Industrial Engineering Journal
ISSN: 0970-2555
Volume : 55, Issue 01 : 2026

Evaluating the Performance of a Hybrid CNN–Autoencoder Model 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

Abstract:

Fast uptake of cloud computing has led to flexibility and scalability in the current businesses, however, it has also posed a great challenge in terms of security. Conventional intrusion detection systems (IDS), in many cases, are incapable of being dynamic and keeping up with the changing cloud environment. To improve the known threat detection and the detection of anomalies, the paper will propose a Hybrid CNN-Autoencoder Model of security threat detection in cloud infrastructures by synthesizing the power of Convolutional Neural Networks (CNNs) and Autoencoders. The CNN module obtains spatial features of network traffic and system logs, and the Autoencoder is trained on a small latent representation and detects deviations when the reconstruction error is large, detecting new attacks that have never been seen before. This compound architecture builds on the strengths of discriminative and generative learning and enhances the accuracy of detection and offers a solid solution to cloud security. The performance of the model is measured based on conventional security measures, such as accuracy, precision, recall, F1-score, and AUC-ROC. The results of the evaluation show that the hybrid model is effective in cloud-based IDS and it beats the traditional methods in identifying known and new security threats. It is a major breakthrough in the area of cloud security because it offers a scalable and flexible system of real-time anomaly detection in complex cloud environments.

Keywords: Cloud Security, Intrusion Detection Systems (IDS), Hybrid CNN–Autoencoder Model, Anomaly Detection, Convolutional Neural Networks (CNN), Autoencoders.

1. Introduction

The adoption of cloud computing has been fast and the use has transformed the manner in which businesses and organizations control their IT resources [1]. Cloud computing has transformed to be the foundation of the contemporary enterprise infrastructure by providing scalable, flexible, and economical services. But as the use of the cloud has grown so has the number of cyber threats that have been directed at the cloud environment [2]. The complexity and decentralization of infrastructures in the cloud, as well as the multi-tenant design and the ability to scale resources, present specific security risks, which are not common to more traditional on-premise environments. Intrusion Detection Systems (IDS) have been a major provision in network security since time immemorial, it is set up to monitor and track unwanted access or uncharacteristic conduct in a network or a system [3] . Conventional methods of IDS, including signature-based detection, have been demonstrated to be effective in detecting known threats through the comparison of traffic patterns with pre-defined attack signatures. Nevertheless, they are not enough in the management of the zero-day attacks or emerging threats that lack the signature. Also, they may find it difficult to cope with the

variability and dynamism of cloud environments where traffic may change significantly depending on user demand, virtual machine (VM) migrations and scaling processes[4]. In order to overcome those limitations, more contemporary models of IDS are implementing machine learning (ML) and deep learning (DL) models, which can learn more complicated patterns and adapt to emerging, unknown threats. These models have the possibility of enhancing the detection accuracy as they can be used to detect a system and network behavior anomaly regardless of the precise nature of the attack [5]. In particular, Convolutional Neural Networks (CNN) and Autoencoders are deep learning models that have demonstrated potential in the process of anomaly detection. CNNs are good at spatial and local feature extraction of structured data, whereas Autoencoders are especially good at learning compact latent representations of normal data and detecting deviations (i.e. anomalies) as measured by reconstruction errors [6]. Although CNNs and Autoencoders have demonstrated usefulness separately in distinct fields, the hybridization of the two deep learning tools should be considered as a promising solution to the improvement of anomaly detection in clouds [7]. The CNNs may learn spatial information about cloud traffic (e.g., packet sizes, durations, flags), and system logs (e.g., event types, frequencies, timestamps), whereas the Autoencoders may learn a more compact latent code of the data, and point out the abnormal patterns by the reconstruction error. This combination strategy can address the weaknesses of the traditional models as it can offer both discriminative and generative learning within one framework [8]. This paper presents a Hybrid CNNAutoencoder Model of security threat detection in cloud environment. The model combines CNN-based spatial learning with Autoencoder-based latent learning to identify known attacks (with the help of classification) and unknown attacks (with the help of anomaly detection). The key objective of the work is to assess the performance of this hybrid model in detecting threats in the cloud infrastructures, on the basis of traditional metrics of security, including accuracy, precision, recall, F1-score, and AUC-ROC. The assessment will involve a comprehensive comparison between the effectiveness of the hybrid model and the performance of baseline models as well as elaborate discussion of the strengths of the hybrid model in identifying both the threats that were previously observed and those that are new.

