A HYBRID CNN–AUTOENCODER FRAMEWORK FOR ANOMALY AND INTRUSION DETECTION IN CLOUD ENVIRONMENTS

D. Fernandez Raj, *Research Scholar, Texas Global University*

Dr B V V Siva Prasad, *Research Supervisor, Texas Global University*

Abstract

The environment of cloud computing produces very large amounts of heterogeneous telemetry, e.g., network traffic flows and system/system-application logs, and in both cases, the anomalies and intrusions are vital to identify in time and are often difficult to detect. Conventional signature-based intrusion detection systems are frequently inefficient against novel and ever-evolving attacks, whereas single-model deep learning systems can only be useful in capturing narrow characteristics of even complicated cloud behavior. In this paper, a security framework in cloud environments based on designing a hybrid CNN-Autoencoder will be presented with the aim of detecting anomalies and intrusions. The suggested architecture simultaneously learns spatial/local feature patterns in a convolutional neural network (CNN) and latent behavioral representations in an autoencoder with reconstruction based deviation scoring. Cloud traffic and system log characteristics are preprocessed, time window aggregated, encoded, and normalized into a single input representation. The CNN branch obtains discriminative feature embeddings, and the autoencoder branch generates latent vectors of compact size and reconstruction error signals to indicate abnormal deviations. These outputs are combined into one representation and sent to a detection module which can make a binary anomaly detection and optional multi-class intrusion classification using the threshold-based and probability-based decision rules. Compared to the studies that are implementation-intensive, the present work is dedicated to the entire methodological framework, data-flow pipeline, fusion strategy, and deployment appropriateness of cloud monitoring/SIEM integration, which gives a clear basis of the future experimental validation and performance analysis.

Keywords

Cloud Computing Security; Intrusion Detection System (IDS); Anomaly Detection; Hybrid Deep Learning; Convolutional Neural Network (CNN); Autoencoder.

1. Introduction

Cloud computing has turned out to be the workhorse of new digital services due to the provision of elastic resources, on-demand scaling, and low-cost deployment to organizations of any size [1]. Nevertheless, the very features that render cloud platforms appealing, like multi-tenancy, virtualization, distributed storage, and dynamic workload migration, expand the attack surface and complicate security monitoring as compared to traditional enterprise networks [2]. Cloud infrastructures constantly produce vast amounts of heterogeneous data, such as network traffic flows and system/application logs, and such data typically has useful signals that can reveal an abnormal behavior, misconfigurations, insider threats or ongoing cyber intrusion [3]. Anomaly detection methods and Intrusion Detection Systems (IDS) are

highly popular in the protection of networks and systems through detection of malicious activities and atypical behavior. The traditional signature-based IDS can be effective in identifying known attack patterns but cannot identify new or emerging threats. Subsequently, the methods of machine learning (ML) and deep learning (DL) have become interesting because they can identify trends in data and extrapolate to novel cases, where handwritten rules cannot be applied [4]. Recent research indicates the recent surge in the use of DL networks, including CNNs, recurrent models (LSTM/GRU), and autoencoders to solve intrusion detection and anomaly detection problems. Although successful, cloud environments pose further problems including extremely skewed attack data, sporadic telemetry, and quickly shifting normal operational behavior, which may amplify the number of false alarms and decrease detection rates [5]. The deep learning methods offer varying capabilities based on the model design. CNNs are powerful at locating local and spatial structures within structured inputs and have been massively utilized in feature learning tasks [6]. Autoencoders on the other hand are strong at learning small latent representations and predicting input data; anomalies tend to increase the reconstruction error and autoencoders can be applied to detect an anomaly [7]. Nevertheless, a single type of model might not be adequate in the cloud security monitoring. CNN-only models can fail to capture latent correlations between combined indicators and autoencoder-only models can fail to capture discriminative spatial feature patterns used to distinguish between benign bursts and malicious traffic [8-10].

In order to overcome these constraints, the present paper will offer a hybrid CNNAutoencoder security architecture to detect anomalies and intrusions in clouds. The main concept is to encode the representations of spatial/local features with CNNs and at the same time learn latent representation and reconstruction behavior with an autoencoder. Through this combination of these complementary representations, the framework is intended to identify both known intrusions (by learned discriminative patterns) and unknown or novel anomalies (by deviations as indicated in latent space and reconstruction error). Also, the framework is made to facilitate multi-source cloud telemetry which is a combination of network traffic features with system log features to enhance cross-layer visibility and eliminate the probability of overlooking cross-stage attacks, which present differently in various data sources. This is a design-oriented paper where the accent is on the architecture, data flow, and decision logic instead of reporting on the experimental findings. The architecture consists of (i) a preprocessing and feature-alignment step to convert cloud traffic and log entries into a format comprehensible to the model, (ii) a CNN network to learn spatial patterns, (iii) an autoencoder network to learn latent features and reconstruct them, (iv) a fusion step that takes spatial and latent features and (v) a detection head that takes spatial and latent features and identifies anomalies and intrusions with a threshold-based or classification-based decision rule.

