A SELF-HEALING FRAMEWORK FOR CLOUD AND VIRTUALIZED NETWORKS USING EXPLAINABLE AI TECHNIQUES

Gokula Krishnan K, Ganesh N, Joe Louis Paul I*
Department of Information Technology
Sri Sivasubramaniya Nadar College of Engineering
Chennai, India
gokulakrishnan2210046@ssn.edu.in, ganesh2210586@ssn.edu.in, joelouisi@ssn.edu.in

***Abstract*—The complexity of modern networks has increased because of the widespread use of cloud computing, virtualization, 5G technology, and extensive Internet of Things deployments. These environments create high volumes of active network traffic, making it hard for network operators to manage operations with standard manual methods. Operators struggle to identify problems early while also needing quick fixes for system faults. Their system performance must effectively handle different types of workload changes.**

Self-healing networks offer a solution to this issue by detecting faults, identifying their root causes, and restoring network functions without human intervention. These systems use machine learning to spot unusual behavior, predict failures, and automatically start response processes as soon as issues arise. This approach gives operational benefits by preventing system interruptions, reducing configuration errors, and lowering business costs through proactive responses.

Automated decision systems provide operational advantages, but organizations still prefer human decision-making. The main challenge comes from complex AI systems that act like black boxes, leaving users unable to see how they work internally. Therefore, implementing Explainable Artificial Intelligence (XAI) becomes essential for any designs using self-healing systems in real-world settings. The system employs XAI to offer clear explanations, helping users build trust, conduct audits, and ensure the system functions as intended.

This research develops a self-healing framework that combines AI with XAI methods to monitor, diagnose, and fix faults in cloud and virtualized networks. The framework automates fault management while providing explanations through visual displays that allow operators to explore root causes and recovery options. The study looks into how supervised learning methods can detect anomalies.

***Index Terms*—Self-healing networks, Software Defined Networks, Network Function Virtualization, Explainable AI, machine learning, predictive analytics, network automation, anomaly detection, fault resolution, autonomous network management**

I. Introduction

Modern networks have become much more complex with the rise of cloud computing, virtualization, 5G, and large-scale IoT deployments. These environments create huge amounts of dynamic interactions across distributed systems, making traditional manual network management hard to scale. Operators find it challenging to detect issues early, respond quickly to failures, and maintain consistent performance as workloads constantly change.

Self-healing networks aim to solve this problem by automatically identifying faults, figuring out their causes, and restoring services without relying on human help. These systems use machine learning techniques to find anomalies, predict failures, and trigger fixes in real time. Since they act proactively instead of waiting for disruptions, they help reduce downtime, prevent configuration errors, and lower operating costs.

Despite these benefits, many organizations are hesitant to fully trust automated decision-making. Their main worry is the lack of visibility into complex AI models that act like black boxes. This creates a need for Explainable Artificial Intelligence (XAI) in any effective self-healing design. By providing clear reasoning behind alarms, predictions, and recovery decisions, XAI builds trust, supports audits, and helps operators confirm that the system performs as intended.

This study presents a self-healing framework that combines AI and XAI techniques for monitoring, diagnosing, and recovering from faults in cloud-based and virtualized networks. The framework automates fault management and also offers explanations through visual and easy-to-understand outputs that help operators grasp root causes and recovery steps. The work examines supervised learning for detecting anomalies, unsupervised models for behavioral analysis, and reinforcement learning for independent decision-making. The goal is to contribute to the development of adaptable, trustworthy, and resilient self-healing systems that are suitable for next-generation network infrastructures.

II. Literature Survey

Early studies on cloud and network resilience laid the foundation for predictive and self-healing systems. They provided simulation platforms and datasets for experimenting with large-scale infrastructures in a controlled environment. CloudSim, introduced by Calheiros et al. [1], allowed detailed modeling of data centers, virtual machines, and resource

allocation strategies. This supported research on performance, cost, and QoS. While influential, it lacked newer features like multi-cloud operation and container-based deployments. Building on this foundation, Elhabbash et al. [2] expanded CloudSim Plus with a fault-injection framework. This framework let researchers assess system robustness during node outages and network issues, highlighting the growing demand for AI-driven recovery methods and diverse, large-scale simulations.

