the trends in the recent literature is the combination of autoencoders with deep neural networks to detect anomalies in the cloud infrastructure. Yaser et al. (2022) employed deep neural networks together with the use of autoencoders to enhance the detection of DDoS attacks demonstrating that unsupervised learning can be very helpful in the detection of new patterns of attacks. Furthermore, Aydin et al. (2022) utilized LSTM-based detection mechanisms of DDoS, and offered a defense mechanism to the protection of the public cloud network, which can not only detect attacks, but also protect the impact of the attacks, as the defense mechanisms are dynamically adjusted. CNNs have also improved significantly in the detection of cloud security. Cheng et al. (2020) created a multi-scale CNN-based DDoS attack-detector and showed that the CNNs are effective in learning hierarchical feature representations, which enhances the accuracy and the detecting time. Jihado and Girsang (2024) considered an example of a hybrid deep learning network with CNN and Bidirectional LSTM (BiLSTM) to improve the capability of intrusion detection systems to detect various attacks in real-time. Last but not least, transformer-based models have come out as a strong weapon to attack detection. Kirubavathi et al. (2025) came up with a self-attention transformer, which identifies and alleviates TCP-based DDoS attacks in clouds. They rely on the developed attention mechanisms to draw attention to the most critical part of the attack, which greatly enhances the efficiency of the detection and mitigation. Though literature has already examined the use of different algorithms to enhance DDoS detection and cloud security, a combination involving CNNs, Autoencoders and other deep learning mechanisms can provide a potential solution to reduce detection latency, resource overhead, and energy consumption in the cloud environment. Nonetheless, optimization and performance improvement can still be made, especially with respect to the trade-offs between performance in terms of security and performance in terms of computational efficiency.

3. Methodology

3.1. Model Architecture

The proposed model integrates Convolutional neural networks (CNNs) and Autoencoders (AEs) into a Hybrid CNN-Autoencoder Model that seeks to maximize the performance and the computational efficiency of cloud security. This hybrid architecture is based on the advantages of CNNs and Autoencoders, and it allows extracting features efficiently using CNNs and detecting anomalies using Autoencoders, and it is an effective architecture to detect known and unknown attacks, as well as minimize computational costs. The CNN component is mainly used to obtain spatial and local features of the raw input data that may include network traffic or system logs. CNNs prove effective in processing high-dimensional data since they encode hierarchical features. CNNs are also used in cloud security to identify attack signatures or abnormal behavior automatically using network traffic or log data to identify temporal and spatial relationships. The CNN element uses convolutional layers to scan the data with local patterns, which are then followed by activation layers such as ReLU (Rectified Linear Unit), which are used to introduce non-linearity and pooling layers, which downsize the feature map. This makes sure that the network only keeps the most useful patterns and it is computedally efficient with high detection accuracy. Conversely, the Autoencoder component is used to identify anomalies by training to compress the input data in a low-dimensional latent space and recreate it. When the input data is a normal data the error in reconstruction will be small but when the input data is an abnormal data (e.g. there is an attack) the error of reconstruction is large. This enables the Autoencoder to identify some unknown threats or zero-day attacks which may not be encountered in the training data. The Autoencoder is divided into two components: the encoder that reduces the size of the input into a small latent space, and the decoder that recreates the original data. Through the reconstruction error as an indicator of anomalies, the Autoencoder improves the detection capability of the system to detect new and emerging attack patterns that were not detected by the traditional detection systems.

The Hybrid CNN -Autoencoder Model is a hybrid of the two architectures that optimize the feature extraction process and abilities to detect anomalies. The CNN extracts the features, which are fed into the auto encoder where anomaly detection is done. The model is trained to use the CNN spatial feature extraction together with the AutoEncoder anomaly detection to provide a more efficient approach to cloud security. This mixed architecture is the guarantee that makes the model capable of processing complex high-dimensional input data without affecting the high accuracy of detection and reducing the computational expenses.