2. Related Work

This section summarizes the previous literature in the field of cloud intrusion detection and anomaly detection, particularly, it discusses the machine learning, deep learning, and representation learning methods. A number of studies gives a general insight into intrusion

detection and anomaly detection methods. According to Ahmad et al. [1] and Mishra et al. [3], systematic studies of machine learning and deep learning methods to IDS are introduced with the limitation of signature-based system and the necessity of adaptive learning-based detection. The paper by Chandala et al. [7] is a very popular original source defining the concepts of anomaly detection, categories, and aspects of evaluation. Moreover, Al-Garadi et al. [11] survey ML/DL approaches to IoT security, which is pertinent since most of the cloud services can handle IoT-driven workloads and are likely to encounter similar large-scale and heterogeneous telemetry issues. These surveys are collectively inspirational to deep feature learning and multi-layered detection in complicated settings. Deep learning has also been exploited to acquire discriminative features directly on network data. The works by Gwon et al. [2], Kasongo and Sun [22], and others use sequence models (LSTM/GRU) and feature embedding to learn temporal correlations between traffic patterns. The authors Hussain and Hnamte [12] also show how the current deep learning pipelines could enhance the performance of the IDS with respect to the traditional ML. Moreover, it is suggested by Andresini et al. [19] that deep feature learning based on multi-channel would be a better way to learn richer representations, which is consistent with the concept of learning multiple views/features, instead of using one feature space. Although these techniques enhance detection, most of them are oriented around network traffic and do not necessarily use system logs or latent anomaly indicators to the fullest. Autoencoders have found extensive applications in anomaly detection since they are trained to recreate normal patterns, and abnormal behavior is therefore more easily identified by reconstruction error. Erfani et al. [5] integrate one-class classification with deep learning to deal with high-dimensional anomaly detection demonstrating the usefulness of representation learning in high-dimensional feature space. In the case of Yan and Han [17], stacked sparse autoencoders are utilized to extract features in order to boost intrusion detection, and it implies that learned compact features can boost the classification accuracy. Later on, Yang et al. [16] propose a supervised adversarial variational autoencoder as an intrusion detector, demonstrating that VAE variants can enhance generalization and decrease overfitting. Khan et al. [21] go even further to expand deep generative techniques using dual VAEs and Gaussian mixture modeling to identify anomalies, which makes sense since latent distributions may be useful to represent subtle anomalies. The studies support the idea of applying the autoencer-style latent learning to the IDS model, particularly to detect the unfamiliar or dynamic attacks. Hybrid deep architectures are intended to grip complementary aspects of information. CNN feature learning has been shown to have very high capability of local/spatial pattern extraction in structured inputs [4]. When applied to IDS, it is possible to enhance the robustness of the multiple representation learners by combining them since various attacks occur in different dimensions of features. Literatures such as the Andresini et al. [19] report that multi-channel representations are able to represent varied traffic behaviors. Likewise, the motivation behind a hybrid CNN-Autoencoder architecture, where CNN is specialized in spatial/local discriminative patterns and the autoencoder is specialized in latent reconstruction deviation, is supported by methods that use deep representation learning in pipelines of anomaly detectors [5], [16], [17]. One of the major challenges of intrusion detection is the issue of a class imbalance, in which the normal samples prevail over the attacks, which might be sparse or not distributed equally. Japkowicz [6] addresses the importance of the issue of class

imbalance and methods of dealing with it, which is quite acute in cloud IDS data. Other recent models of IDS specifically deal with imbalanced traffic and coarse-to-fine classification to enhance minor attack detection [15]. Also, real-time detection is essential in the cloud environments because of the pace of attacks and scope of services. Nizam et al. [20] present a real-time deep anomaly detection system of multivariate time-series in industrial IoT that has the same limitations as cloud monitoring systems (streaming data, low latency). These publications encourage the development of structures that can be effective and minimize false alarms and still have high sensitivity to detection.

Table: Summary of Related Work

Ref	Approach / Model	Data Type Focus	Main Idea / Contribution	Key Limitation (for cloud IDS)
[1]	Survey (ML & DL IDS)	Network traffic	Reviews IDS using ML/DL and trends	No specific hybrid multi-source design
[2]	LSTM + feature embedding	Network traffic (sequential)	Learns temporal patterns for intrusion detection	Mainly traffic-based; needs multi-source fusion
[3]	Survey (ML for IDS)	Network traffic	Detailed analysis of ML techniques for IDS	Not focused on deep hybrid representations
[5]	One-class SVM + deep learning	High-dimensional data	Scales anomaly detection to large feature spaces	Not tailored to cloud traffic + logs fusion
[6]	Class imbalance strategies	Generic	Explains imbalance problem and solutions	Does not propose a cloud IDS model
[7]	Survey (Anomaly detection)	Generic	Defines anomaly detection categories/methods	Needs modern DL/cloud-specific mapping
[11]	Survey (IoT security ML/DL)	IoT traffic & systems	Reviews ML/DL methods for security analytics	Not specific to cloud telemetry/log fusion
[12]	DL-based IDS (conference)	Network traffic	Shows DL improves IDS over classical methods	Limited detail on hybrid latent + spatial learning
[14]	Multi-stage deep pipeline	Network traffic	Uses staged DL pipeline for malicious traffic	Not explicitly spatial+latent hybrid learning
[15]	Coarse-to-fine IDS (imbalance)	Network traffic (imbalanced)	Improves classification for rare attack classes	Mostly supervised; less focus on unknown anomalies
[16]	Supervised adversarial VAE	Network traffic	Learns robust latent features using VAE + regularization	Does not combine CNN spatial features explicitly
[17]	Stacked sparse autoencoder	Network traffic	Feature extraction with AE improves IDS	AE-only may miss local discriminative patterns
[19]	Multi-channel deep features	Network traffic	Learns multi-view representations for IDS	Lacks explicit reconstruction-based anomaly cue
[20]	Real-time deep anomaly detection	Multivariate time-series	Streaming anomaly detection framework	Not focused on cloud IDS datasets specifically
[21]	Dual VAE + GMM	Attributed networks	Latent modeling with generative approach for anomalies	Complex; not designed for traffic+log fusion
[22]	GRU-based IDS	Wireless network data	Learns temporal dependencies efficiently	Mainly single-source; needs cross-layer visibility