Early attempts to integrate AI into network self-healing were explored by Perumallaplli [3]. He combined machine learning, anomaly detection, and reinforcement learning to automate fault diagnosis and corrective actions. However, challenges like scalability and adapting to rapidly changing technologies remained. At the same time, Panigrahi and Borah [4] examined the CICIDS2017 dataset to aid intrusion detection research. They pointed out its realistic traffic patterns but also its limitations in class imbalance and the absence of emerging attack types.

Explainable AI techniques like LIME [5] and SHAP [6] were developed to help interpret the behavior of complex ML models. These techniques provided feature-level explanations, but both faced limitations in computational cost and scaling for large deployments. Bai et al. [7] compared various sequence modeling architectures. They found that Temporal Convolutional Networks (TCNs) often outperformed RNNs and LSTMs in stability and parallelization. However, challenges persisted with hybrid architectures and extremely long sequences.

The AI-supported self-healing ideas later expanded to power and communication infrastructures [8] and cellular networks [9]. These approaches used machine learning, deep learning, and reinforcement learning to detect anomalies, predict failures, and automate recovery. Despite promising results, issues with real-time performance, data quality, and rapid environmental changes still hindered production deployment. Enterprise networks [10] and hybrid intrusion detection systems [11] adopted ensemble and rule-enhanced models to improve detection accuracy and reduce false positives, though they still needed better training data and runtime adaptability.

In cloud computing, AI-based solutions [10], [12], [13], [16], [18] incorporated LSTMs, predictive analytics, and explainable techniques. They predicted failures, triggered automatic responses like workload migration, and optimized resource usage. These systems showed clear improvements in reliability and reduced downtime. Additional contributions included the Raziq performance metrics dataset [14] and EdgeLens [15]. These tools provided benchmarking resources and real-time anomaly detection for edge environments, but they still faced limitations in dataset diversity, continuous retraining, and managing dynamic workloads.

Further advances in SDN-based self-healing were presented by Nobre et al. [17]. They demonstrated how centralized control and analytics could enhance fault visibility and support dynamic flow reconfiguration. Remaining challenges involve scaling controllers and achieving seamless multi-domain integration. Together, these contributions show a shift from simulation-driven experimentation to advanced AI and deep-learning architectures. These developments enable predictive, proactive, and explainable self-healing capabilities, while also highlighting ongoing challenges in scalability, adaptability, real-time inference, and deployment across various cloud and network environments.

III. Methodology

A. Overview

The proposed methodology unifies multiple approaches, with a focus on addressing current gaps in self-healing and AI managed networking solutions. The methodology, via simulation, predictive analytics, and explainable functionality enables the ability to identify, predict, and correct faults associated with cloud and virtualised platforms in real-time. Details regarding the main components and phases of the methodology are provided.

B. Data Collection and Preprocessing

Historical and live network logs, resource monitors, and cloud telemetry tools provide the data for the system to make predictions. Typical metrics consist of CPU and memory uses, disk I/O and network throughput [3], [14]. There are three types of preprocessing operations that are performed on the dataset: normalizing data, processing missing values and creating temporal/stats features. Sampling strategies and reweighting strategies provide a method of handling class imbalance within anomaly datasets.

C. Predictive Modeling

1) Network Anomaly Detection: In this work, an ensemble approach based on Long Short-Term Memory (LSTM) networks and Multilayer Perceptrons (MLP) is investigated as a means of performing anomaly detection on sequential data (via the LSTM branch) and non-sequential data (via the MLP branch). Ultimately, utilizing LSTMs for modeling the sequential dependencies, and MLPs for processing the non-sequential inputs provides complementary strengths to enhance the overall accuracy of detection, while minimizing false positives [16], [18].

2) Resource Usage Forecasting: Temporal Convolutional Networks (TCNs) are used to forecast CPU and RAM utilization. TCNs offer stable long-range sequence modeling with efficient parallel computation, making them suitable for resource prediction in dynamic cloud workloads [7].