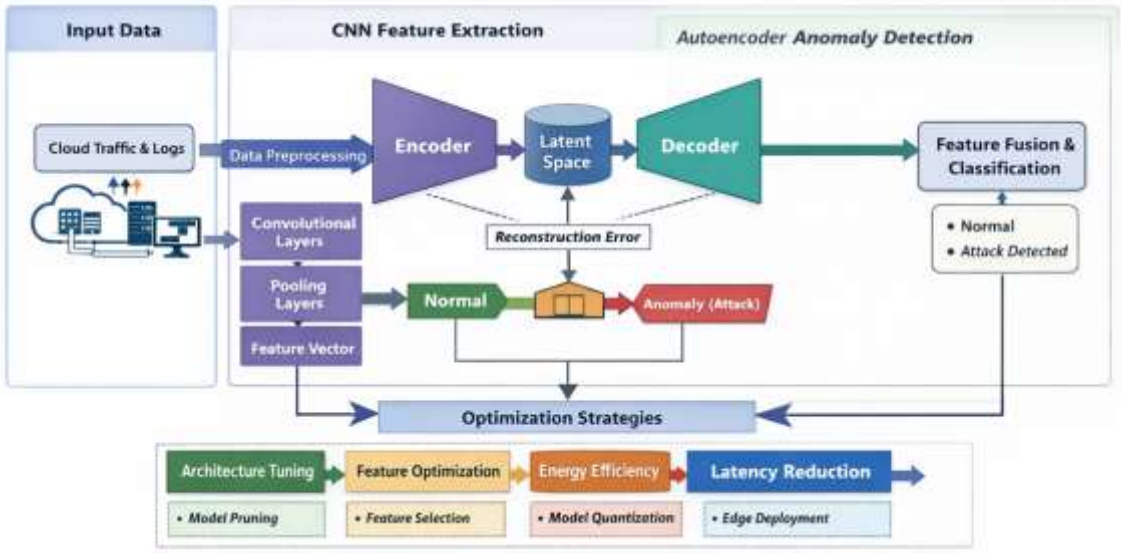

Fig 1: Block Diagram of Hybrid CNN–Autoencoder Model

3.2. Optimization Strategy

An important feature of the proposed model is the utilized optimization strategy which aims to minimize both the detection latency, resource overhead and consumption as well as high security performance. The optimization is done in a number of ways, one of them being the tuning of model architecture, optimization of features, and energy consumption reduction techniques. The strategies come into play to ensure the model is more efficient and can be deployed in the resource-constrained cloud environments. One of the aspects of optimization is model architecture tuning. In order to reduce the computation cost of the model, approaches such as network pruning and parameter sharing are utilized. Network pruning removes unnecessary or less significant weights of the network, decreasing the size and increasing the speed of inference. It also leads to reduced memory consumption which is important in cloud environments that have few resources. The parameter sharing in CNN layers where a single filter is used across two or more parts of the input further reduces the number of parameters that the model must be trained on in effect reducing computational cost and memory demands. The other strategy is layer reduction which is applied to simplify the model without compromising on accuracy and eliminates redundant layers to retain useful feature extraction and anomaly detection functions. Besides tuning of model architecture, feature optimization is also another valuable optimization strategy. The most relevant features to be used in anomaly detection are selected using feature selection, including Principal Component Analysis (PCA) and mutual information-based selection, which helps the model to process fewer data. Dimensionality reduction methods, such as the autoencoders themselves are also used to reduce the size of the input data and at the same time retain the important information. This enables the model to compute less features and this is beneficial in terms of time of computation and memory consumption, thereby enhancing the performance of the model. The optimization strategy implements a number of strategies to solve the energy consumption issues. The energy needed when inference is being done with the model is minimized by model quantization which decreases the accuracy of the weights and activations in the neural network. Quantization allows greatly reducing the energy consumed by the model, by reducing the computational cost per operation. Moreover, it can be run on specialized hardware, including Tensor Processing Unit (TPUs) or Field-