Research Gaps

Based on the literature reviewed, it is apparent that deep learning, in particular, recurrent models, CNN-based feature learners, and autoencoder/VAE-based representation learning has been used to enhance intrusion and anomaly detection performance [2], [12], [16], [19], [22]. Nevertheless, cloud environments still have gaps: (i) a number of approaches concentrate on

only one source of data (only traffic or only system behavior), (ii) some of the models are more discriminative but weaker in detecting unseen anomalies, and (iii) imbalance and cloud normal variability is a challenge to handle [6], [15]. Thus, a hybrid CNN-Autoencoder architecture, in which the patterns of spatial features as well as latent reconstruction errors are learned, and is extendable to synthesize traffic and log telemetry, is well motivated. This is what the framework of this paper is based upon.

3. Methodology

3.1 Architecture of Proposed Framework

The framework that is proposed is aimed to identify anomalies and intrusions in cloud computing systems by collaboratively learning both spatial/local patterns and latent representations on two large sources of cloud telemetry: network traffic and system logs. The framework also integrates these complementary signals instead of using one type of data to enhance visibility through the cloud stack. On the whole, the architecture is based on a pipeline of Input to Feature Learning (CNN and Autoencoder) to Fusion to Detection, with each block contributing a specific type of representation to effective threat detection. During the input layer, the data of cloud traffic and system logs are gathered and converted into structured feature vectors. Traffic logs can contain flow level attributes such as duration, number of packets, number of bytes, protocol, and connection statistics, whereas logs can contain event types/ids, severity, frequency, and time based patterns based on system or application activity. Given that the two sources might not be of the same format and frequency, an initial shift (such as time-window aggregation) can be performed such that both traffic and log characteristics reflect the identical observation period. The resulting normalised feature matrix is the input of the deep learning modules. The CNN module will extract spatial and local features patterns in terms of the input representation that has undergone preprocessing. Spatial here means organized associations between features (e.g. combinations of traffic attributes or correlated log indicators at the same time window). Through convolution and pooling, the CNN is able to learn discriminative feature maps, which point to local dependencies and recurring patterns related to benign behavior or known attack patterns. This assists the model to learn the short-range feature interactions effectively and the model does not require any manual feature engineering. Simultaneously, the autoencoder module acquires a small latent representation of the same input and tries to recreate the original feature vector. The encoder reduces the input to a low-dimensional latent space that captures normal cloud behavioral patterns, whereas the decoder recovers the input as a result of this latent vector. When the abnormal or malicious activity takes place, the contribution will not resemble the acquired normal form, and the reconstruction error will be more significant. In this way, the autoencoder gives two useful cues, which are the latent embedding (reflecting hidden structure) and the reconstruction error (reflecting deviation to expected behaviour), which particularly come in useful when detecting unknown or zero-day anomalies. The fusion module then takes the information of both learning paths and creates a united feature representation. In particular, the framework is capable of concatenating CNN feature vector to the autoencoder latent one, and may also add reconstruction error as another indicator of anomaly. The significance of this fusion step is that it combines discriminative

spatial patterns (provided by CNN) and generative/latent deviation cues (provided by the autoencoder). Consequently, the nascent representation is more informative and resilient than either representation, particularly in the cloud environment where legitimate changing operational patterns can be similar to attack patterns. Lastly, the decision output is generated at the detection module. The output may be set to binary anomaly detection (normal vs abnormal) or multi-class intrusion classification (normal and specific attack types). In binary detection, a classification probability and/or a reconstruction- error threshold could be used to make the decision. In the multi-class detection, the combined features are fed through fully connected layers and a softmax classifier which assigns an attack classification. In both systems, the architecture facilitates a dynamic decision rule, which can be optimized to reduce the number of false alarms, yet keep high detection sensitivity in cloud systems.