3) Imbalanced Anomaly Classification: LightGBM is applied to classify anomaly categories under skewed class distributions. Its gradient-boosted structure improves minority-class sensitivity and provides computational efficiency [8], [11].

D. Automated Self-Healing Actions

Outputs from the detection and prediction models feed into a decision layer responsible for triggering recovery actions. Depending on the detected event, the system may initiate traffic

rerouting, workload migration, resource scaling or configuration adjustments to maintain performance and availability [8], [12], [18].

E. Evaluation Metrics and Validation

1) Anomaly Detection Metrics: Accuracy, Precision, Recall and F1-Score are used to evaluate the anomaly detection models:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

2) Resource Forecasting Metrics: The TCN model is evaluated using MSE, RMSE, MAE, R^2, and Pearson correlation:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{5}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{6}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{7}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{8}$$

$$\text{Corr} = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(\hat{y_i} - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}\sqrt{\sum_{i=1}^{n}(\hat{y_i} - \bar{\hat{y}})^2}} \tag{9}$$

3) Self-Healing Metrics: System-level performance is evaluated using detection latency, recovery time, uptime and service availability [8], [9], [13]. Cross-validation and controlled simulations are used to assess model robustness.

F. Integration and Scalability

The models are integrated into a modular framework that supports scaling across cloud nodes and heterogeneous network devices. For a consistent results when in different workloads and traffic patterns, recurrent retraining and continuous learning are implemented [15], [18].

G. Summary

The approach uses LSTM-MLP ensembles to detect anomalies, TCN's to predict resource use, and LightGBM to classify based on an imbalance sensitivity. When combined with an automated response mechanism, this system can provide proactive and reliable self-healing in cloud and virtualized networks.

IV. System Design and Architecture

The architecture of this system is designed as a multi-layer system whereby it detects failures, evaluates reasons behind failures and repairs itself with minimal user involvement. The remainder of this document will outline the design, data sources, processing steps, model architectures, interpretable components and self-healing loop of the CloudSim Plus system.

A. Overall Architectural Layers

1) Data Collection Layer: The real-time signals are collected by this layer from three principal sources including: network logs of traffic patterns and protocol behaviour; CP and RAM amounts utilized to determine resource usage and cloud performance data to monitor VM telemetry and health of systems. The continuous stream helps to capture short-lived anomalies or performance spikes for subsequent analysis.

2) Data Preprocessing Layer: The process by which collected data are transformed into usable forms goes through five phases of systematic transformation. These phases include

1) Normalizing heterogeneous metric value ranges
2) Extracting identifiable features from original raw signal(s)
3) Handling missing data
4) Recognizing temporal patterns
5) Balancing the class skews that exist among the anomaly datasets

The results of this transformation process will produce a data model format that has reduced noise and enhanced quality of features.

3) Predictive Modeling Layer: This specific layer is equipped with various specialized parallel operating models, as follows.

- **Network Anomaly Detection**, includes an LSTM model for sequential pattern recognition, an MLP for feature-driven classification and an ensemble model combining both types of models in one for added robustness
- **Resource Prediction** makes use of a Temporal Convolution Network("TCN") which utilizes dilated causal convolution with multi-head attention to provide anticipatory resource demand forecasting as well as detection of potential capacity constraints.
- **Advanced Anomaly Classification** is accomplished by employing a LightGBM for low frequency classification of the target or minority class in addition to providing highly efficient real-time inference.

Each of these models produces a prediction along with a corresponding confidence score which forms the basis of any subsequent diagnostics or decisions to be made.

4) XAI (Explainable AI) Layer: SHAP Explanation: Feature-level explanations of all model predictions will use SHAP values to show which features made the greatest contribution towards each individual anomaly detection (class) and resource forecast prediction, allowing for full transparency as to how the model generated its prediction. In so doing, the

