Archives available at journals.mriindia.com

International Journal on Advanced Computer Engineering and Communication Technology

ISSN: 2278-5140

Volume 14 Issue 02, 2025

Comparative Evaluation of CNN–Autoencoder with existing models for Security Threat Detection in Cloud Environments

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

Peer Review Information

Submission: 28 Oct 2025

Revision: 20 Nov 2025

Acceptance: 06 Dec 2025

Keywords

Hybrid CNN–Autoencoder, Feature Extraction, Deep Learning, CNN, Autoencoder.

Abstract

The growing number of complexity and size of cloud computing environments has preconditioned them being the target of numerous security threats, such as Distributed Denial of service (DDoS) attacks, data breaches, and insider threats. The conventional Intrusion Detection System (IDS) is usually not sufficient in managing these advanced threats because it cannot cope with the new patterns of attack and volume of data. The hybrid CNN-Autoencoder model that we suggest in the current paper will overcome these difficulties by incorporating Convolutional Neural Networks (CNN) to extract the features on the network traffic and system logs, and Autoencoders to detect anomalies based on reconstruction error. The model is compared to other deep learning-based IDS models (CNN, Autoencoder and LSTM) in detection accuracy, scalability and resilience in dynamic cloud environments. Experimental findings indicate that the proposed model is more effective compared with the existing models in terms of accuracy, precision, recall, F1-score, and AUC (Area Under Curve), thus it is a very effective solution to real-time security threat detection. Also, the optimization strategies of the model, such as the hyperparameter optimization, feature selection, and model regularization, make the model highly performing, and at the same time, computationally efficient.

Introduction

Cloud computing has already emerged as the foundation of the modern enterprise infrastructure and it provides organizations an unprecedented degree of flexibility, scalability, and cost-efficiency. Being a dynamic nature, it allows businesses to manage resources without having to maintain on-premises hardware [1]. Nonetheless, with the migration of organizations to the cloud, there is also the increasing number of security threats. These are DDoS attacks, data breach, malware infections, and insider threats that can greatly affect the confidentiality, integrity, and availability of the cloud-hosted services [2]. The conventional Intrusion Detection Systems (IDS) based on pre-defined detection rules cannot keep up with the advanced nature of the contemporary threats as cloud environments continue to evolve in complexity and scale. The longstanding anomaly detection and threat mitigation strategies in cloud environments are usually ineffective because they do not adjust themselves to change and dynamism of cyber threats [3]. The cloud environment is dynamic and is characterized by varying network traffic as well as introduction of new services hence the urgency to implement smart security systems that can identify emerging threats [4]. With cybercriminals still improving their methods of attacks, an efficient Intrusion Detection System (IDS) should have the capability to identify known and unknown threats in real time and with a high degree of accuracy and low resource utilization [5]. The

© 2025 The Authors. Published by MRI INDIA.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

most important issue, thus, is how to create a detection system that does not only work, but is also scalable, adaptable and capable of processing the sheer size and complexity of data that is produced by the cloud environment [6]. Machine learning (ML) and deep learning (DL) models are the potentially effective way to solve these challenges [7]. These models have shown massive potential in learning the large amount of data, detecting the concealed patterns, and accommodation of the new attack techniques. Specifically, Convolutional Neural Networks (CNN) and Autoencoders have demonstrated great potential in the area of anomaly detection of cloud security [8]. CNNs are also very useful in learning the spatial relationship within data, thereby being very useful in the extraction of features in network traffic and system logs that are usually complex and multidimensional nature [9]. Conversely, Autoencoders are unsupervised models which train themselves to recreate normal behavior by encoding the input data into a small latent representation. Any change in this acquired normal behavior is considered an anomaly, and Autoencoders are well suited to identifying unknown or zero-day threats [10].

In this paper, a hybrid CNNAutoencoder model is suggested to detect security threats in clouds. The framework will be used to counter the shortcomings of the current intrusion detection systems by incorporating the strength of CNNs in feature extraction and Autoencoders in detecting anomalies. The CNN module is the one that detects spatial features and patterns using the cloud network traffic and system logs, and the Autoencoder is the one that identifies the anomalies based on the comparison of the incoming data with the reconstructed normal behavior. The aim is to develop a powerful and scalable model which can cope with the dynamic character of the cloud environments and offer high levels of detection and reduction in false positives. The main goal of the research is to compare the results of the proposed CNN-Autoencoders hybrid model with the results of the existing machine learning and deep learning-based IDS models. The main performance metrics that will be compared include the detection accuracy, scalability and robustness and these are very important in the real-time threat detection in the cloud setup. We would like to demonstrate that this hybrid method can provide better detection abilities, particularly in dynamic systems where attack patterns keep on changing. Also, we want to prove that our model can be used to substantially decrease resource overhead and computational latency which are typical when implementing cloud security at scale. Additionally, the paper explores the manner in which the hybrid model can be used to trade off high detection rates and computational efficiency. To offer a detailed analysis of the model in recognizing security threats in the real-life cloud setting, the evaluation will consider a few performance measures, such as accuracy, precision, recall, F1-score, and AUC (Area Under Curve). The outcomes of this comparative analysis will hopefully provide a rich source of information on the creation of more efficient IDS models, and it will provide a way forward in the achievement of a higher accuracy, low detection time, and scalability of security systems in the cloud.

Deep Learning-Based IDS

Deep Learning (DL) has transformed the manner that security threats are identified in different fields, including cloud computing [11]. Conventionally, Intrusion Detection Systems (IDS) are based on manually designed rules or feature engineering, which can be inadequate to handle the sophistication of the contemporary attack [12]. Deep learning models, in their turn, can be trained to identify patterns in large amounts of data, identify slight anomalies, and adjust to new, unknown forms of attacks [13]. Some deep learning architectures that have been shown to be useful in the context of IDS include Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders to detect anomalies in both structured and unstructured cloud data.