3.2 Data Preprocessing and Feature Formation

The data of monitoring clouds is usually heterogeneous, noisy and has varying rates of generation. Thus, a preprocessing pipeline is needed to transform the records of network traffic and system logs into a clean and regular feature representation to be used in deep learning. The proposed framework will have preprocessing functionality that (i) eliminates data quality problems, (ii) standardizes traffic and log entries into a unified observation unit, and (iii) converts raw fields to normalized numerical features. Primary, unfiltered traffic and log data is filtered to eliminate duplicate records, missing records and corrupt records. Incomplete sessions or invalid values (e.g. negative duration, protocol/port field missing) are fixed or eliminated where feasible in the case of traffic flows. In the case of logs, message filters can be filtered by severity or source and missing entries (e.g. debug only messages) can be replaced with default values or removed depending on the noise level. The step of cleaning is necessary to make sure that the framework is informed by consistent and reliable signals in place of artifacts. The framework then extracts the features of each source of data individually. The attributes that are used to construct flow-level features of network traffic include duration, the number of packets/bytes, average packet size, source/destination port, protocol, and connection statistics. In the case of system logs, features are obtained based on event IDs/types, level of severity, frequency of events, and time-related trends (e.g., inter-arrival time, number of events of a particular type within a window). When there are text messages in the logs, they can be transformed into a straightforward form of indicators like flags on keywords or template IDs, maintaining the design light and standardized. Because traffic and logs might not even be synchronized (different time stamps and sampling rates), time-window aggregation step is implemented. In this step, all the traffic flows and log events that take place in a definite time frame (say, 5 seconds, 30 seconds or 1 minute) are bundled to constitute one observation. Traffic features can be aggregated in each window with sum/mean/max (e.g. total bytes, average duration), and log features with counts or frequencies (e.g. number of authentication failures, number of error-level events). The result of this is synchronized traffic-log feature vectors that characterize the identical cloud behavior interval. Following aggregation, categorical attributes (e.g. protocol type, port category, log severity, or event type) are encoded into a numerical format using label encoding or one-hot encoding. Numeric variables are then normalized by numeric scaling (including Min -Max scaling or standardization) to make all the features fall within similar

ranges. It is significant in that deep models are sensitive to the scale of features, and unscaled features can overshadow learning in an unfair way. Lastly, the processed results of network traffic and system logs are inputted into one feature matrix. The rows correspond to aligned observation windows and the columns to normalized traffic or log feature. This matrix is the last input of the hybrid CNN-Autoencoder network. To plan training and evaluation, the dataset may be divided into training, validation and testing partitions, and later resampling or threshold tuning (as part of the evaluation plan in the future) can be used to address the issue of class imbalance. In general, the given preprocessing plan will provide that both telemetry sources can bring meaningful and machine-readable patterns to perform anomaly and intrusion detection in a cloud setting.

For each time window (t), take:

- network traffic features (xn(t))
- system log features (xl(t))

Combine them into one vector:

$$xt = [xt(n) \;||\; xt(l)]$$

Normalize it:

$$x{\sim}t = Normalize(xt)$$

CNN extracts local/spatial patterns:

$$ht = CNN(x{\sim}t)$$

Encoder creates latent representation:

$$zt = Enc(x{\sim}t)$$

Decoder reconstructs input:

$$x\hat{}t = Dec(zt)$$

Reconstruction error (anomaly score):

$$et = \| x{\sim}t - x\hat{}t \|$$

Combine CNN and latent and error:

$$ut = [ht \;||\; zt \;||\; et]$$

3.3 Detection Logic steps for the hybrid CNN–Autoencoder framework algorithm

Step 1: Collect Cloud Telemetry

Constantly gather two concurrent streams of security telemetry of the cloud environment:

(a) gateway, VPC/VNet flow logs, IDS sensors, or virtual switch network traffic/flow records, and

(b) VM/container/authentication service system/application logs, and cloud audit logs. Always have a timestamp on every record so the records can be synchronized in the future.

Step 2: Clean and Validate Data

Do quality checks on both sources to eliminate noise and errors- delete duplicates and bad records, deal with missing fields (drop, fill, or default), filter out irrelevant lines in the log where necessary (e.g. all debug messages), etc.

This will minimize false alarms due to incomplete or noisy information.

Step 3: Convert Raw Fields into Features

Transform raw telemetry into measurable features, For traffic: duration, packets, bytes, protocol, ports, flags, rates, flow statistics

For logs: event IDs, severity levels, counts of key events (failed logins, errors), frequency indicators, and time-gap statistics
If log messages are unstructured text, convert them into structured indicators (event template IDs or keyword flags).

Step 4: Time Alignment Using Windowing

Because traffic and logs arrive at different speeds, align them using fixed time windows. (e.g., 10s/30s/1min).
For each window, aggregatetraffic features using sum/mean/max (e.g., total bytes, average flow duration), log features using counts/frequencies (e.g., number of authentication failures, number of error events)
This produces one combined observation per time window.

Step 5: Encode and Normalize the Feature Vector

Encode categorical values (protocol, log type, severity) using one-hot or label encoding.
Normalize all numeric features (Min–Max scaling or standardization).
This ensures features are comparable in scale and stabilizes deep learning training.

Step 6: CNN-Based Spatial/Local Representation Learning

Feed the preprocessed feature vector into the CNN branch (Conv + activation + pooling).
The CNN learns local and spatial relationships among features, such as correlated traffic patterns and log signals within the same window.
The CNN output is a compact feature embedding representing spatial/local patterns.

Step 7: Autoencoder-Based Latent Learning and Reconstruction

Concurrently, feed the same input to the autoencoder branch- the encoder encodes the input as a low-dimensional latent variable, the decoder recovers the original input given the latent variable.

Calculate a reconstruction error score (difference between original and reconstructed input).
The increased reconstruction error implies that there is some odd behavior and that the unknown anomalies are detected.