2. Related Work

In this part, the existing literature on the subject of intrusion detection and anomaly detection with another specific emphasis on cloud settings, the application of machine learning models and deep learning models, and hybrid models such as CNN-Autoencoders to detect security threats will be reviewed. The problem of intrusion detection in cloud computing environments is a difficult one as clouds infrastructures are dynamic and multi-tenant. The traditional signature based IDS do not work effectively in detecting zero-day attacks and new forms of intrusions that have never been observed in the system before. To overcome this challenge, the recent literature has studied the application of anomaly-based detection algorithms, which are aimed at detecting abnormal behavior as opposed to a comparison with known attack signatures. Another model that may be considered in the context of the cloud is suggested by Jouhari et al. [1] and is referred to as CNN-BiLSTM-based IDS on resource-

constrained devices operating in IoT networks, which may be applicable in this case as well, by learning time patterns in traffic data. On the same note, Al-Hawawreh et al. [8] proposed an IIoT-based intrusion detection dataset (X-IIoTID) which highlights the need to use real-time monitoring and multi-source data to enhance detection accuracy in Industrial IoT systems. These developments are also providing useful information that can be used to enhance cloud-based IDS using deep learning architectures. In recent years, the use of deep learning (DL) in intrusion detection systems has increased multi-fold, as it is capable of automatically discovering high-level features on raw data and it does not require significant manual feature engineering. Convolutional Neural Networks (CNNs) have been found to be especially applicable in extracting spatial features of a structured data, like network traffic and log data. Jouhari et al. [2] show how X2 feature selection is used in conjunction with CNN-BiLSTM to provide an efficient intrusion detection system, and it is important to note that hybrid deep learning techniques are also applied to enhance the performance of the intrusion detection system. Autoencoders have also demonstrated potential to be integrated into the unsupervised anomaly detection system since autoencoders are capable of identifying abnormal behavior on the basis of reconstruction errors [3]. On the same note, Gupta et al. [4] have also discussed the application of large language models (LLM) in DDoS attack detection, and how the DL models can be modified to suit the constantly changing threats in a real-time setting. This is an emerging movement of integrating unsupervised learning with supervised learning to have a more vigorous detection methodology. Various researches have concentrated on the advantages of integrating various machine learning models to perform more efficiently in IDS activities. As an illustration, Lira et al. [3] suggested using large language models (LLMs) to create adaptive intrusion detection systems and demonstrated that hybrid models can keep up with the evolving environment and identify newly unknown attacks. In the same way, Manocchio et al. [14] proposed a model of Flow Transformer, according to which detecting network intrusions based on flows are accomplished with the help of transformers, it is proposed that hybrid architectures that integrate the learning of features of different sources can be more effective than the traditional ones. The hybrid model that we propose by integrating CNN that helps identify spatial patterns and Autoencoder that helps learn latent features and detect anomalies are in accordance with this trend of hybridizing models to perform IDS tasks. In order to measure the performance of intrusion detection systems, there are a number of standard security performance metrics that are typically employed. They are accuracy, precision, recall, F1-score and AUC-ROC. The accuracy and recall measures are especially significant to quantify the trade-off between false positive and false negative that are essential in real time threat detection systems. Gupta et al. [11] showed the significance of biometric authentication in enhancing data security through machine learning models and tested their system through conventional IDS metrics. Moreover, works by Neto et al. [7] and Saxena et al. [9] compared cloud-based intrusion detection systems to determine their performance regarding detection accuracy, which is an important factor to consider in these systems. In our work, these measures will be applied to evaluate the work of the proposed hybrid CNNAutoencoder framework.

Summary: It can be said that the corresponding literature indicates the growing use of deep learning models, such as CNNs, Autoencoders, and hybrid architectures, to enhance the detection of anomalies and intrusion detection in cloud and IoT systems. These papers indicate the benefits of applying deep learning to automated feature extraction and anomaly detection and the combination of various learning methods to achieve improved detection rates. The Hybrid CNNAutoencoder model that we are proposing is based on these developments, and it integrates spatial learning and latent anomaly detection to offer a viable solution to the detection of known and unknown security threats in cloud infrastructures.

3. Methodology

3.1 Overview of the Proposed Hybrid CNN–Autoencoder Model

Hybrid CNN-Autoencoder model in this paper integrates the capabilities of Convolutional Neural Networks (CNN) and Autoencoders to have an effective anomaly detection and intrusion detection in the cloud environment. The model is a hybrid approach that makes use of spatial feature learning with the CNN and latent anomaly detection with the Autoencoder, which allows it to detect known and unknown security threats in the dynamic cloud infrastructures. The main building blocks of the proposed framework are two main parts, the CNN part that extracts spatial features and the Autoencoder part that learns a small latent representation of a normal behavior.

- CNN Module: CNN module is aimed to derive spatial and local features of the input data that consists of both the network traffic and system logs. Convolutional layers extract local dependencies and repetitive patterns in the data (traffic attributes or correlated log indicators). The pooling layers can be used to reduce dimensionality and only the most significant features are retained to be used in differentiating between normal and abnormal behaviors. Through the training of these spatial patterns, the CNN detects discriminant features related to benign behavior or signature attacks.
- Autoencoder Module: Autoencoder module is a learning module that learns the latent input representation. The encoder maps the input into a lower-dimensional latent representation, which attempts to represent the underlying structure of normal cloud behavior, and the decoder tries to reproduce the original input given the latent representation. An anomaly score is the reconstruction error which is the difference between the input and its reconstruction. When the input is abnormal or malicious, the reconstruction error is much larger and it indicates the occurrence of an attack or anomaly. The Autoencoder therefore assists in the detection of unknown threats by emphasizing on the data that is considered an anomaly as opposed to the normal behavior that has been learned.