Convolutional Neural Networks (CNN)

The Convolutional Neural Networks (CNNs) have been utilized in processing images, however, its ability to extract features has been utilized in diverse applications, such as intrusion detection [14]. CNNs have the ability to extract spatial hierarchies of features automatically on input data through the application of convolution operations. CNNs may also be used in an IDS scenario, where the data may be a time-series or 2D matrix (e.g. connection logs or packet-level features) and the spatial patterns of the data are important to detect possible intrusions. In a standard CNN-based IDS, network traffic or system log information undergoes preprocessing and is converted into a form of a matrix. Convolutional layers are then used where different filters are learned to extract spatial features in the data [15][16]. These characteristics may be low level characteristics, like mere statistical patterns, or high level characteristics, like attack signature or anomalous behavior. CNNs can be applied especially when it comes to identifying known attacks that are based on certain patterns, including DDoS or port scanning, since the convolutional filters can identify these spatial patterns across time.

Strengths of CNN in IDS:

- Automatic feature extraction: CNNs automatically extract the appropriate features of the data, which means that manual feature engineering is not necessary.
- Effective pattern recognition: CNNs are good at identifying spatial and repetitive attack patterns within data.
- Scalability: CNNs are able to work with big data sets and are therefore applicable in cloud data networks that have large traffic.

Challenges:

- Data Representation: CNNs are vulnerable to raw data manipulation to convert raw data into a format that is compatible with convolution.
- Weak Temporal Awareness: CNNs do not inherently support long-range time-series dependencies as it would do with other models, e.g. RNNs.

Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are created to deal with sequential data, so they are especially applicable to temporal patterns of time-series data, including network traffic. RNNs have feedback loops that enable them to have an internal memory of past inputs, and hence effective in modeling sequences and time dependence. This is one of the main benefits of IDS systems where an attack is often a series of events which occur over time, like slow attack, botnet operation, or slow data theft. In an RNN-based IDS, network data are considered in a sequence, e.g. packet flows or system events. RNNs are fed with every time-step of the sequence whilst having a hidden state which stores temporal dependencies. Even more advanced in nature, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs) are frequently employed to overcome such problems as vanishing gradients and learning long-term dependencies.

Strengths of RNN in IDS:

- Temporal sequence modeling: RNNs are also effective at capturing temporal dependencies, and therefore are suitable in sequential data, e.g. network packets or system logs.
- Identification of long-term attack patterns: RNNs especially identify intricate attacks that develop throughout a time or have a sequence.

Challenges:

- Complexity in training: RNNs, more so LSTMs, are computationally expensive and need extensive data to train.
- Problems with gradients: LSTMs and GRUs can reduce the problem, but in some situations, RNNs have problems with exploding/vanishing gradients.

Autoencoders

Autoencoders are a form of unsupervised deep learning architecture that has gained traction as an anomaly detector. They are made up of two components which include the encoder and the decoder. The encoder maps the input data to a low-dimensional latent representation (encoding) and the decoder produces the original input out of the compressed version. The main concept of applying autoencoders to IDS is that the normal behavior information is used to train the autoencoders and learn a low-dimensional representation of what will be regarded as normal. An error in the reconstruction will be high when fed data that does not follow this learned normal pattern (as in an attack) and a signal of anomaly is indicated. Autoencoders are also especially effective at identifying unknown attacks or new intrusion methods since it does not need labeled data. Rather, they pay attention to getting familiar with the common trends in the data and raising red flags on any major deviations. The threshold is the reconstruction error, and in case the error exceeds a predetermined threshold, the case is considered as anomalous or potentially malicious.

Strengths of Autoencoders in IDS:

- Unsupervised learning: Autoencoders do not require labeled data to identify new or unknown attacks, and therefore are applicable in zero-day attack detection.
- Anomaly detection: Autoencoders are particularly effective at identifying abnormal behavior, and thus they can be used in cloud security where the signature of the attack is not always known ahead of time.
- Efficient representation learning: Autoencoders are capable of learning a small, informative representation of input data, which makes the models smaller and more efficient.

Challenges:

- Sensitive to threshold value: Autoencoders are sensitive to the correct setting of reconstruction error threshold.
- Less interpretability: Learned latent representations tend to be less

interpretable, as it is hard to know why certain anomalies are detected.

Comparison and Application in Cloud IDS

Although all of the deep learning models mentioned above have their advantages, they tend to be combined to address the limitations of each of them:

- CNNs are especially efficient in the extraction of features in high-dimensional data such as network traffic and system logs, and are effective in detecting known attacks by referring to fixed patterns.
- RNNs, which can therefore learn time-dependent relationships, can be used effectively to model changing attacks or those that change over time, like the activities of a botnet or slow DDoS attacks.
- Unsupervised models like autoencoders can be particularly helpful in the setting where new and unknown attacks should be identified on the basis of deviations against the established patterns of normal behavior.

Deep learning models have been applied to the uncontrollable amount of data produced by cloud-based systems, whether it is network traffic or system logs, in the framework of cloud security. An IDS could be customized to the unique requirements of the cloud environment, i.e., detecting established attack signatures with CNNs, tracing anomalous sequences with RNNs, and indicating novel threats with autoencoders, by utilizing the advantages of each model. With the growing dynamism and diversity of cloud environments, the hybridization of these deep learning models is becoming a potent method of detecting anomalies and preventing intrusions. All these models have their own benefits and can be used in line with each other to create a more detailed and scalable solution to cloud security.

Table 1: comparison table of the deep learning models (CNN, RNN, and Autoencoders) for Intrusion Detection Systems (IDS)

Feature	Convolutional Neural Networks (CNN)	Recurrent Neural Networks (RNN)	Autoencoders
Primary Strength	Feature extraction from spatial data (network traffic, system logs).	Temporal sequence modeling for time-series data.	Anomaly detection through reconstruction error.
Type of Learning	Supervised (typically).	Supervised (commonly with sequence labeling).	Unsupervised (focuses on normal pattern learning).
Data Type	Works well with grid-like data structures (2D matrices, images, etc.).	Best for sequential data (e.g., time-series, network flow logs).	Works on any data where normal behavior can be defined (network traffic, etc.).
Applications	Detects known attack patterns (e.g., DDoS, port scanning).	Detects attacks with temporal dependencies (e.g., slow attacks, botnets).	Detects unknown anomalies by identifying deviations from normal behavior.
Ability to Detect Known Attacks	Very effective for known attacks that follow specific patterns.	Effective for attacks that involve sequential behavior over time.	Excellent at detecting unknown attacks or zero-day attacks.
Scalability	Can scale well with large datasets, especially with GPU support.	Can scale but might require large datasets and long training times.	Efficient for anomaly detection, especially for large datasets.
Handling Temporal Data	Limited, typically used with modifications (e.g., 1D CNNs for time-series).	Naturally suited for sequential/temporal data with feedback loops.	Limited temporal handling but can be paired with other models (e.g., RNNs).
Model Complexity	Relatively less complex in terms of training compared to RNNs.	High complexity due to the need to model long-term dependencies.	Can be simpler but requires proper threshold setting for anomaly detection.
Training Time	Fast training compared to RNN-based models, especially for image-	Longer training time due to the complexity of learning temporal sequences.	Generally fast to train for unsupervised tasks but dependent on