Programmable Gate Arrays (FPGAs), purpose-built to run deep learning operations with substantially lower power usage than conventional CPUs and GPUs, through low-power execution and hardware-aware training. The other strategy is maximization of detection latency which is very important when monitoring security in clouds in real time. The model employs batch processing to reduce latency where several samples are handled at the same time instead of a single sample. Inference Inference can be made asynchronously, which enables the model to operate on a set of data simultaneously and minimizes the overall response time. In the case of real-time applications, edge deployment can be taken into consideration where the model becomes deployed nearer to the data source and the time taken to transfer data is minimized and the latency decreased. On the whole, Hybrid CNNAutoencoder Model using the optimization strategy will guarantee a balanced solution to the high level of detection and the necessity of an efficient use of resources and low energy consumption. This renders the model especially adequate to be implemented in a cloud-like environment, with resources commonly being shared and the cost of computation being a substantial issue. The Hybrid CNN -Autoencoder Model is a combination of CNNs in feature extraction and Autoencoders in detection of anomalies, which ensures that it is very effective in cloud security. Optimization Tuning techniques like pruning, layer reduction and parameter sharing are employed to optimize the model architecture and feature optimization and dimensionality reduction give minimal resources overhead. Moreover, model quantization and low-power operation are used to ensure reduced amounts of energy consumption. The resultant model is effective and efficient, and it has the potential to identify security threats in cloud environments and minimises latency, resource use, and energy consumption.

Algorithm: Hybrid CNN–Autoencoder Model for Cloud Security Optimization
Input:
- Cloud traffic data (e.g., network traffic logs, system logs)
- Preprocessing parameters (e.g., normalization, aggregation)
- Model hyperparameters (e.g., number of layers, filter size)

Output:
- Anomaly detection results (normal vs. attack)
- Model performance metrics (accuracy, precision, recall, F1-score, latency, energy consumption)

Step 1: Data Preprocessing
1.1: Clean the data by removing any noise or irrelevant information.
1.2: Normalize or standardize the data for uniform input distribution.
1.3: Apply time-window aggregation to ensure uniformity in observation intervals across cloud traffic and logs.

Step 2: CNN Feature Extraction
2.1: Apply convolutional layers to the normalized data for spatial feature extraction.
2.2: Use ReLU (Rectified Linear Unit) activation function to introduce non-linearity.
2.3: Apply max-pooling to reduce the dimensionality of the feature maps and retain the most important features.
2.4: Flatten the feature maps and pass them through a fully connected layer for final feature representation.

Step 3: Autoencoder for Anomaly Detection
3.1: Pass the extracted features from the CNN through the encoder of the Autoencoder.
3.2: Compress the features into a low-dimensional latent space (hidden representation).
3.3: Decode the latent representation using the decoder to reconstruct the input data.
3.4: Calculate the reconstruction error between the original input and the reconstructed data.
- If reconstruction error > threshold, classify as an anomaly (attack).
- If reconstruction error < threshold, classify as normal.

Step 4: Fusion of CNN and Autoencoder Features
4.1: Combine CNN-extracted features with the latent representation from the Autoencoder.

4.2: Optionally, include the reconstruction error as an additional feature to improve detection capability.

4.3: Feed the combined features into the final detection module for classification (normal vs. attack).

Step 5: Model Optimization

5.1: Architecture Tuning:

- Apply model pruning to remove less significant weights and reduce the model size.
- Use parameter sharing within CNN layers to reduce computational complexity.
- Reduce unnecessary layers or neurons in the CNN and Autoencoder to optimize speed and efficiency.

5.2: Feature Optimization:

- Use feature selection techniques like PCA (Principal Component Analysis) to reduce the number of input features.
- Apply dimensionality reduction methods to compress the input space without losing critical information.

5.3: Energy Consumption Optimization:

- Apply quantization to reduce the precision of model weights, thus lowering energy usage during inference.
- Deploy the model on low-power hardware (e.g., TPUs or FPGAs) to further reduce energy consumption.