Step 8: Fuse Spatial Features, Latent Features, and Deviation Score

Combine (concatenate) the outputs from both branches- CNN embedding (spatial/local), latent vector (compressed behavior representation), reconstruction error (anomaly score) as an optional fusion feature.
This fusion creates a stronger joint representation than any single feature set alone.

Step 9: Threat Detection Decision (Binary / Multi-Class)

Send the fused vector to the detection head:

- For binary mode: output probability of attack vs normal
- For multi-class mode (optional): output probabilities for different attack categories
  A final decision can useclassification threshold (probability-based)
  and/orreconstruction threshold (error-based)
  For example: declare Attack if classifier score exceeds a threshold OR reconstruction error exceeds a threshold.

Step 10: Alert Generation and Logging

Generate a security alert containingpredicted label (Normal/Attack), confidence score (classifier output), anomaly score (reconstruction error), time window and key features contributing to the alert (optional). Store the alert for auditing and incident response workflows, and stream alerts to SIEM/SOC tools if deployed in production.

4. Implementation And Results Discussion

The hybrid CNN -Autoencoder framework proposed is intended to act as a cloud security analytics pipeline which receives network traffic characteristics and system log characteristics, preprocesses the characteristics into synchronized time-windowed records, and uses deep representation learning to perform detection. During implementation, cloud traffic (e.g., flow statistics duration, packets, bytes, protocol and port attributes) and system logs (e.g. event IDs/types, severity, frequency counts) are initially cleaned, encoded and normalized and aggregated into fixed windows such that both sources represent identical observation intervals. The resulting feature vectors are simultaneously inputs of a CNN module to learn patterns of spatial/local features and an auto-encoder to learn a latent representation, which is compact, and reconstruction behavior; they are used to compute an error as an indicator of an anomaly. Then, CNN embeddings, latent vectors, and (optionally) reconstruction error are combined into a single joint feature representation, which is then fed into a detection head that produces either a binary (normal vs attack) decision or a multi-class intrusion label. Lastly, the system produces an alert with the estimated label and the confidence/anomaly score, which makes the design appropriate to be integrated with cloud monitoring or SIEM pipelines to monitor security in near real-time.

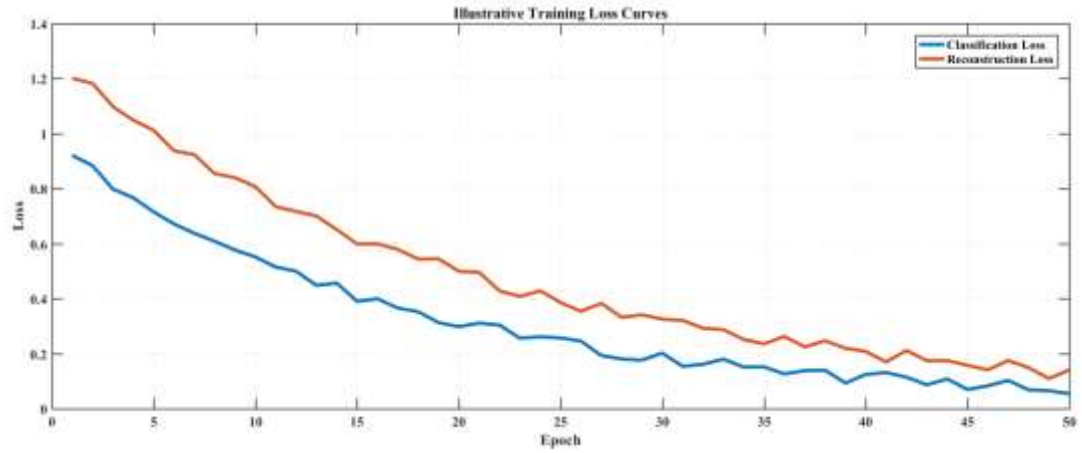

Fig: Training Loss Curves

This number demonstrates the learning process of the hybrid CNNAutoencoder model with respect to training epochs by presenting two losses: classification loss and reconstruction loss. The loss in classification progressively reduces with the CNN-based detection head becoming increasingly capable of distinguishing between normal activity and attack activity by using learned patterns of features. Meanwhile, the reconstruction loss also goes down as the autoencoder learns a low-dimensional latent representation of typical cloud behavior and becomes more able to reconstruct normal-like inputs. The continuous decreasing movement of both curves is a sign of constant training and the fact that the two learning aims (discriminative learning to detect and reconstruction learning to anomaly cues) are enhancing concurrently is also evidence to support that the motivation of the hybrid architecture usage.