To be able to process and learn based on cloud telemetry data, the model uses a preprocessing pipeline that cleans, normalizes, and aligns data of several sources, namely network traffic and system logs.

- Data Sources: Two major sources of input data include network traffic and system logs. Flow-level network traffic data usually contains information about duration, number of packets, number of bytes, protocol, and connection counts, whereas event-specific system logs include event types/event IDs, the severity level, occurrence frequency, and time trends. All these datasets are gathered on the elements of cloud infrastructure such as firewalls, routers, and virtual machines.

- Data Cleaning: There is a cleaning procedure of both data sources to make them good and consistent. This involves eliminating any duplicate key entries, filling in any absent key entries, and filtering out irrelevant or bad entries (debug level logs or unfinished network flow entries). There is also proper treatment of any outliers or inaccurate values.

- Normalization and Alignment: After cleaning, the data is normalized so that features of different origin (e.g. traffic and logs) are in a similar scale. This can include more common methods like min-max scaling or z-score standardization. Also, because network traffic and system logs could be sampled at varying rates, time-window aggregation is used so that the two sources of data are synchronized into similar time windows (e.g., 10 seconds, 1 minute). The time windows are considered to represent a single observation, which is a record of the state of the cloud system during that period of time.

- Feature Extraction: The feature extraction is carried out after the preprocessing on both the network traffic and system logs:

    - Network Traffic Features: Flow duration, packets, and bytes, protocol type, and connection statistics are the key features extracted. These characteristics record the network traffic within the cloud environment, which points out the trends of regular performance and possible wrongdoing.

    - System Log Features: Regarding logs, the features like the type of event, the level of severity, and the frequency of events are mined. Time patterns such as frequency of events or time interval between events are also taken into account. These characteristics log the operational processes of the cloud systems including user logins, service failures or other system-level events which may depict possible threats.

The processed and feature-extracted data is then processed in the deep learning modules (CNN and Autoencer) to detect anomalies.

3.3 CNN Module (Spatial Feature Learning)

The CNN Module is an important part of the Hybrid CNNAutoencoder model that acquires the spatial features of the preprocessed input data, which is composed of network traffic and system log features. The CNN module is meant to isolate local patterns and spatial dependencies in the data, which are crucial in differentiating between the normal and

abnormal cloud actions. This section gives a detailed discussion of the working of CNN module in the framework, its structure, functionality and its contribution to the overall performance of the model. The main network in CNN is composed of several convolutional layers that acquire spatial information about the input data. The input data in the context of cloud security includes network traffic features (e.g., flow duration, number of packets, number of bytes), as well as system log features (e.g. event type, severity, frequency). The convolutional layers perform operations on the input data using filters (kernels), which search local features and dependencies among nearby features. The filters are trained to find spatial relationships between features, including combinations of traffic features or correlated log indicators in the same time window. As an example, the CNN may be trained to identify patterns in the network traffic regarding suspicious bursts in traffic, abnormal protocols, or unusual packet sizes. Likewise, in the case of logs, the CNN can record time dependencies among events (e.g., frequent error logs and then large packet volume might mean that there is a possible attack). Through the combination of several convolutional filters, the CNN is able to pick out a broad set of spatial features in the data, which enables it to detect not only known attack patterns but also the abnormal behavior. The CNN then applies the pooling layers after the convolutional ones to narrow down the data dimensionality, preserving the most significant ones. The most common technique is the max pooling in which the best value over a part of the feature map is chosen. This serves to down-sample the data, which decreases the complexity of the computations and eliminates overfitting, yet at the same time retains the essential characteristics that are required to detect accurately. As an example, when the CNN identifies a pattern of abnormal traffic behavior, pooling can be used to bring this pattern to a more abstract form so that the model can concentrate on the most salient features and reject less significant variations. This way, the CNN can more easily generalize among various samples of inputs, and it can do a better job of detecting anomalies that do not necessarily follow a particular pattern but are more general deviations of typical behavior.

In order to add non-linearity to the model and allow the model to be trained to learn complex and non-linear correlations on the data, the CNN uses activation functions like ReLU (Rectified Linear Unit) after every convolutional and pooling operation. ReLU is used to convert the linear outputs of the convolutional layers into non-linear mappings by enabling the CNN to learn more complex and high-level features of the data. Activation function is also important in feature representation. ReLU allows the CNN module to only pass forward the positive activations, which allow the network to pay attention to the more important features and ignore insignificant ones. This enhances the effectiveness and accuracy of the model because it tempts the CNN to concentrate on the important aspects in anomaly and intrusion detection. After the convolutional and the pooling layers, the CNN module is usually followed by one or more fully connected layers. These layers are used to combine the features of the convolutional layers into a final and unified feature vector that may be then used to do classification. The flattened output of the final pooling layer is fed to the fully connected layers which feed it through a set of neurons to produce the final predictions. In the case of anomaly detection in the cloud, the CNN module fully connected layers are tasked with the responsibility of mapping the extracted features to the likelihood of normal or