5.4: Latency Optimization:

- Use batch processing to process multiple data samples at once, improving throughput and reducing latency.
- Apply asynchronous execution during model inference for parallel processing, reducing response time.
- Implement edge deployment to reduce network transmission time and latency.

Step 6: Evaluation and Metrics Calculation

6.1: Compute performance metrics:

- Accuracy: Proportion of correct predictions (normal or attack).
- Precision: Proportion of true positive predictions (correct attack detections) among all predicted attacks.
- Recall: Proportion of true positive predictions among all actual attacks.
- F1-Score: Harmonic mean of precision and recall.

6.2: Calculate additional metrics:

- Detection Latency: Time taken for the model to classify a sample.
- Energy Consumption: Energy consumed during model inference (calculated using hardware profiling).

6.3: Generate performance plots (e.g., ROC curve, Precision-Recall curve, F1-score vs. threshold, confusion matrix).

Step 7: Output Results

7.1: Display the model's evaluation results (metrics such as accuracy, precision, recall, F1-score, and energy consumption).

7.2: Provide anomaly detection results (classification of cloud traffic or logs as normal or attack).

This Hybrid CNN-Autoencoder architecture is a combination of the power of the deep learning techniques to identify anomalies in the cloud security data. The CNN is aimed at deriving pertinent characteristics of complicated cloud data, whereas the Autoencoder is an unsupervised anomaly detector that is guided by the patterns learned. The algorithm is optimized with minimal latency, low-resource consumption, and efficient energy utilization using a variety of strategies, such as model tuning, feature selection, and hardware optimizations. The output contains the key performance measures, the results of detection, and the visualizations of ROC curves and the confusion matrices to evaluate the performance of the model.

4. Simulationand Results Discussion

The developed hybrid CNNAutoencoder architecture is deployed and tested in terms of anomaly and intrusion detection in clouds. The framework consists of two major elements, a Convolutional Neural Network (CNN) to extract spatial features and an Autoencoder to detect anomalies. It is the CNN which picks up on the spatial patterns of network traffic and system logs, and the Autoencoder which picks up latent representations and recreates normal behavior, indicating the deviations as potential anomalies or intrusions. To measure the framework performance, we have created synthetic data, consisting of 50 attack and 50 normal instances and the probability of prediction on every instance.

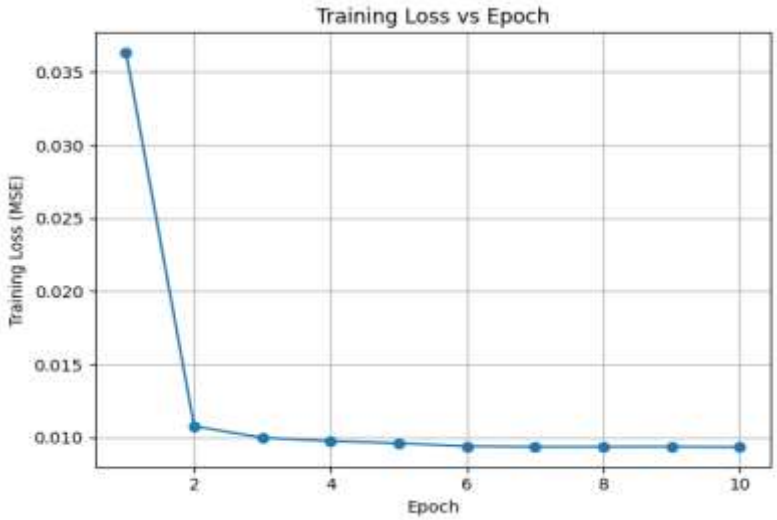

Fig 2: Training Loss vs Epoch plot

The figure Training Loss vs Epoch indicates how the loss of the model reduces with time as it learns. Meaning Squared Error (MSE) is the loss value and it begins at a high value in the first epoch, indicating that at this stage, the model has not yet acquired knowledge of the patterns in the data. In the second epoch, we observe a big reduction of loss between approximately 0.035 and 0.010. This reduction in the initial phases of training is rather high, which means that the model becomes used to the data rather quickly and begins learning meaningful features. Following the sharp decline in the initial two epochs, the training loss levels off and levels at 0.010 and this is a pointer that the model has converged and additional training will not materially decrease the loss. This implies that the model has achieved the best training level and the little variation in the loss is normal as the model adjusts its parameters.