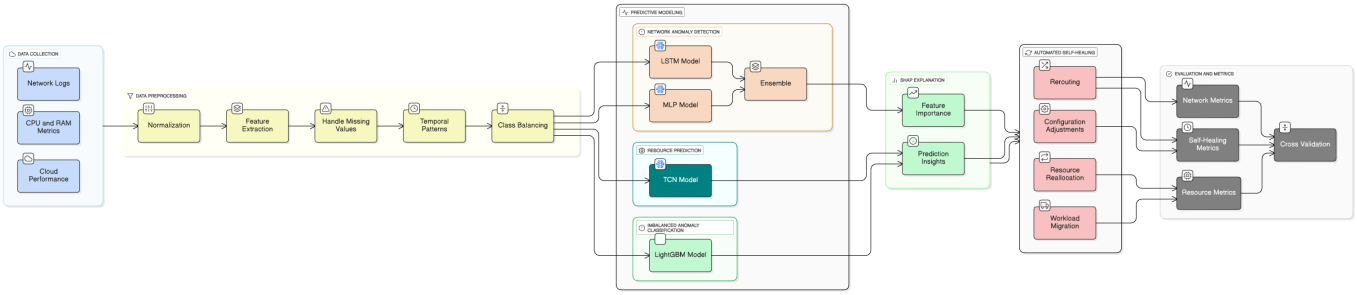

Fig. 1. Block diagram of the proposed self-healing framework

SHAP explanations will assist with a root cause analysis and provide guidance on the appropriate healing strategies to take by identifying the primary drivers of each prediction (feature).

5) Automated Self-Healing Layer: Whenever there are resource shortages or anomalies detected, this layer takes action to correct the problems found. These actions include:

- **Redirecting**: Reroutes network traffic around unreliable paths or congested links.
- **Changing Configurations**: Changes system configurations, updates firewall rules, and makes changes to minimize the issues detected.
- **Redistributing Resources**: Redistributes computing resources dynamically (computing resources can be added or removed) via vertical and horizontal scaling of VMs and by optimising workload placement.
- **Moving Workloads**: Moves services between degraded or overloaded hosts and healthy nodes to maintain service availability.

All actions taken are logged with their timing and outcome metrics.

6) Evaluation and Metrics Layer: After healing, the system gathers all types of performance measurements in the following areas:

- **Network Metrics**: Packet loss rates, latency distribution, throughput variability, and connection reliability.
- **Performance Metrics**: Accuracy in detection, false positive rates, successful healing, and model inference times.
- **Resource Metrics**: CPU use, memory use, energy consumption, and VM lifecycle statistics.

The system uses this information to refine its algorithms through iterative processing and validate the effectiveness of the healing process.

7) End-to-End Workflow: The complete process starts with data collection, goes through data cleansing, predictive model layer (where multiple models can generate a prediction), and then it goes to SHAP-based explanations before executing automated healing actions. The final step is reporting. Once the system starts all of these steps in a continuous loop and is able to adjust to changing workload conditions and failure patterns it creates a closed loop self-improving architecture.

B. CloudSim Plus Environment

The primary test platform to evaluate the pipeline is CloudSim Plus. Custom simulations were developed for generating workloads, VMs was allocated, and controlled failures were injected after configuring JDK, Maven and the required libraries. The modular design of CloudSim Plus allows detailed control of the behaviour of hosts through their scheduling policies and event-driven behaviour of the resources. This enables realistic resource fluctuations, link outages and stress events to be reproduced. The Cloudsim Plus simulator is also deterministic, which gives the training and validation of the entire detection, forecast and healing models a clear ground truth.

C. Dataset Foundation

The three data sets used in the framework capture three different perspectives of behavior seen in both the cloud and the network:

- **CICIDS 2017**: This is a complete dataset of network traffic, including a variety of attack scenarios such as Denial of Service (DDoS) attacks, infiltration, brute-force attempts, botnet activity, etc. This dataset is helpful for both multi-class attack recognition as well as evaluating the robustness of an attack detection system.
- **NSL-KDD-derived dataset**: This dataset is well-balanced, making it useful for binary anomaly detection. Its structure allows it to be used for temporal modeling with LSTMs, without severe class imbalance problems.
- **Kaggle cloud performance dataset**: This dataset contains time series of monitoring information from the CPU, RAM, and power from the cloud. These time series can be utilized for forecasting future resource usage.

All three datasets can be utilized to identify malicious behavior occurring on either or both systems, identify how the performance of either or both systems has degraded, and forecast any potential shortage of resources.