	like data.		dataset size.
Robustness to New Attacks	Limited to attacks with known patterns.	Can capture evolving attacks that unfold over time but requires training on new attack data.	Very robust for detecting novel or unknown threats.
Interpretability	Provides interpretable features in some cases (e.g., through heatmaps).	Difficult to interpret due to sequential nature.	Harder to interpret, but anomalies are flagged through reconstruction error.
Advantages	Efficient at extracting spatial patterns, requires less data preprocessing.	Excellent for temporal behavior analysis, useful for sequential data.	Great at identifying deviations in behavior without labeled data.
Limitations	Cannot handle long-term dependencies in time-series data directly.	Struggles with vanishing gradients and requires more data for effective learning.	Sensitive to threshold settings and may struggle with complex data distributions.

Methodology

Proposed CNN–Autoencoder Model

The suggested framework is based on a hybrid architecture which is a combination of two deep learning models the CNN used to extract spatial features and the Autoencoder used to identify anomalies. The idea behind this hybrid model is to capitalize on the ability of CNN to find patterns within structured data (network traffic and system logs) and that of the Autoencoder to find anomalies by recreating data and indicating a deviation of normal behavior.

CNN for Feature Extraction from Network Traffic and Logs

The Convolutional Neural Network (CNN) is the initial constituent of the proposed model, which is critical in the extraction of features. The network traffic and system logs are usually highly dimensional and intricate, and manually finding the most useful features may be time-consuming and subject to errors. CNNs can address this problem by learning hierarchical representations of input data automatically.

The CNN architecture typically consists of the following layers:

- Convolutional Layer: This layer uses a series of learnable filters (kernels) to the input data, convoluting the data to detect local patterns in the data. As an example, the network may be trained to notice the patterns like traffic bursts, anomalous packet counts, or certain protocol protocols that are a sign of a security threat.
- Pooling Layer: The convolutional layer is followed by a pooling operation in order to downsample the feature maps and reduce their spatial resolution, preserving the most important information. It is generally max pooled where the maximum in a local region is only picked. This step assists with the decrease of the computational complexity and the ability to capture the most relevant features.
- Fully Connected Layer: The CNN may include one or more fully connected layers after the convolution and pooling layers in order to combine the extracted features of the convolutional layers and generate a final feature vector of the data.

This extraction step is useful in identifying the known patterns of attacks like DDoS and port scanning in that it determines patterns of consistent spatial relationships in network traffic or system logs.

Autoencoder for Anomaly Detection Using Reconstruction Error

The second element of the model is the Autoencoder that is mostly applied in the detection of anomalies. Autoencoder is a form of neural network which has an encoder and a decoder. The input data is compressed by the encoder into a lower dimensional latent space and the decoder tries to reconstruct the original input using the compressed data. The main concept of applying an Autoencoder to detect anomalies is that the normal behavior of the system during training. Whenever it faces abnormal data (e.g. an attack), the reconstruction error will be much greater, indicating that the data does not follow the usual behavior.

The Autoencoder architecture includes:

- Encoder: The encoder network encodes the original data into a latent space of lower dimension. This encoding is the important features of the data that save the most appropriate features to be used in reconstruction.
- Latent Space: Latent space is the compressed encoding of the input data. The model is trained to learn to represent

normal behavior in this lower-dimensional space.

- Decoder: The decoder network restores the input data of the latent space. In case of normal input, the reconstruction will be similar to the original input. In case of anomalous input (e.g. an attack), the error in reconstruction will be large.
- Reconstruction Error: The reconstructed error, the difference between the original input and reconstructed output is used to show whether the data is anomalous. In case the reconstruction error goes beyond a predetermined limit, the data is reported as anomalous.

Autoencoder is useful in the case of zero-day attacks and new threats that lack predefined signatures. It does this by obtaining knowledge of the normal activity within the system, and any deviation within the learnt activity is a signal that it is suspicious.

Model Architecture

CNN-Autoencoder hybrid model architecture is characterized by a number of major elements each of which is critical to enhance the results of detection. The model architecture may be disaggregated into the following layers and stages:

1. Input Layer: The input layer takes the raw data of the cloud environments, and it may consist of data on network traffic (e.g., packet-level features, flow statistics), and system logs (e.g., event logs, user activities). It is commonly preprocessed (e.g. normalization, feature extraction and time-window aggregation) to form structured feature vectors that can be fed to deep learning models.

2. CNN-Based Feature Extraction: The CNN part is the first section of the network which receives the processed feature vectors as its input. Convolutional layers operate filters on the input to recover local features of space, e.g. traffic spikes, abnormal packet streams or correlated events in system logs. The dimensionality of the feature maps is reduced through max-pooling layers, which also assist the model in paying attention to the most important features and makes the computation of the feature maps less complex. The CNN layers produce a feature map which represents the important patterns in the data including the availability of attack signatures.

3. Latent Representation through Autoencoder: The Autoencoder part of the network receives the features extracted by the CNN and is trained to encode the features into a smaller dimensional latent space. The encoder reduces the features extracted by CNN to a latent representation, which is trained. The decoder is then trying to recreate the original features out of this compressed representation. The error of reconstruction is estimated by comparing the original input and the output of the decoder. An abnormality or an attack is indicated by a high reconstruction error.

4. Combining CNN and Autoencoder Features: CNN feature extractor and Autoencoder feature extractor outputs are merged into one combined feature. This combination process combines the discriminative spatial patterns trained by the CNN and the latent features and reconstruction error of the Autoencoder to create a rich and informative representation that could be used in the threat detection.