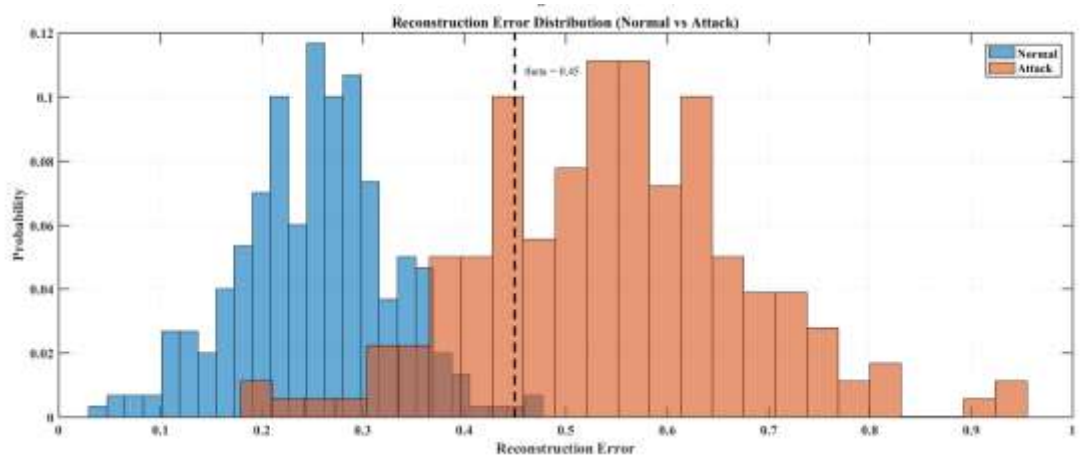

Fig : Reconstruction Error Distribution (Normal vs Attack)

The reconstruction errors generated is compared in this histogram when the triggers of normal and attack samples are used. Normal samples are more likely to produce lower reconstruction error, since the autoencoder is trained to recreate normal cloud behavior patterns, and it can be reconstructed successfully. On the other hand, attack samples move further in the direction of increased reconstruction error since malicious or anomalous examples do not fit the learned normal structure and are more difficult to reconstruct through the autoencoder. The vertical line (θ=0.45) is an example of anomaly threshold: the samples

with a value exceeding this one can be considered suspicious. The visual process of partial separation of the two distributions indicates the usefulness of reconstruction error as an indicator of an anomaly, especially in the detection of previously unknown threats that do not necessarily have corresponding attack signatures.

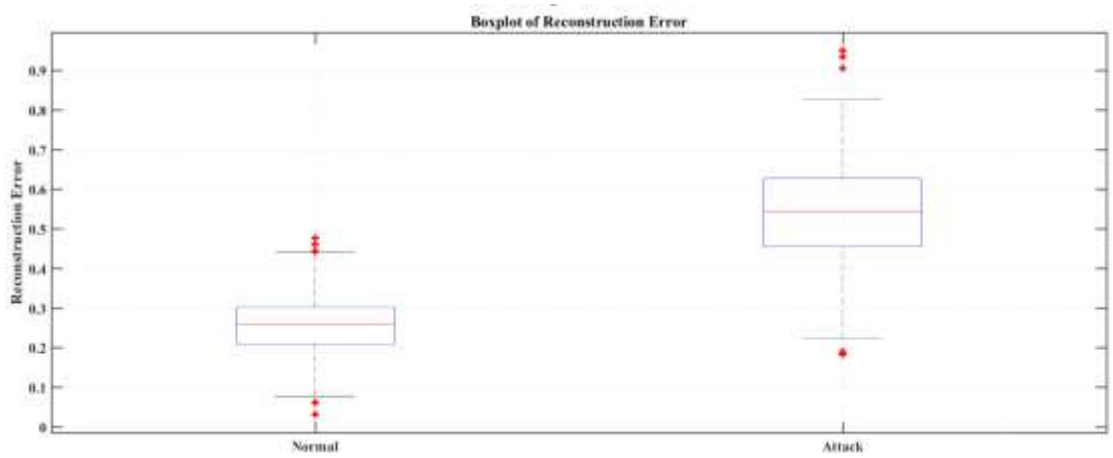

Fig: Boxplot of Reconstruction Error

The boxplot gives a brief statistical account of the reconstruction error of both classes (normal and attack). The median reconstruction error and the spread (interquartile range) of the attack group is higher and more dispersed in value, indicating that the attack samples not only give higher errors on average, but also are more dispersed. This is anticipated, as various types of attacks may result in varying levels of deviation of the normal cloud behavior, which causes varying reconstruction challenge. The normal group also has less median and smaller range meaning it is more reproducible under the normal conditions of operation. Comprehensively, this plot empowers the view that reconstruction error can be a powerful anomaly score and can be used to support the use of it in addition to the classifier output in the hybrid decision rule.

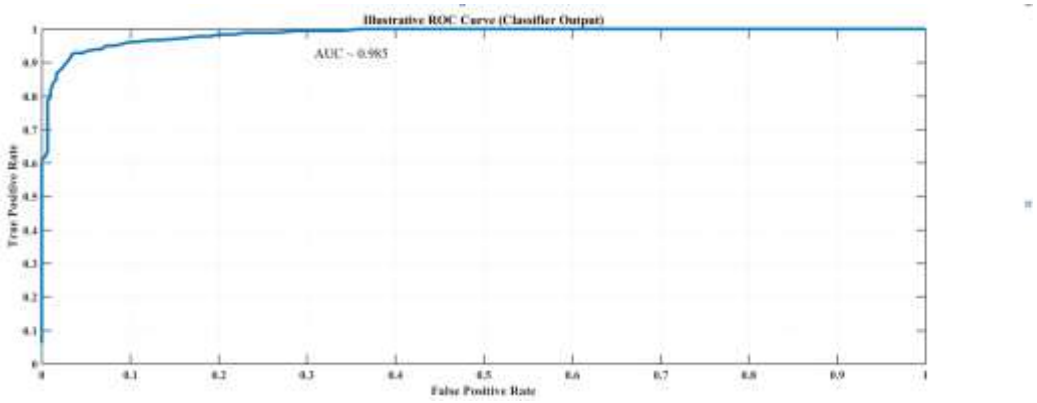

Fig: ROC Curve (Classifier Output)

ROC curve is used to measure the decision-making capacity of the classifier to differentiate between attacks and normal behavior at all potential decision thresholds. The curve is steep at the top-left area, which implies that the model has the potential to get a high true positive rate (identify most attacks) and a low false positive rate (label normal activity as attacks). In this

illustrative example, the AUC printed on the plot is enormously large, indicating that there is a strong separability in the scores of the classifier. In theory, this justifies making the CNN-derived features (and fused representation) to generate confident classification results, and it also demonstrates that the choice of threshold has an impact on the trade-off between detection sensitivity and false alarms, which is a point of concern when deploying cloud IDS.