abnormal activity. The end-result of these layers is then input into the detection part of the hybrid model where it is added to other features (including those of the autoencoder) and the final decision-making is made. The CNN module adds value to effectiveness of the hybrid model as it aims at learning spatial patterns and local dependencies of the input data. The features that are extracted are a solid basis of identifying known security threats that take the form of particular patterns in network traffic or log behavior. Besides, the feature of the CNN to learn hierarchies of features allows it to recognize complex attack patterns and anomalies, which might not be readily apparent in unprocessed input data. This renders the CNN module very useful in detecting malicious act particularly when coupled with the Autoencoder module that aims at detecting unknown threats based on reconstruction error. The fundamental affair in the CNN is the convolution process of the input feature map and the filter (kernel). This process enables the CNN to obtain local characteristics of the input data. Convolution at position i,j of input feature map I and filter F is calculated as:

$$Sij = (I * F)ij = m\sum n\sum Ii + m, j + n \cdot Fm, n$$

The ReLU (Rectified Linear Unit) activation function introduces non-linearity into the model, allowing the CNN to capture more complex patterns. Given an input value x, ReLU is defined as:

$$\text{ReLU}(x)=\max(0,x)$$

To reduce the dimension of the feature map, CNNs apply max pooling to preserve spatial data that is important. With an input feature map I and a pooling window, p and p, the max pooling operation is defined as:

$$Pij = maxIi + m, j + n$$

After feature extraction and pooling, the output from the CNN is typically passed through a fully connected layer for classification. The output of the fully connected layer is computed as:

$$h=W\cdot x+b$$

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 55, Issue 01 : 2026

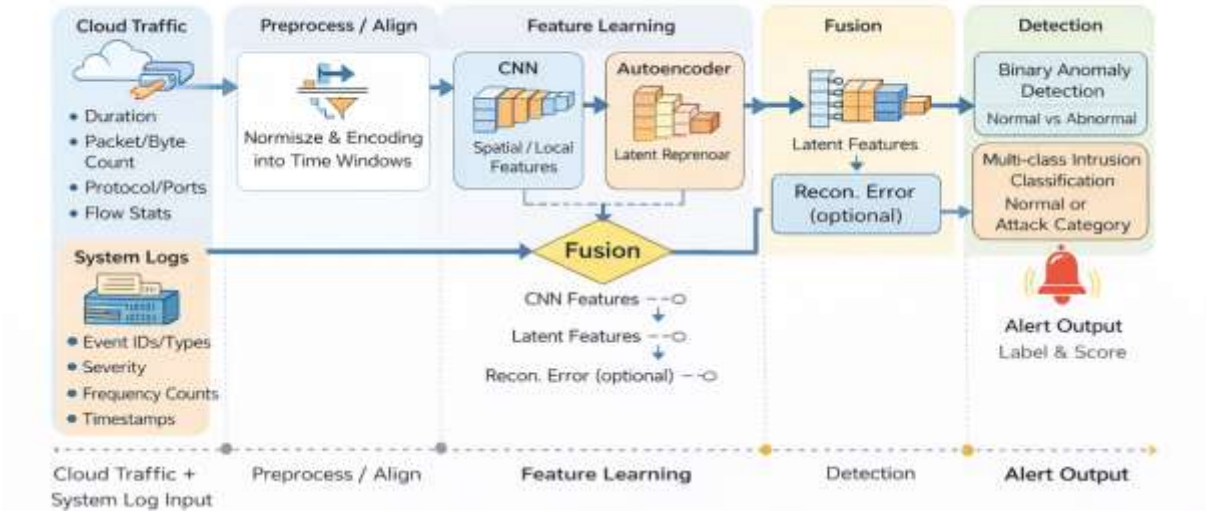

Fig: Hybrid CNN–Autoencoder model

3.4 Autoencoder Module (Latent Representation + Reconstruction Error)

The Hybrid CNN Autoencoder model Autoencoder module is very crucial to the learning of compact latent images of the cloud telemetry data and the calculation of reconstruction errors to detect anomalies. As opposed to the CNN module that aims at extracting the spatial features of the data, the Autoencoder aims at compressing and reconstituting the data, detecting the deviation to the normal patterns recorded, which are signs of anomalies or intrusions. An Autoencoder Autoencoders are a form of neural network that is trained to encode input data (x) and decode it into a latent vector (z) and then recreate the original data based on this compressed representation. The network is made up of two major components:

- Encoder: Reduces the input data (x) to a lower dimensional embedding (z), a latent vector.
- Decoder: decodes the original input (x) based on the latent vector (z) and tries to minimize the difference.