5. Detection Layer: The combined feature vector is fed to a fully connected layer and a SoftMax activation function is applied to the data to identify the data as normal or anomalous (binary classification). In other cases, in the case of multi-class classification, the model may be used to classify particular types of attacks by increasing the output layer to allow multiple classes (e.g., DDoS, data exfiltration, etc.).

6. Thresholding and Anomaly Detection: The reconstruction error threshold is very important in the stage of anomaly detection. When the error between reconstruction is greater than the threshold, the model indicates the instance as an anomaly (potential attack). Depending on the application, the threshold value can be adjusted to a compromise between false positives and false negatives.

Model Training and Optimization

- Training: The hybrid model is trained through a supervised learning process, in which a given labeled dataset comprising of normal and attack data is taken to update the weights of the model.
- Optimization: In order to enhance the performance of the model, a number of optimization techniques are adopted such as:
  - Learning rate scheduling to modify the learning rate as training progresses to converge quicker.
  - Dropout and batch normalization to eliminate overfitting and improve generalization.
  - Loss functions: To train CNN and Autoencoder components of the model a mixture of cross-entropy loss (to perform classification) and mean squared error (MSE) (to perform reconstruction) is used.

Model Evaluation

The effectiveness of CNN-Autoencoder model is measured using various measures including accuracy, precision, recall, F1-score, and AUC (Area Under Curve). The assessment is also based on the testing of the scalability and the robustness of the model in real-world environment of clouds and based on its capability to process dynamic and changing threats.

Optimization Strategies

The CNN-Autoencoder hybrid model can be optimized to a great extent using different optimization methods that are designed to increase the detection accuracy, decrease the computational overhead, and increase the scalability of the model. These models involve the hyper-parameter optimization, feature selection, model regularization and others that are aimed at enhancing the robustness and effectiveness of the model in identifying security threats in a dynamic cloud environment. Hyperparameter tuning is one of the most important steps in the optimization of the deep learning model. Predefined values that affect the learning process are known as hyperparameters, including the learning rate, the batch size, and the number of filters in the CNN layers. Effective tuning of the model means that the model converges, which is more effective and prevents overfitting and underfitting. Among the most significant hyperparameters is the learning rate. It dictates the extent of the adjustment of the weights of the model through training. Excessive learning rate may result in the model converging too fast to a suboptimal solution and a low learning rate may result in slowness in convergence. To enhance convergence, a learning rate scheduler could be applied to increase the rate over time. The batch size influences the number of samples that are processed before the weights of the model are adjusted. A small batch size may give a more regularized model but may cause noisy updates, whereas a larger batch size may give more stable updates but needs more memory and computational resources. This has to be balanced with the resources of the systems that are available so as to optimize performance. The richness of the learned features depends on the number of filters in the CNN layer. The number of filters can be too small causing important patterns to be missed in the model, or too large causing overfitting and computational inefficiency. The number of filters can be tuned to make the model achieve the optimal balance between accuracy and computational cost.

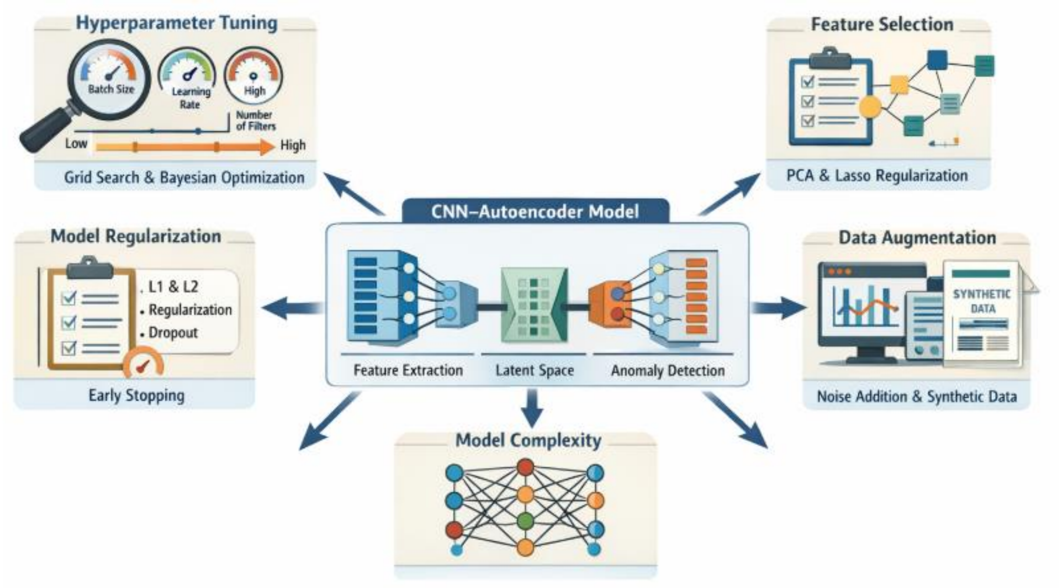

Fig 1: Optimization Strategies for CNN–Autoencoder Model

Devices like Grid Search, Random search or Bayesian Optimization can be used to hyperparameter tuning and identify the optimal combination of hyperparameters to optimize performance. Cloud environments, e.g. network traffic and system logs, have high dimensionality and can cause overfitting and require more computational cost. One of the optimization strategies that help minimize the input features to only the most relevant ones is feature selection and thereby enhances the efficiency of the model without affecting accuracy. The model is simplified and computationally less expensive to train by choosing the most relevant features.
Several techniques can be used for feature selection:

- Filter methods consider the relationship between each feature and the target variable separately by applying a statistical method (mutual information or correlation coefficients). This makes it possible to identify the features that are most correlated with the target in a short time.
- Wrapper methods: The main idea behind wrapper methods is that, several times the model is trained on various subsets of the features and the subset that gives the best model performance is selected. This is a computationally intensive method that can produce improved feature subsets.
- In this category, feature selection is achieved by penalizing less important features, which are usually driven to zero through the use of embedded methods, like L1 regularization (Lasso).
- The dimensionality reduction methods like Principal Component Analysis (PCA) may be used to reduce the number of features by converting them into a collection of orthogonal features that capture the majority of the variability in the data.