D. Data Preparation Pipeline

The datasets that are being used have various structures, levels of granularity and formats meaning there is a need for a unified preprocessing workflow to have all datasets prepared similarly. **CICIDS 2017**: CICIDS 2017 was cleaned, imputed,

encoded using labels, and reduced in terms of its features using Random Forest based ranking for feature reductions. A stratified train/test split allowed for the class imbalance to remain intact. **NSL-KDD**: All categorical features in NSL-KDD were converted to numerical data and scaled appropriately (i.e., Min-Max scaling). Each record of data was converted to pseudo-sequence records, into a fixed length sequence of records, and thus LSTMs may now recognize how network behaviours can change over time. **Cloud performance telemetry**: A sliding window technique was used to normalize, interpolate and segment cloud performance telemetry data into sequences to build TCNs. All chronological order records were maintained when creating lagged features and prior to chronological splitting. Thus this pipeline produces a standardised foundation of features which allows for an effective learning process when developing training/testing on all models.

E. Model Architecture

There is more than one model within this single system working simultaneously to perform a specific role each.

1) MLP Classifier: Uses dense layers with rectified linear units (ReLU), batch normalization and dropout to process a set of selected features from the NSL-KDD dataset for feature based anomaly detection but does not deal directly with feature based anomaly detection.

2) LSTM Network: Stacks LSTM layers to capture the temporal (time based) transitions from the sequentialized data flow in order to build a more accurate model and use Adam optimization technique and binary cross-entropy loss function for optimization.

3) LSTM–MLP Hybrid Ensemble: Both models work separately and the average of the probability outputs from the models is taken to enhance stability and decrease potential overfitting

4) Temporal Convolutional Network (TCN): Utilized for forecasting of cloud resources, it uses dilated causal convolutions (DCC) to allow long-range temporal dependencies to be learned as well as utilizing the module known as multi-head attention to focus on relevant temporal segments of the model.

5) Decision Tree and LightGBM Baselines: A very basic and simple version of a decision tree gives an easy to understand baseline for the accurate detection of anomalies after finding this simple measurement both LightGBM can improve detection of an anomaly and offers real-time fast performance for detecting anomalies especially in the minority class.

F. Explainability Layer

A SHAP module evaluates feature contributions for predictions produced by the MLP, LSTM, ensemble, LightGBM and TCN models. SHAP values highlight influential signals behind each anomaly or performance deviation, helping identify likely root causes and guiding the choice of healing actions.

G. Simulation and Self-Healing Environment

The full self-healing loop runs inside CloudSim Plus. The simulator manages a datacenter with configurable hosts, VMs and cloudlets. It supports:

- failure injection
- workload surges
- resource saturation
- dynamic VM scaling

Metrics from the execution environment are routed to the predictive layer. When a model detects an anomaly or forecasts a resource shortage, SHAP explanations are inspected and CloudSim Plus triggers recovery actions, including VM migration, horizontal or vertical scaling, workload reallocation or flow isolation. Recovery time, detection latency and service availability are recorded for evaluation.

H. Summary of the Architecture

The system functions as a closed-loop process:

1) CloudSim Plus generates workloads, telemetry and injected failures.
2) Data is processed through the unified preprocessing pipeline.
3) Models generate anomaly and resource predictions.
4) SHAP extracts explanations for abnormal outcomes.
5) CloudSim Plus executes corrective actions.
6) Performance metrics are logged to refine the system.

This design forms the foundation of a resilient, adaptive and explainable self-healing cloud environment.

V. Results and Discussion

This chapter presents the evaluation of all models used in the proposed self-healing framework. The analysis covers anomaly detection (MLP, LSTM and Ensemble), multi-class attack classification (Decision Tree and LightGBM), resource forecasting with the TCN model and interpretability assessment using SHAP.

A. Role of CloudSim Plus in Experimental Validation

CloudSim Plus was used to generate synthetic workloads, resource fluctuations and fault events. The output allowed us to recreate realistic behaviours while keeping the environment controlled. Figure 2 displays a sample output from the CloudSim Plus simulator and represents the type of workload traces used to train and validate the models.