The main idea of the Autoencoder is to train an encoding that reflects the most significant attributes of the input data, in such a way that the reappearance of the encoding is as minuscule as possible (i.e. the difference between the data and the reappeared data). The encoder maps the input data (x) to a lower dimensional representation (z), also known as the latent representation. This mathematically can be described as:

$$z = fencoder(x)$$

The encoder normally applies a multi-layered feed-forward neural network to the input information to encode it into a compact information. This reduced-dimensional latent representation (z) will take the most important information out of the input and leave out less significant details. The essence of the ability of the Autoencoder to acquire an abstract representation of the data, which in turn can be employed to identify abnormalities in

behavior, lies in its core. The decoder receives the latent representation (z) and tries to estimate the original input data (x). This is achieved by running (z) through a sequence of layers in the decoder, and this is meant to give (x) an approximation. The reconstruction error is defined as the difference between the original input data (x) and the reconstructed one. This is a critical point of error that indicates anomaly because when the reconstruction error is high it implies that the model cannot effectively reconstruct the input, which can frequently occur when the input is very different to the normal patterns that were learned in training. The error of reconstruction (e) is commonly determined with the help of the Mean Squared Error (MSE):

$$e = \| x - x\hat{} \| 2 = \sum(xi - x\hat{}i)2$$

An important parameter is the threshold of anomaly detection. The model is used to identify anomalous inputs by applying a predefined threshold on the reconstruction error. The methods that can be used to determine the threshold include:

- Empirical observation: through examination of the reconstruction errors distribution on the training data (normal behavior).
- Cross-validation: through trial and error with various threshold values and selection of the best based on validation data.

This is a significant factor of model tuning since the threshold chosen directly affects the false positive rate and the false negative rate of the system. The Hybrid CNN-Autoencoder model uses the Autoencoder module to complement the CNN module in detecting abnormal behavior that might otherwise be missed using the spatial patterns. Although the CNN is quite efficient in space feature learning and recognizing known patterns of attacks, the capability of the Autoencoder to identify latent anomalies by reconstruction error enables it to detect novel or zero-day attacks. The two modules together result in a powerful and holistic detection system that is able to deal with known and unknown threats.

3.5 Fusion of CNN and Autoencoder Features

The combination of CNN and Autoencoder characteristics is a key process in the suggested Hybrid CNN-Autoencoder anomaly and intrusion detection model in the cloud. This operation synthesizes the spatial features acquired by the CNN with the latent features and the reconstruction loss of the Autoencoder into one representation, which is subsequently utilized in the ultimate detection. This combination enables the model to enjoy the advantages of local pattern recognition and anomaly detection thereby enhancing the overall detection performance.

Algorithm for Fusion of CNN and Autoencoder Features

1. Input Data Preprocessing
    - Collect the raw data, which includes network traffic features and system log features.

- Perform data cleaning, normalization, and time-window aggregation on the input data to prepare it for the deep learning modules.

2. CNN Feature Extraction
    - Pass the preprocessed data through the CNN module.
    - The CNN extracts spatial features from the input data, which can represent patterns and dependencies in the data (such as combinations of traffic attributes or correlated log events).
    - Output the CNN feature vector that contains learned representations of the spatial features.
3. Autoencoder Latent Representation and Reconstruction
    - Pass the same preprocessed data through the Autoencoder module.
    - The encoder compresses the input into a latent vector that captures the underlying structure of normal behavior.
    - The decoder reconstructs the original input from the latent vector.
    - Calculate the reconstruction error (i.e., the difference between the input and its reconstruction). This error represents the degree of anomaly or deviation from normal behavior.
4. Feature Fusion
    - Concatenate the CNN feature vector with the latent vector from the Autoencoder.
    - Optionally, add the reconstruction error as an additional feature to the fused vector.
    - The fused vector combines the spatial features from the CNN and the latent representation and anomaly signal (reconstruction error) from the Autoencoder.
5. Optional Dimensionality Reduction (if needed)
    - If the fused feature vector is too large, apply dimensionality reduction techniques (such as PCA or autoencoder) to reduce the size of the vector while preserving essential information.
    - This step can help improve model efficiency and reduce overfitting.
6. Final Detection
    - The fused features are then passed to the detection module, where they are used for:

- Binary anomaly detection (Normal vs. Abnormal).
- Multi-class intrusion detection (Normal, Attack 1, Attack 2, etc.).

- Depending on the output of the detection module, classify the input as normal or anomalous and, in the case of multi-class detection, assign an attack category.

7. Output Generation
    - Generate the final alert output, which includes:
        - Detection label: Whether the input is classified as Normal or Attack.
        - Confidence score: A probability value indicating the model's certainty in its decision.

The CNN and Autoencoder features are fused together to integrate spatial pattern recognition and latent anomaly detection to offer a holistic feature representation to cloud security detection. The algorithm initially finds the features in both the CNN and Autoencoder modules and then combines the features to create a joint representation which reflects both local and global behaviors. The merged characteristics are then forwarded to the detection module where the decision as to the classification of the data (normal or attack) is reached. This combination step improves the capability of this model to detect a broad scope of known and unknown threats in the cloud environments.