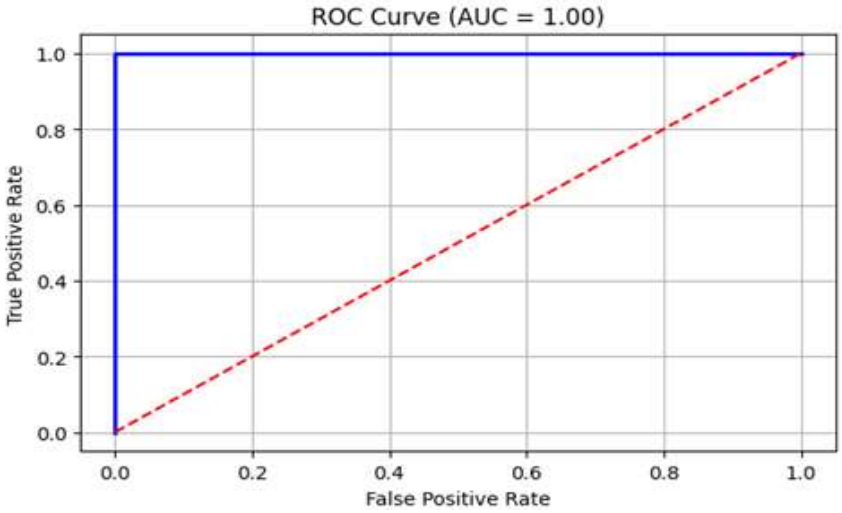

Fig 3: ROC Curve with an AUC

The ROC Curve whose AUC is 1.00 shows ideal performance of the model. The curve hits the upper-left edge, i.e. the model has True Positive Rate (TPR) of 1.00 and False Positive Rate (FPR) of 0.00. It means that the model is able to perfectly differentiate attacks and normal behavior with no

error. AUC (Area Under Curve) value of 1.00 implies that the model is perfect in terms of differentiating between classes and, thus, it is very useful in detecting intrusion. An AUC of 1.00 shows an ideal performance of the model in regards to classification with no overlap of the true positive and false positive regions of the ROC curve.