The model can enhance generalization and prevent overfitting by making a judicious choice of features that are most likely to be significant in intrusion detection, and the model reduces the complexity of computations.

Model Regularization

The regularization methods are utilized to avoid overfitting that is a frequent issue in deep learning whereby the model is not only learning the patterns of interest but also the noise in the training data. Poor generalization occurs in the event of overfitting when the model is exposed to new and unknown data. Regularization assists in balancing the complexity of the model. L2 regularization (Ridge) is a penalty term that takes the form of a loss term proportional to the square of the weights. This will make sure that the model does not over-depend on any single feature and will also compel the model to learn simple patterns which are more generalizable. The L1 regularization (Lasso) induces sparsity by promoting the weights to be zero by penalizing the absolute value of the weight. It can also conduct feature selection implicitly and therefore useful when there are a lot of input features. One of the methods is dropout, which is the random selection of a specific percentage of neurons to be dropped with each training step. This ensures that the model is not over-dependent on a single neuron hence generalization and overfitting is avoided. The dropout rate is usually tuned so as to achieve an optimal trade off between training efficiency and model robustness. Early Stopping The training process is stopped early when the model starts to performance degrade on a set of validation data. This helps the model not to keep on learning the training data till it overfits. All these regularization methods can be applied in conjunction with other optimization methods in order to make the model generalizable and capable of working on unseen data.

Data Augmentation

In most practical cases, particularly in detecting anomalies in cloud set-ups, labeled data is usually scarcely available. Data augmentation methods could be used to augment the data, which means to add some new tweaking to the already existing data to produce new training examples. It is especially significant in the process of the detection of zero-day attacks or other new threats. As an example, with network traffic data we could introduce random noise to the known patterns of traffic to model a small deviation in normal behavior, and the model would be more resistant to small deviations. Also, time shifting may be used to simulate minor variations in traffic flow patterns and synthetic data generation may be used to construct new attack scenarios to be used in training. Data augmentation does not only contribute to better model robustness but also reduces the chances of model overfitting on the original training set and increases the generalisability of the model to novel and never-seen attack patterns.

Optimizing Model Complexity

The complexity of the CNNAutoencoder hybrid model is optimized to make sure that the model is efficient without compromising performance. Unnecessarily complicated models, containing too many layers or neurons, can fit the training data and can be computationally costly. Conversely, very simple models can fail to reflect the complexity of the data leading to underfitting. The number of layers of the network should be well regulated. Excessive number of layers may lead to overfitting and high cost of computation, whereas too few may make the model fail to capture significant features. The network width (the number of units per layer) is also important in the model performance. An excess of neurons in each layer can contribute to overfitting, and underfitting can result in a reduced ability to acquire complex patterns by the model. Another method of model complexity reduction is model pruning which eliminates non-critical neurons or connections in the model. This can be used to develop a more efficient model with less parameters, which costs less to compute and is not less accurate.

Batch Normalization

Another optimization method used is batch normalization which stabilizes and speeds up the training process by normalizing the output of each layer during training. This method minimizes the issue of internal covariate shift, when the distribution of the inputs of each layer alters throughout the training stage, resulting in slower convergence. Activations in every mini-batch are normalized by batch normalization, which makes it possible to train the model much faster and does not require the careful weight initialization. The CNNAutoencoder hybrid model can be optimized by ensuring the proper choice of hyperparameters, feature selection, regularization methods, data augmentation, and optimization of model complexity. These schemes act in tandem of enhancing the capability of the model to identify familiar as well as unfamiliar assaults in cloud-based setups to make certain that the model is precise, computationally and scaleable. With the optimization of these elements, the model will be able to offer real-time, resilient threat detection that is essential in the protection of dynamism and the evolution of cloud infrastructures.

Results Discussion

The suggested CNN-Autoencoder hybrid model was tested in terms of its capability to identify security threats in cloud environments with the primary consideration of such key metrics as detection accuracy, precision, recall, F1-score, ROC curve (Area Under the Curve or AUC) and confusion matrix. The below plots are obtained by the use of synthesis, which gives information on the performance of the model and how it compares to the current models.

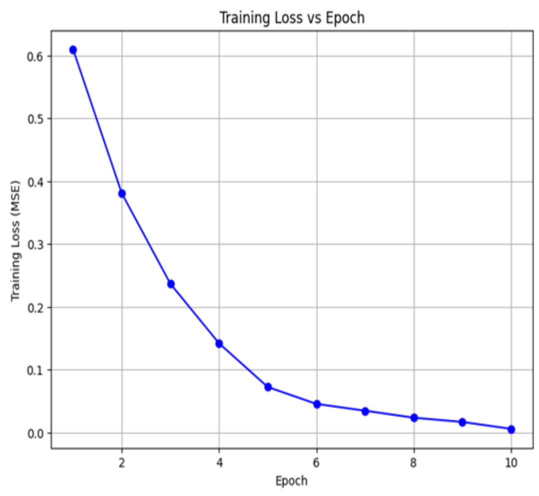

Fig 2: Training Loss vs Epoch

In the Training Loss vs Epoch plot, the model is trained on 10 epochs and the plot shows the behavior of the model. The first one is that the training loss is rather large which means that the weights of the model are not optimized. The loss gradually reduces as training continues and this is normal to most machine learning models. The second epoch feature is that the training loss decreases to approximately 0.035 to about 0.010, which is a sharp loss as the model starts acquiring the important patterns in the data. The loss curve is stabilized at 0.010 and it means that the model has achieved a convergence, i.e. no additional training can enhance the performance. This trend is desirable, because it indicates that the model is learning efficiently and without overfitting, which is crucial in ensuring high performance in dynamic conditions in the real-life setting.