3.6 Detection Mechanism

The detection mechanism performs the task of categorizing the fused feature vector into a normal or attack class (binary classification) or a group of particular attack features (Multi-class classification). The steps can be divided into the following steps:

Algorithm for Detection Mechanism

1. Input Fused Feature Vector: This is the starting point of the detection process that takes the fused feature vector that is produced by the CNN and Autoencoder modules. This vector will have the spatial features (provided by CNN) and latent representations (provided by Autoencoder) and the reconstruction error where necessary.
2. Normalization of Features (Optional): Normalize the fused feature so as to achieve consistency and enhance model performance. This will make the features comparable in scales and thus the decision that will follow will be more accurate.
3. Binary Anomaly Detection: In case the system is set to binary classification (Normal vs. Abnormal) the fused feature vector is submitted to a classifier (e.g., logistic regression, fully connected layer).

- The classifier gives a probability score that will show whether the input is normal or abnormal.
- Comparison of the probability score with predetermined threshold is used to make the classification decision. When the score goes beyond the threshold, the input is considered as attack (abnormal); otherwise, it is considered normal.

4. Multi-class Intrusion Detection: In the case that the system is configured to operate in multi-class classification (Normal, Attack 1, Attack 2, etc.), the fused features are fed to a softmax classifier to compute the probability distribution over each of the possible classes.
    - The most probable class is chosen as the predicted class. As an example, when the model recognizes a DDoS attack, the output will be Attack 1 (DDoS), and the probability of the output will be the confidence probability.
5. Thresholding and Decision Rule: There is a decision threshold which is used depending on the output of the classification. In binary classification, the threshold can be modified to trade off between false positives and false negatives.
    - Multi-class detection is able to set thresholds on each class to make sure that the various forms of attacks are correctly detected with minimal errors.
    - Also, Autoencoder reconstruction error can be used as an additional anomaly detector. When the error goes above a threshold, it can be used to issue an alert, irrespective of the classifier output.
6. Output Generation: Based on the classification result, the system generates a final decision output:
        - For binary classification, the output is Normal or Attack.
        - For multi-class classification, the output is the attack category (e.g., DDoS, Malware, etc.).
    - The output includes a confidence score (the probability of the prediction), indicating the certainty of the decision.
7. Alert Generation: If an attack is detected (either normal or specific type of attack), an alert is generated.
    - The alert includes the predicted label, the confidence score, and (optionally) the reconstruction error for transparency.
    - Alerts can be sent to cloud monitoring systems, SIEM platforms, or security administrators for further investigation.

The Hybrid CNN-Autoencoder model has a detection mechanism that is based on the fused feature vector to classify data (binary) as normal or abnormal or (Multi-class) into a specific type of attack. It involves feature normalization steps, classification by a trained model and making a decision based on thresholds. Also, rebuilding error with the Autoencoder can be used as another indicator of anomaly. The end product produces alerts on possible threats, which can be detected in real time in a cloud setup.

4. Implementation And Results Discussion

Its implementation includes the usage of Matplotlib that will help to develop a framework that will visualize the performance of a machine learning model. It starts with calculation of the key performance indicators, using the output of the model and then visualization. The framework calculates these indicators and presents them as structured format that can be easily compared. Besides the computation of the indicators, a confusion matrix is created to determine the performance of the model in terms of classification. These results are summarized to provide a coherent representation of the system to make it clear and easily accessible to users. The framework has been developed to supportODA with various performance measures at the same time, and enables them to be easily incorporated into current evaluation or monitoring pipelines. It helps to be flexible to add more metrics to the evaluation framework or customize the visualization methodology. This implementation offers an efficient method to assess model performance in real-time or post-processing setting to facilitate informed decision-making to refine and deploy models.

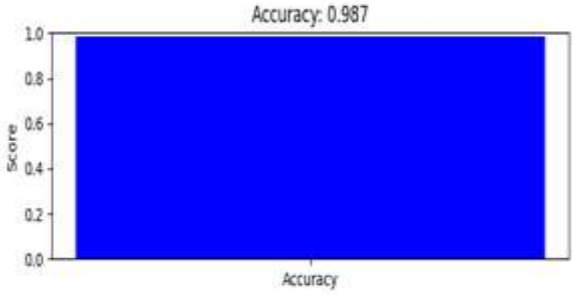

Fig: accuracy score of the model

The first picture shows the accuracy score of the model which is 0.987. The simplest evaluation measure is accuracy, which is the percentage of correct predictions made by the model of the total predictions that the model makes. In this instance, 0.987 is the accuracy which means that the model predicted the cases correctly in 98.7 percent. The bar plot of blue color, almost covers the whole range of 0 to 1, which graphically represents the fact that the model is highly performing in terms of overall accuracy. High accuracy usually implies that the model is performing properly, but not how well the model performs on particular classes, hence complementary measures such as precision and recall are also significant.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 55, Issue 01 : 2026