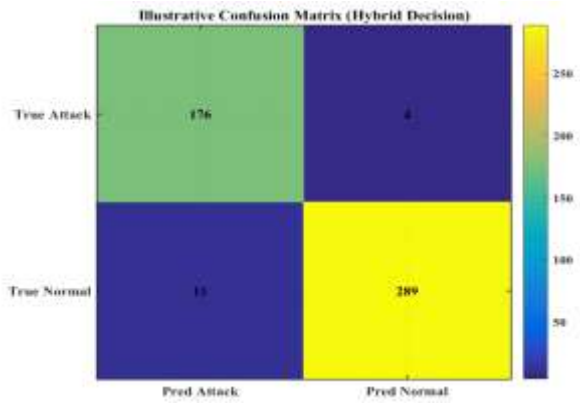

Fig: Confusion Matrix (Hybrid Decision)

The confusion matrix is the overall performance in detection sessions when the hybrid decision logic (based on classification probability and/or reconstruction error thresholds) is used. The diagonal cells depict accurate prediction: true attacks are predicted to be attacks and true normals are predicted to be normals. The cells that are off-diagonal indicate the error: false positives, when normal activity is detected as an attack (which adds to the alert fatigue) and false negative, when an attack is not detected (which is a serious risk). The diagonal values in this figure are predominant, i.e., the number of samples that are correctly classified is large and the number of errors is minimal, i.e. false alarm and miss rates are low at the selected thresholds. Such a plot will be useful, as it will convey the practical working effect of the model, how many alerts are correct versus how many are mistakes.

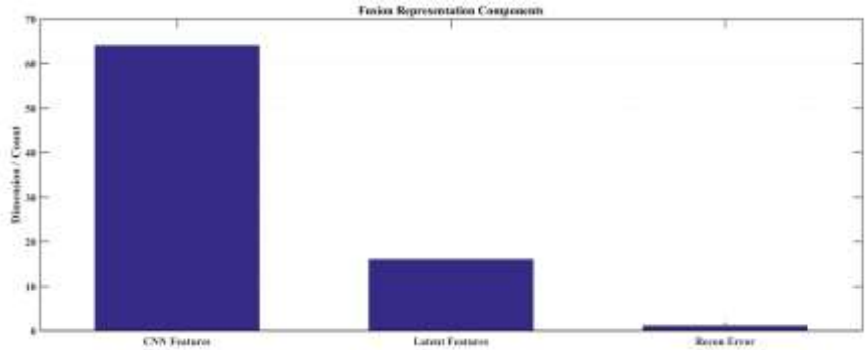

Fig: Fusion Representation Components

This bar plot describes the makeup of the fused feature vector that is used during the detection step. The biggest part is the CNN feature embedding, which is spatial/local relationships of traffic and log features on a time window basis. The second element is related to the autoencoder latent representation, which represents compressed behavioral structure trained on normal patterns. The last minor element is the reconstruction error, which is a

direct deviation mark that determines the abnormality of a sample. Collectively, these elements show how the suggested framework can integrate complementary information, namely, discriminative spatial features (CNN) and generative latent/anomaly cues (autoencoder) to produce a stronger representation of detecting known intrusions and unknown anomalies in cloud environments.

5. Conclusion

The paper introduced a design-based hybrid CNN Autoencoder model in the detection of anomalies and intrusions in cloud computing systems. The suggested methodology merges two complementary learning abilities: CNN module identifies pattern of spatial/local features on structured cloud telemetry and auto encoder module learns latent representations that are compact and provides a reconstruction based deviation signal which can be useful in detecting abnormal and possibly unknown behaviors. The framework enables an approach to multi-source visibility by preprocessing network traffic attributes with system log attributes and aligning the time window, encoding, and normalizing the data before the attributes are combined to support multi-source visibility and eliminate the dependency on a single stream of data. The fusion module combines CNN embeddings, latent vectors and optional reconstruction error into a common representation, which allows whether to detect anomalies in binary mode, or in optional multi-class mode of intrusion classification. On the whole, the framework provides a well-defined architectural blueprint that could be used to monitor the cloud and deploy SIEM-style. The future research will be devoted to the complete implementation and experimental testing on real or benchmark datasets, comparison of the model with baseline models, ablation testing, threshold tuning, and performance under class imbalance and evolving attack conditions.

References

[1]. Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," Trans. Emerg. Telecommun. Technol., vol. 32, no. 1, pp. e4150, 2021.

[2]. H. Gwon, C. Lee, R. Keum, and H. Choi, "Network intrusion detection based on LSTM and feature embedding," 2019, arXiv:1911.11552.