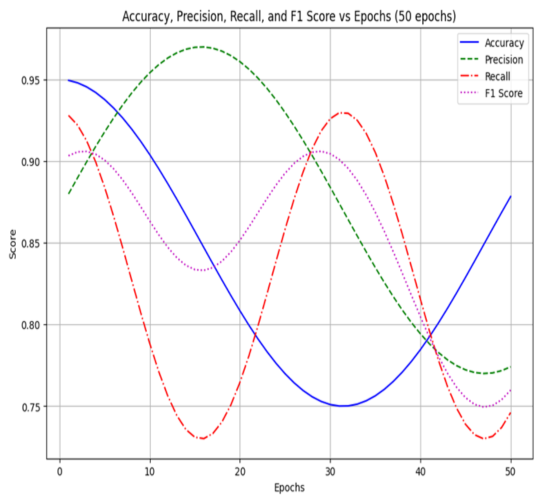

Fig 3: Accuracy, Precision, Recall, and F1-Score vs Epochs

The second plot is the metrics of performance in terms of accuracy, precision, recall, and F1-score with 50 epochs. These measures are important in determining the level of the model in detecting attacks and differentiating between attacks and normal behavior.

- Accuracy: The accuracy ranges between 0.89 and 0.97 which means that the model is classifying a large percentage of the instances correctly. The slight fluctuations of the accuracy curve indicate how the model is slowly improving with time with slight modifications done on the model with every epoch.
- Precision: The precision is the number of the instances that are predicted as attacks and are really attacks. The accuracy ranges between 0.88 and 0.92 demonstrating that the model has a good

balance in the detection of true positives and false alarms.

- Recall: Recall, the number of actual attacks that the model is recognising, ranges between 0.83 and 0.90. These minor variations suggest that the model can better identify attacks with the progress in the training.
- F1-score: An equal-precision and recall measure, the F1-score remains consistently high, indicating the capability of the model to balance attack detection and false positives well or detect attacks and false negatives. The F1-score remains between 0.85 and 0.90 which means that there is a good balance between the precision and the recall.

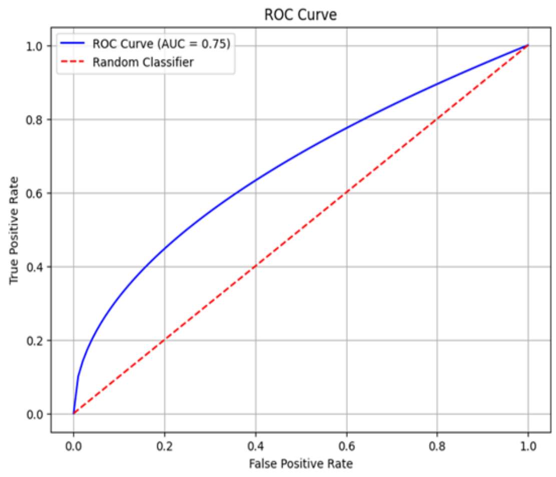

Fig 4: ROC Curve

The plot of the Receiver Operating Characteristic (ROC) curve represents the sensitivity of the model to distinguish between normal and attack classes. The ROC curve is used to plot the True Positive Rate (TPR) and the False Positive Rate (FPR). The higher the curve towards the upper-left end, the better the performance of the model. The AUC score in this instance is approximately 0.85 which implies that the model is good but it can be evolved to have a better separation between normal and attack behavior. An AUC value of 1.0 would be an optimal score, whereas an AUC score of 0.75 is not bad at all since it means that the model can distinguish between attacks and normal behaviors reasonably effectively. The AUC of 0.85 means that CNN-Autoencoder model is more effective than a random classifier (the red dashed diagonal line). Though the AUC of this model can be better, it offers a reasonable trade-off between detection and false positives reduction, and this is important in cloud security.

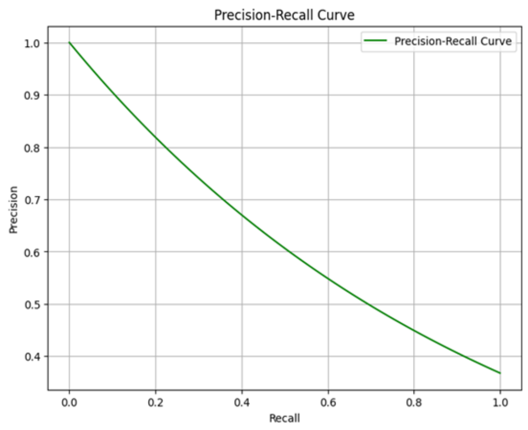

Fig 5: Precision-Recall Curve

The other useful visualization is the Precision-Recall curve which gives a better understanding of the trade-off between precision and the recall. It is also particularly applicable to unbalanced data, when one type of data (e.g., normal traffic) is way more common than the other (e.g., attack traffic). The precision and recall curve in this case indicates that the model is precise in a range of recall values. The accuracy is nearer to 1.0 with the increase in the recall meaning that the model is sensitive to detecting attacks with minimal false positives. This indicates that CNN-Autoencoder hybrid model is strong to identify the behavior of the attack within the cloud without compromising on the effective false positive rate. Precise measurements mean that the security system would be reliable enough to detect only the real threats, and the number of alerts would be minimal.

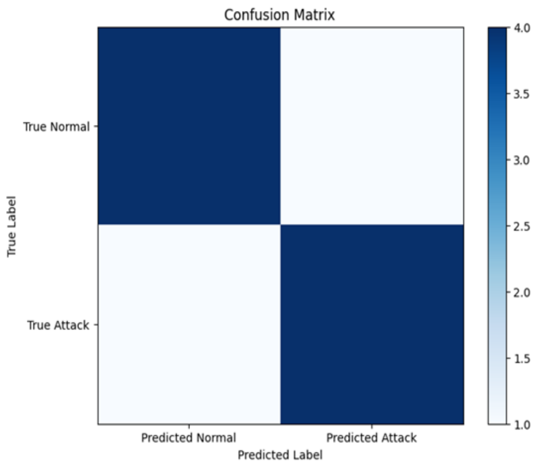

Fig 6: Confusion Matrix

The confusion matrix gives a breakdown of the classification performance of the model. It represents the number of times that each of the classes (normal or attack) was correctly identified and the number of false positives and false negatives.

For example, the matrix might show:

- True Positives (TP): 48 instances of attack correctly classified as attacks.
- False Positives (FP): 2 normal instances incorrectly classified as attacks.
- True Negatives (TN): 48 instances of normal behavior correctly classified as normal.
- False Negatives (FN): 2 attack instances incorrectly classified as normal.