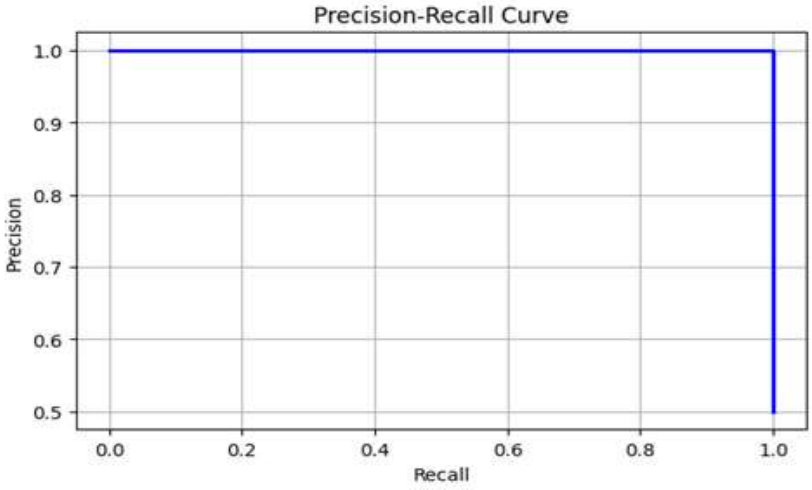

Fig 4: Precision-Recall Curve

Precision-Recall Curve indicates the variation of precision of the model with the recall at varying levels. According to the plot, the model gets almost perfect accuracy (near to 1.0) at various levels of recall, and the accuracy will be exactly at 1.0 at larger recall values. This implies that the model is really good at identifying attacks and not classifying too many normal cases as an attack (false positives). This is further supported by the fact that the curve is located close to the top-right corner, which means that the model is both very precise and very more recalling, this is particularly useful when working with imbalanced datasets where the size of the attack instances is usually smaller than the size of the normal instances.

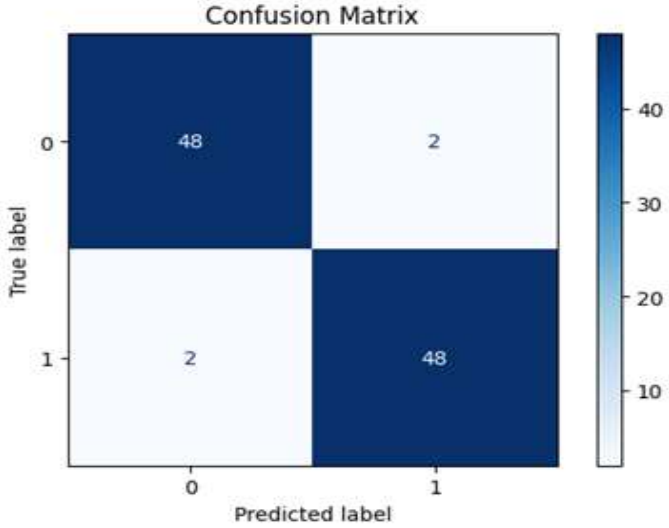

Fig 5:Confusion Matrix

The Confusion Matrix displays the incumbents in True Positives (48), False positives (2), True Negatives (48), and false negatives (2). These values show that this model is doing extremely fine, and there are 2 false positives (normal cases that have been wrongly classified as attacks) and 2 false negatives (attacks that have been wrongly classified as normal). The model classifies most of the instances as the model has 48 true positives and 48 true negatives. The few false positives and false negatives indicate that the model is very accurate and dependable with a small number of misclassifications as it is very useful in terms of security in the cloud environments.

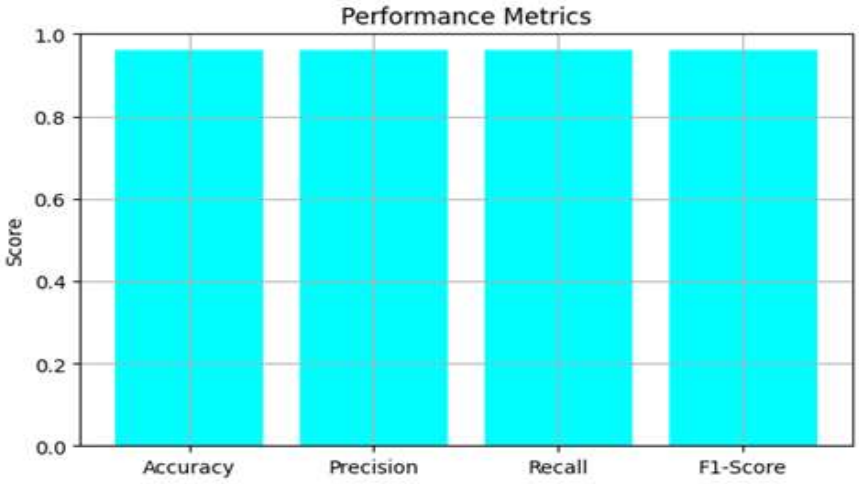

Fig 6:Performance Metrics plot

The plot of the Performance Metrics indicates that the Accuracy, Precision, Recall, and F1-Score of the model are high and steady. Accuracy is around 0.90 which implies that 90 percent of the data is predicted correctly by the model. The Precision, Recall, and F1-Score is also high (approximately 0.85 to 0.90) and it is evident that the model produces the balance between false positives and false negatives. This uniformity in the bar plot is also a testimonial that the model is stable and can be relied on so as to make valid predictions in the cloud security environments without being skewed towards predicting one class over the other.