[3]. P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," IEEE Commun. Surveys Tuts., vol. 21, no. 1, pp. 686–728, 1st Quart., 2019.

[4]. H. Lee, M. Park, and J. Kim, "Plankton classification on imbalanced large-scale database via convolutional neural networks with transfer learning," in Proc. IEEE Int. Conf. Image Process. (ICIP), Phoenix, AZ, USA, Sep. 2016, pp. 3713–3717.

[5]. S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "Highdimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," Pattern Recognit., vol. 58, pp. 121–134, Oct. 2016, doi: 10.1016/j.patcog.2016.03.028.

[6]. N. Japkowicz, "The class imbalance problem: Significance and strategies," in Proc. Int. Conf. Artif. Intell., vol. 56, 2000, pp. 111–117.

[7]. V. Chandola, A. Banerjee, and V. Kumar, ‘‘Anomaly detection,’’ ACM Comput. Surv., vol. 41, no. 3, pp. 1–58, Jul. 2009, doi: 10.1145/1541880.1541882.

[8]. R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, ‘‘Deep learning and its applications to machine health monitoring,’’ Mech. Syst. Signal Process., vol. 115, pp. 213–237, Jan. 2019, doi: 10.1016/j.ymssp.2018.05.050.

[9]. J. Yin and W. Zhao, ‘‘Fault diagnosis network design for vehicle on-board equipments of high-speed railway: A deep learning approach,’’ Eng. Appl. Artif. Intell., vol. 56, pp. 250–259, Nov. 2016, doi: 10.1016/j.engappai.2016.10.002.

[10]. J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, ‘‘Deep learning for smart manufacturing: Methods and applications,’’ J. Manuf. Syst., vol. 48, pp. 144–156, Jul. 2018, doi: 10.1016/j.jmsy.2018.01.003.

[11]. M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, ‘‘A survey of machine and deep learning methods for Internet of Things (IoT) security,’’ IEEE Commun. Surveys Tuts., vol. 22, no. 3, pp. 1646–1685, 3rd Quart., 2020, doi: 10.1109/COMST.2020.2988293.

[12]. J. Hussain and V. Hnamte, ‘‘Deep learning-based intrusion detection system: Modern approach,’’ in Proc. 2nd Global Conf. Advancement Technol. (GCAT), Oct. 2021, pp. 1–6, doi: 10.1109/GCAT52182.2021.9587719.

[13]. M. Pradhan, S. K. Pradhan, and S. K. Sahu, ‘‘Anomaly detection using artificial neural network,’’ Int. J. Eng. Sci. Emerg. Technol., vol. 2, no. 1, pp. 29–36, Jan. 2012.

[14]. H. Nasreen Fathima and S. P. S. Ibrahim, ‘‘Multi-stage deep investigation pipeline on detecting malign network traffic,’’ Mater. Today, Proc., vol. 62, pp. 4726–4731, Jan. 2022, doi: 10.1016/j.matpr.2022.03.211.

[15]. Y. A. Jerusha, S. P. S. Ibrahim, and V. Varadharajan, ‘‘An effective network intrusion detection model for coarse-to-fine attack classification of imbalanced network traffic,’’ Int. Res. J. Adv. Sci. Hub, vol. 5, pp. 531–540, May 2023, doi: 10.47392/irjash.2023.s072.

[16]. Y. Yang, K. Zheng, B. Wu, Y. Yang, and X. Wang, ‘‘Network intrusion detection based on supervised adversarial variational auto-encoder with regularization,’’ IEEE Access, vol. 8, pp. 42169–42184, 2020, doi: 10.1109/ACCESS.2020.2977007.

[17]. B. Yan and G. Han, ‘‘Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system,’’ IEEE Access, vol. 6, pp. 41238–41248, 2018, doi: 10.1109/ACCESS.2018.2858277.

[18]. X. Shi, Z. Chen, H. Wang, D. Yeung, W. K. Wong, and W. Woo, ‘‘Convolutional LSTM network: A machine learning approach for precipitation nowcasting,’’ in Proc. Adv. Neural Inf. Process. Syst., Jan. 2015, pp. 1792–1806.

[19]. G. Andresini, A. Appice, N. Di Mauro, C. Loglisci, and D. Malerba, ‘‘Multi-channel deep feature learning for intrusion detection,’’ IEEE Access, vol. 8, pp. 45346–45359, 2020, doi: 10.1109/ACCESS.2020.2976874.

[20]. H. Nizam, S. Zafar, Z. Lv, F. Wang, and X. Hu, ‘‘Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT,’’ IEEE Sensors J., vol. 22, no. 23, pp. 22836–22849, Dec. 2022, doi: 10.1109/JSEN.2022.3211874.

[21]. W. Khan, M. Haroon, A. N. Khan, M. K. Hasan, A. Khan, U. A. Mokhtar, and S. Islam, "DVAEGMM: Dual variational autoencoder with Gaussian mixture model for anomaly detection on attributed networks," IEEE Access, vol. 10, pp. 91160–91176, 2022, doi: 10.1109/ACCESS.2022.3201332.

[22]. S. M. Kasongo and Y. Sun, "A deep gated recurrent unit-based model for wireless intrusion detection system," ICT Exp., vol. 7, no. 1, pp. 81–87, Mar. 2021.