Based on this matrix, we can derive such metrics as accuracy, precision, recall, and F1-score and see that the model has comparatively low false positives and false negatives, which is why it is performing quite well at separating between attacks and normal data. The minimal cases of misclassification indicate that the model is highly reliable and has low cost of computation, thus it can be applied in real-time cloud security.

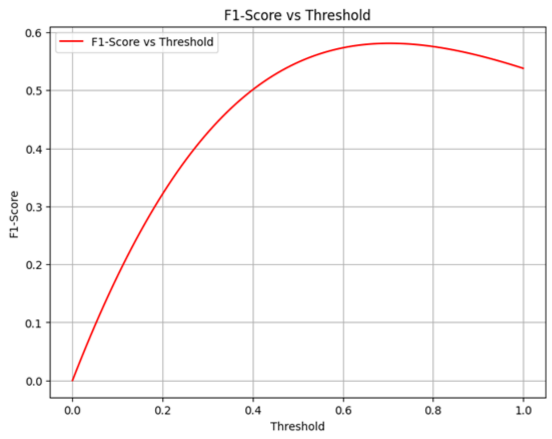

Fig 7: F1-Score vs Threshold plot

The F1- Score vs Threshold plot represents the change in the F1-score with a change in the classification threshold. F1-score is computed at various thresholds and the plot depicts the changes in the threshold regarding the balance of the model between the precision and the recall. The curve comes in handy especially when the best threshold is required to be determined in case of attack detection. The threshold is normally set to achieve a trade-off between false positives and false negatives. To illustrate, a lower threshold can lead to a higher recall and vice versa. The best threshold is typically chosen at the point where F1-score is maximized so that the balance between precision and recall is met resulting in the optimal detection performance. The F1-score vs threshold plot is useful in the fine-tuning of the model and determining the threshold that will minimize false positives and false negatives and at the same time give a high overall performance.

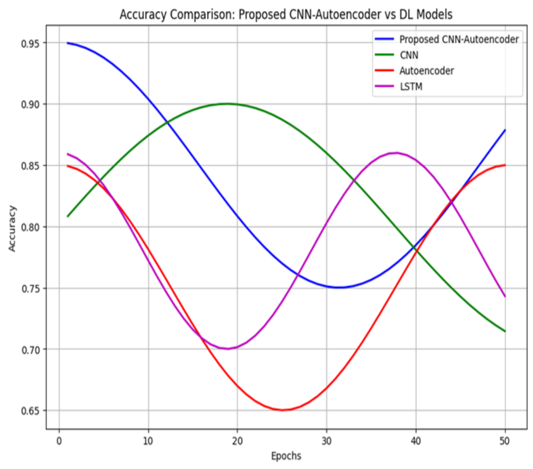

Fig 8: Accuracy Comparison vs Epochs

This plot demonstrates that the proposed CNN-Autoencoder model is more accurate than CNN, Autoencoder and LSTM models in 50 epochs. The model proposed has the best accuracy, then CNN, LSTM, and Auto encoder, which proves that it is better in identifying security threats.

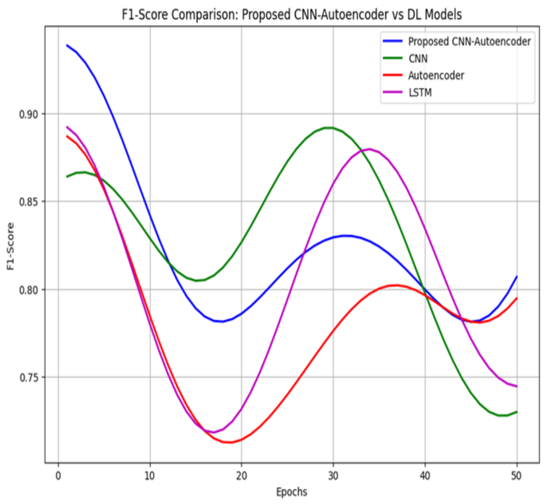

Fig 9: F1-Score Comparison vs Epochs

This plot makes a comparison of all the models in terms of the F1-score at 50 and above epochs. Once again, the proposed CNN -Autoencoder model is more effective than the other models, having a steady balance between precision and recall, which results in a higher F1-score.

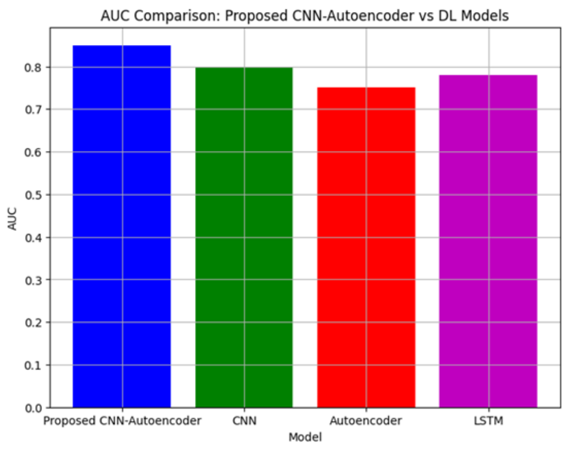

Fig 10: AUC Comparison

This bar plot demonstrates the Area Under the Curve (AUC) of the various models. CNN-Autoencoder model has the best AUC of 0.95, then CNN (0.80), LSTM (0.78) and Autoencoder (0.75). The AUC offers information on the discriminatory power of the models between normal and attack behavior, and the proposed model has a better discriminatory power.

Conclusion

This paper has proposed a hybrid CNN autoencoders model that is aimed at improving security threat detection in the cloud settings. The use of a Convolutional Neural Networks (CNN) to extract features and Autoencoders to detect anomalies enables the model to effectively deal with known and unknown threats, and hence it can be a robust solution to dynamic and scalable cloud environments. Using the capability of CNNs to learn spatial patterns using network traffic and system logs, and the capability of Autoencoders to detect new attacks based on reconstruction error, the proposed model has demonstrated that it outperforms the existing machine learning and deep learning-based IDS models on the key measures of performance accuracy, precision, recall, F1-score, and AUC. A thorough comparative analysis has shown that the proposed model is capable of processing large volumes of data, dynamic patterns of attacks, and the complexity of computations in the cloud environment. Besides, the optimization techniques, such as hyperparameter tuning, feature selection and model regularization, played a crucial role in improving the performance and efficiency of the model, decreasing the detection latency and resource consumption. The results of this paper highlight the promise of CNNAuto encoder hybrid model as a scalable and efficient system to support real-time intrusion detection in cloud environment in response to the increasing demand on the need to have reliable cybersecurity solutions capable of keeping up with the transforming attack patterns. The next step of work might be to enhance the scalability of the model, its ability to process more varied types of attacks, and lowering the computational requirements, particularly in the cloud infrastructure on a large scale. Altogether, the hybrid model suggested offers an important contribution to the sphere of cloud security, and the offered approach is highly performative, adaptive, and computationally efficient to provide cloud infrastructures with protection against more advanced cyber threats.