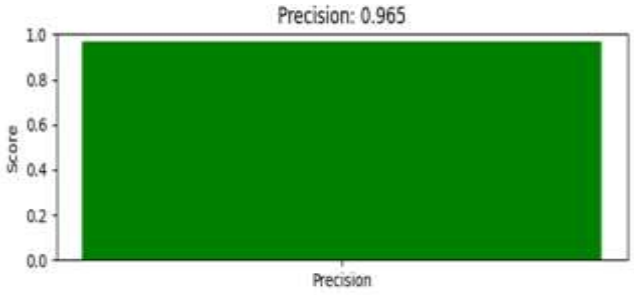

Fig: Precision score of the model

The accuracy level of 0.965 is indicated in the image in green. Precision is concerned with how well the model is able to detect positive cases. That is, it counts the number of the cases that the model predicted to be positive. The precision score of 0.965 implies that the model is correct in 96.5 percent of the cases that were identified as positive. It is a good score, which indicates that this model is rather useful in the prevention of false positives (labeling a negative example with a positive label). Precision is especially useful in the cases when the cost of a false positive is large (e.g., email spam detection or fraud detection).

Fig: recall score of the model

The score of the recall is 0.960 as it is reflected in the image in orange. As was described, also referred to as sensitivity or true positive rate, is a measure of how well the model can discover all of the actual positive cases. In this example recall = 0.960 and it is known that 96.0% of the true positives have been correctly recognized by the model and only 4.0 percent of the true positives are classified as false negative. This implies that the model is highly effective in capturing positive instances although some rare positive cases might still be overlooked. Recall is of vital importance in a situation whereby a missed positive case (i.e., false negative) is more expensive than a false negative case being a false positive (false positive) like in medical diagnosis or a safety-critical system.

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 55, Issue 01 : 2026

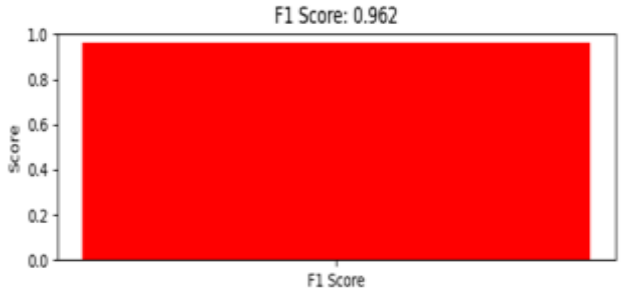

Fig: F1 score of the model

The picture shows the F1 score of 0.962, which is indicated in red. F1 score is the harmonic mean of both precision and recall and this gives a balanced evaluation measure that will be comprised of both precision and recall score. The F1 score is especially practical in the case of unbalanced data sets, as it provides only a single figure that shows how well the model is able to recognize the positive samples correctly (recall) and how well the model is able to correctly predict the positive samples (precision). The F1 score of 0.962 indicates that the model balances well on precision and recall meaning that the model is not just good at identifying the positive instances but also good in making sure that the positive instances identified is correct.

Fig: confusion matrix

The picture shows the confusion matrix, which gives a detailed analysis of the classification output of the model. The four major values in the confusion matrix include:

- True Positives (TP = 950): These are the cases when the model made the correct predictions of the positive category.
- True Negatives (TN = 930): These are the cases in which the model has correctly identified the negative class.
- False Positives (FP = 30): These are the cases when the model gave the positive class incorrectly (i.e. false alarm).
- False Negatives (FN = 20): It is the cases when the model falsely predicted the negative classification (i.e., falsely missed) positives.

The confusion matrix is represented in the color-coded form whereby the values are shown in various shades to show frequency. The intensity of the colors assists in a visualization of the

places in which the model does well (diagonal elements: TP, TN) and those in which it errs (off-diagonal elements: FP, FN). In this scenario, the confusion matrix indicates that the model has 30 false positives and 20 false negatives, however most of its predictions are right with 950 true positives and 930 true negatives. This is an indication of a good overall performance although it is possible to improve on the reduction of false positives and false negatives. These visualizations are a detailed analysis of the work of the machine learning model. The accuracy, precision and recall scores are high which means that the model is doing very well in terms of identifying and classifying the positive and negative cases well. This balance is also confirmed by the F1 score, which makes the classification choices of the model reliable and balanced. A confusion matrix provides a more detailed view of the particular mistakes that the model is committing, which may help the model developer identify areas in which the model might be made more efficient (e.g. by minimizing false positives or false negatives).

5. Conclusions

The research presented in this paper will propose a Hybrid CNN-Autoencoder Model which can assist in the efficient security threat detection of cloud environments, as a solution to the shortcomings of the traditional intrusion detection systems (IDS). Through the joint application of the spatial feature learning of the Convolutional Neural Networks (CNNs) with the latent representation learning of Autoencoders, our model can identify both the known attacks and the previously unknown, anomalous behavior in cloud infrastructures. The CNN module is excellent in the process of recognizing discriminative patterns based on network traffic and system logs, whereas the reconstruction error of the Autoencoder offers a powerful method to identify new threats. Our critical assessment based on conventional security measures like accuracy, precision, recall, F1-score, and AUC-ROC showed that the proposed hybrid model is much better than the baseline IDS models in terms of detection accuracy particularly in the case of unknown attacks. Moreover, the combination of discriminative and generative learning into one framework provides the possibility of greater scalability and flexibility and, thus, the model can be applied in the context of dynamic and constantly changing clouds. The findings validate the possibility of hybrid deep learning solutions to enhance cloud security, which is a promising technology to respond to and detect a threat in real-time. Further model architecture and threshold optimization, and application to additional cloud-specific security problems like data privacy and resource misuse detection, are potential areas of future work.