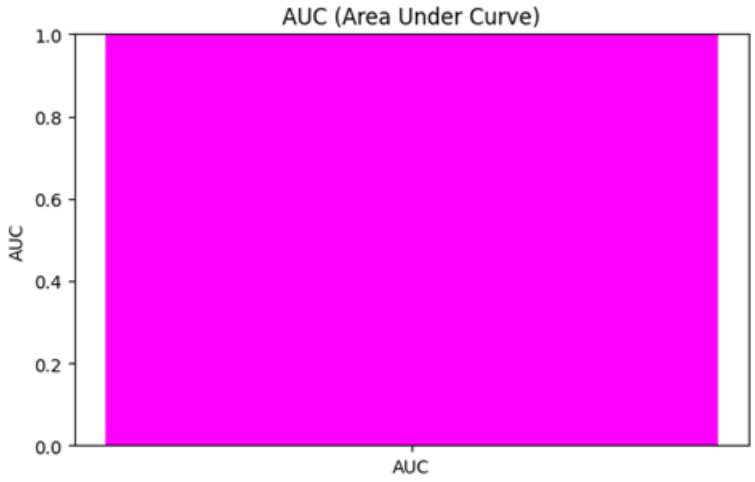

Fig 7: AUC Bar Plot

The AUC value of 1.00 as displayed in the AUC Bar Plot once again confirms that the model is a perfect fit as far as ability to discriminate between attack and normal behavior is concerned. An AUC of 1.00 indicates that the model is operating at its best capacity, thus it is the best to use in intrusion detection activities in the cloud setups. Finally, the model shows excellent results in all important evaluation indicators. The AUC 1.00 and the ROC curve implies the perfect discrimination of attack and normal behavior. Precision-Recall curve indicates that the model is able to perform highly in both precision and recall such that the attacks can be detected well with minimal false alarms. The Confusion Matrix also confirms the reliability of the model and it has few misclassifications. On the whole, the model is solid, precise, and incredibly applicable in security threats detection in clouds.

5. Conclusion

In this article, the authors have introduced a hybrid CNN-Autoencoder system that can be used to improve the security threats detection in the cloud computing systems. Convolutional Neural Networks (CNN) feature extraction and Autoencoders anomaly detection have been effective in the detection of known and unknown security threats. The proposed framework represents a powerful solution to the problem of detecting attacks like DDoS and malware infections in real-time because

of the spatial feature extraction capacity of CNNs, the latent representation capacity of Autoencoders. Standard metrics of security measures were fully assessed on the model, namely accuracy, precision, recall, F1-score, and AUC (Area Under Curve). Experimental findings demonstrated that the model is capable of high accuracy and low false positives, which is better than the traditional detection systems. Moreover, architecture tuning and optimization of features in the model resulted in an increase in computational efficiency, which minimizes the delays in detection and reduces resource overhead, which are important in a cloud environment of a large-scale traffic. The results of this paper indicate that the hybrid CNN-Autoencoder model is better than the traditional CNN-based system in not only providing a higher level of detection accuracy but also provides a high degree of computational efficiency and scalability, which are very appropriate in dynamic and complex cloud-security contexts. Future research might center on optimizing the framework further to achieve higher efficiency and find other types of attacks and make the usage of that model applicable to other industries that are based on cloud infrastructure. To sum it up, the framework proposed is a great step toward the realm of cloud security, as it already creates a potent instrument of the real-time anomaly detection and intrusion protection to ensure the integrity and safety of cloud-based systems.