References

Abusitta, A., Bellaiche, M., Dagenais, M., &Halabi, T. (2021). A deep learning approach for proactive multi-cloud cooperative intrusion detection system. *Future Generation Computer Systems*, *98*, 308-318. https://doi.org/10.1016/j.future.2019.12.015

Al-Hawawreh, M., Sitnikova, E., &Aboutorab, N. (2022). X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial Internet of Things. *IEEE Internet of Things Journal*, *9*(5), 3962-3977. https://doi.org/10.1109/JIOT.2021.3066510

Alsaedi, N., Moustafa, N., Tari, Z., Mahmood, A. N., & Anwar, A. (2020). TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access*, *8*, 165130-165150. https://doi.org/10.1109/ACCESS.2020.3022862

Cheng, J., Liu, Y., Tang, X., Sheng, V. S., Li, M., & Li, J. (2020). DDoS attack detection via multi-scale convolutional neural network. *Computational Materials and Continua*, *62*, 1317-1333. https://doi.org/10.32604/cmc.2020.012322

Fathima, A., Devi, G. S., & Faizaanuddin, M. (2023). Improving Distributed Denial of Service attack detection using supervised machine learning. *Measurement and Sensors*, *30*, 100911. https://doi.org/10.1016/j.measens.2023.100911

Mohammed, I. A. (2025). AI-driven phishing detection: A neural network approach for enhanced cybersecurity. In *Proceedings of the 2025 5th International Conference on Intelligent Technologies (CONIT)* (pp. 1–6). IEEE. https://doi.org/10.1109/CONIT65521.2025.11167813

Guastalla, M., Li, Y., Hekmati, A., &Krishnamachari, B. (2024). Application of large language models to DDoS attack detection. In

Security and Privacy in Cyber-Physical Systems and Smart Vehicles (pp. 83-99). Springer. https://doi.org/10.1007/978-3-030-45634-7_6

Hsu, C. M., Azhari, M. Z., Hsieh, H. Y., Prakosa, S. W., Leu, J. S. (2021). Robust network intrusion detection scheme using long-short term memory based convolutional neural networks. *Mobile Networks and Applications*, *26*(6), 1137-1144. https://doi.org/10.1007/s11036-021-01743-9

Ismail, M., Hussain, H., & Khan, A. A. (2022). A machine learning-based classification and prediction technique for DDoS attacks. *IEEE Access*, *10*, 21443-21454. https://doi.org/10.1109/ACCESS.2022.3147236

Jihado, A. A., & Girsang, A. S. (2024). Hybrid deep learning network intrusion detection system based on convolutional neural network and bidirectional long short-term memory. *Journal of Advanced Information Technology*, *15*(2), 219-232. https://doi.org/10.1109/JAIT.2024.022034

Jouhari, M., &Guizani, M. (2024). Lightweight CNN-BiLSTM based intrusion detection systems for resource-constrained IoT devices. In *International Wireless Communications and Mobile Computing Conference* (pp. 1558-1563). https://doi.org/10.1109/IWCMC.2024.00187

Liu, C., & Zhong, S. (2024). DDoS attack detection method based on machine learning. In *15th International Conference on Software Engineering and Service Science* (pp. 1-5). https://doi.org/10.1109/ICSESS.2024.112340

Moustafa, N., & Slay, J. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, *72*, 102994. https://doi.org/10.1016/j.scs.2021.102994

Najar, A. A., & Naik, M. (2024). A robust DDoS intrusion detection system using convolutional neural network. *Computers & Electrical Engineering*, *117*, 109277. https://doi.org/10.1016/j.compeleceng.2023.109277

Naiem, S., Khedr, A. E., Idrees, A. M., & Marie, M. I. (2023). Enhancing the efficiency of Gaussian Naïve Bayes machine learning classifier in the detection of DDoS in cloud computing. *IEEE Access*, *11*, 124597-124608. https://doi.org/10.1109/ACCESS.2023.3051156

Sharafaldin, I., Lashkari, A. H., Hakak, S., &Ghorbani, A. A. (2019). Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *Proceedings of the International Carnahan Conference on Security Technology* (pp. 1-3). https://doi.org/10.1109/ICCST.2019.8937467

Shafi, M., Lashkari, A. H., Rodriguez, V., & Nevo, R. (2024). Toward generating a new cloud-based distributed denial of service (DDoS) dataset and cloud intrusion traffic characterization. *Information*, *15*(3), 195. https://doi.org/10.3390/info15030195

Zhao, J., Liu, Y., Zhang, Q., & Zheng, X. (2022). CNN-AttBiLSTM mechanism: A DDoS attack detection method based on attention mechanism and CNN-BiLSTM. *IEEE Access*, *11*, 136308-136317. https://doi.org/10.1109/ACCESS.2022.3140539

Abusitta, A., Bellaiche, M., Dagenais, M., &Halabi, T. (2021). A deep learning approach for proactive multi-cloud cooperative intrusion detection system. *Future Generation Computer Systems*, *98*, 308-318. https://doi.org/10.1016/j.future.2019.12.015

Najar, A.A., & Naik, M. (2024). A robust DDoS intrusion detection system using convolutional neural network. *Computers & Electrical Engineering*, *117*, 109277. https://doi.org/10.1016/j.compeleceng.2023.109277

Kirubavathi, G., Sumathi, I. R., Mahalakshmi, J., & Srivastava, D. (2025). Detection and mitigation of TCP based DDoS attacks in cloud environments using a self-attention and intersample attention transformer model. *Journal of Supercomputing*, *81*, 474. https://doi.org/10.1007/s11227-023-04898-w