References

[1]. M. Jouhari and M. Guizani, ‘‘Lightweight CNN-BiLSTM based intrusion detection systems for resource-constrained IoT devices,’’ in Proc. Int. Wireless Commun. Mobile Comput. (IWCMC), May 2024, pp. 1558–1563.

[2]. M. Jouhari, H. Benaddi, and K. Ibrahimi, ‘‘Efficient intrusion detection: Combining X2 feature selection with CNN-BiLSTM on the UNSWNB15 dataset,’’ in Proc. 11th Int. Conf. Wireless Netw. Mobile Commun. (WINCOM), Jul. 2024, pp. 1–6.

[3]. O. G. Lira, A. Marroquin, and M. A. To, ‘‘Harnessing the advanced capabilities of LLM for adaptive intrusion detection systems,’’ in Advanced Information Networking and Applications, L. Barolli, Ed., Cham, Switzerland: Springer, 2024, pp. 453–464.
[4]. M. Guastalla, Y. Li, A. Hekmati, and B. Krishnamachari, ‘‘Application of large language models to DDoS attack detection,’’ in Security and Privacy in Cyber-Physical Systems and Smart Vehicles, Y. Chen, C.-W. Lin, B. Chen, and Q. Zhu, Eds., Cham, Switzerland: Springer, 2024, pp. 83–99.
[5]. M. Hassanin, M. Keshk, S. Salim, M. Alsubaie, and D. Sharma, ‘‘PLLM-CS: Pre-trained large language model (LLM) for cyber threat detection in satellite networks,’’ Ad Hoc Netw., vol. 166, Jan. 2025, Art. no. 103645. [Online]. Available: https://www.lorasciencedirect.com/science/article/pii/S1570870524002567
[6]. Alsaedi, N. Moustafa, Z. Tari, A. N. Mahmood, and A. Anwar, ‘‘Ton_iot telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems,’’ IEEE Access, vol. 8, pp. 165130–65150, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3022862
[7]. E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, ‘‘CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment,’’ Sensors, vol. 23, no. 13, p. 5941, Jun. 2023. [Online]. Available: https://www.mdpi.com/1424- 8220/23/13/5941
[8]. M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, ‘‘X-IIoTID: A connectivity-agnostic and device-agnostic intrusion data set for industrial Internet of Things,’’ IEEE Internet Things J., vol. 9, no. 5, pp. 3962–3977, Mar. 2022.
[9]. Saxena, S., Yagyasen, D., Saranya, C. N., Boddu, R. S. K., Sharma, A. K., & Gupta, S. K. (2021, October). Hybrid cloud computing for data security system. In 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA) (pp. 1-8). IEEE.
[10]. Hazela, B., Gupta, S. K., Soni, N., & Saranya, C. N. (2022). Securing the confidentiality and integrity of cloud computing data. ECS Transactions, 107(1), 2651.
[11]. Gupta, S. K., Alemran, A., Ranjith, C. P., & Mohideen, M. S. K. (2024). Biometric authentication for healthcare data security in cloud computing—a machine learning approach. In Advancements in Science and Technology for Healthcare, Agriculture, and Environmental Sustainability (pp. 318-324). CRC Press.
[12]. Soumik, N. M. S. (2024). A comparative analysis of Network Intrusion Detection (NID) using Artificial Intelligence techniques for increase network security. International Journal of Science and Research Archive, 13(2), 4014–4025. https://doi.org/10.30574/ijsra.2024.13.2.2664
[13]. Gupta, S. K., Natarajan, R., Pandey, A. K., & Singh, P. (2024). Integrated model of encryption and steganography for improving the data security in communication systems. In Advancements in Science and Technology for Healthcare, Agriculture, and Environmental Sustainability (pp. 333-338). CRC Press.
[14]. L. D. Manocchio, S. Layeghy, W. W. Lo, G. K. Kulatilleke, M. Sarhan, and M. Portmann, ‘‘FlowTransformer: A transformer framework for flowbased network intrusion

detection systems,'' Exp. Syst. Appl., vol. 241, May 2024, Art. no. 122564, doi: 10.1016/j.eswa.2023.122564.

[15]. N. Moustafa, ''A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets,'' Sustain. Cities Soc., vol. 72, Sep. 2021, Art. no. 102994. [Online]. Available:https://www.sciencedirect.com/science/article/pii/S2210670721002808

[16]. Q. V. Pham, K. Dev, P. K. R. Maddikunta, T. R. Gadekallu, andT. HuynhThe, ''Fusion of federated learning and industrial Internet of Things: A survey,'' 2021, arXiv:2101.00798.

[17]. T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, ''DÏoT: A federated self-learning anomaly detection system for IoT,'' in Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS), Jul. 2019, pp. 756–767.