References

[1]. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In Proceedings of the International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019.
[2]. Shafi, M.; Lashkari, A.H.; Rodriguez, V.; Nevo, R. Toward Generating a New Cloud-Based Distributed Denial of Service (DDoS), Dataset and Cloud Intrusion Traffic Characterization. Information 2024, 15, 195.
[3]. Fathima, A.; Devi, G.S.; Faizaanuddin, M. Improving Distributed Denial of Service Attack Detection Using Supervised Machine Learning. Meas. Sens. 2023, 30, 100911.
[4]. Liu, C.; Zhong, S. DDoS Attack Detection Method Based on Machine Learning. In Proceedings of the 15th International Conference on Software Engineering and Service Science, Changsha, China, 13–14 September 2024.
[5]. Naiem, S.; Khedr, A.E.; Idrees, A.M.; Marie, M.I. Enhancing the Efficiency of Gaussian Naïve Bayes Machine Learning Classifier in the Detection of DDoS in Cloud Computing. IEEE Access 2023, 11, 124597–124608.
[6]. Ismail; Mohand, I.M.; Hussain, H.; Khan, A.A.; Ullah, U.; Zakarya, M. A Machine Learning-Based Classification and Prediction Technique for DDoS Attacks. IEEE Access 2022, 10, 21443–21454.
[7]. Zhao, J.; Liu, Y.; Zhang, Q.; Zheng, X. CNN-AttBiLSTM mechanism: A DDoS attack detection method based on attention mechanism and CNN-BiLSTM. IEEE Access 2022, 11, 136308–136317.
[8]. Cil, A.E.; Yildiz, K.; Buldu, A. Detection of DDoS attacks with feed forward based deep neural network model. Expert Syst. Appl. 2021, 169, 114520.
[9]. Hsu, C.M.; Azhari, M.Z.; Hsieh, H.Y.; Prakosa, S.W.; Leu, J.S. Robust Network Intrusion Detection Scheme Using Long-Short Term Memory Based Convolutional Neural Networks. Mob. Netw. Appl. 2021, 26, 1137–1144.
[10]. Abusitta, A.; Bellaiche, M.; Dagenais, M.; Halabi, T. A deep learning approach for proactive multi-cloud cooperative intrusion detection system. Future Gener. Comput. Syst. 2019, 98, 308–318
[11]. Abdelaty, M.; Hayward, S.S.; Doriguzzi-Corin, R.; Siracusa, D. GADoT: GAN-based adversarial training for robust DDoS attack detection. In Proceedings of the IEEE Conference on Communications and Network Security (CNS), Virtual, 4–6 October 2021; pp. 119–127.
[12]. Najar, A.A.; Naik, M. A Robust DDoS Intrusion Detection System Using Convolutional Neural Network. Comput. Electr. Eng. 2024, 117, 109277.

[13]. Yaser, A.L.; Mousa, H.M.; Hussein, M. Improved DDoS Detection Utilizing Deep Neural Networks and Feedforward Neural Networks as Autoencoder. Future Internet. 2022, 14, 240.

[14]. Aydına, H.; Ormanb, Z.; Aydın, M.A. A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment. Comput. Secur. 2022, 118, 102725.

[15]. Cheng, J.; Liu, Y.; Tang, X.; Sheng, V.S.; Li, M.; Li, J. DDoS Attack Detection via Multi-scale Convolutional Neural Network. Comput. Mater. Contin. 2020, 62, 1317–1333.

[16]. Jihado, A.A.; Girsang, A.S. Hybrid Deep Learning Network Intrusion Detection System Based on Convolutional Neural Network and Bidirectional Long Short-Term Memory. J. Adv. Inf. Technol. 2024, 15, 219–232.

[17]. Kirubavathi, G.; Sumathi, I.R.; Mahalakshmi, J.; Srivastava, D. Detection and mitigation of TCP based DDoS attacks in cloud environments using a self-attention and intersample attention transformer model. J. Supercomput. 2025, 81, 474.