B. Neural Network-Based Anomaly Detection Models

MLP, LSTM and Ensemble models were trained using features derived from the NSL-KDD dataset. Table I provides a summary of the classification performance, which includes a direct comparison between classification accuracies, precision, recall, and F1 score.

1) Confusion Matrices: Figures 3–5 illustrate the performance of each model on normal and anomaly samples and display errors that verify class balance, false alarms and missed detections.

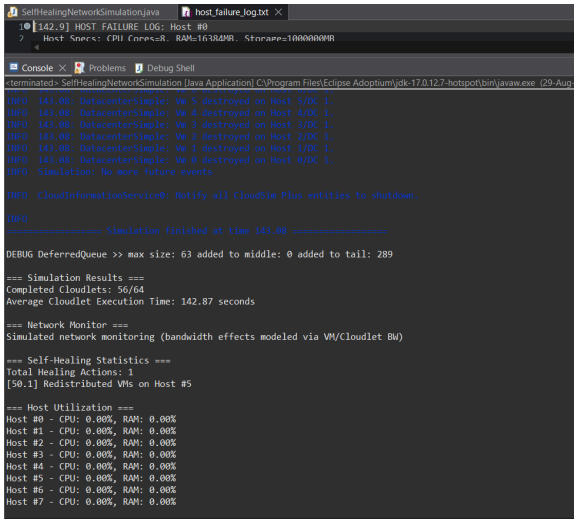

Fig. 2. Example CloudSim Plus workload and anomaly generation.

TABLE I
PERFORMANCE OF MLP, LSTM AND ENSEMBLE MODELS

Metric	MLP	LSTM	Ensemble
Accuracy	98.51%	97.08%	98.27%
Precision	98.48%	97.12%	98.25%
Recall	98.54%	97.03%	98.29%
F1-Score	98.51%	97.07%	98.27%

C. Comparison of ROC–AUC Curves

Illustrated in Figures 6, 7 and 8 are the ROC-AUC Curves for each of these three Anomaly Detection Models. These ROC-AUC Curves indicate how well each of these models can separate normal from attack traffic at various threshold settings.

D. Decision Tree vs LightGBM

Table II shows the performance of the Decision Tree on Attack Classification; and Figure 9 provides a visual representation of misclassifications by Attack Class in the form of a confusion matrix.

TABLE II
DECISION TREE PERFORMANCE

Accuracy	99.70%
Precision	74.97%
Recall	75.18%
F1-Score	74.74%

LightGBM performance metrics are shown in Table III. The confusion matrix (Figure 10) shows that LightGBM has a high degree of classification consistency compared to that of XGBoost.

Figures 11 and 12 compare the receiver operating characteristic (ROC) curves of both LightGBM and XGBoost. The ROC curves show that LightGBM is the better classifier when using the AUC metric.

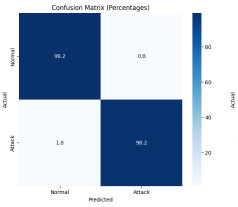

Fig. 3. MLP Confusion Matrix

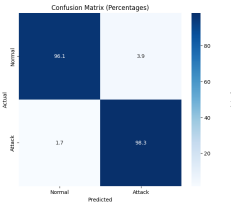

Fig. 4. LSTM Confusion Matrix

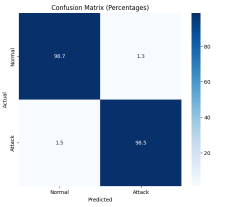

Fig. 5. Ensemble Confusion Matrix

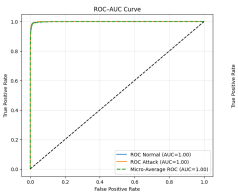

Fig. 6. MLP ROC–AUC

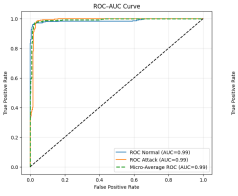

Fig. 7. LSTM ROC–AUC

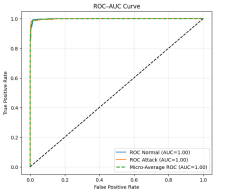

Fig. 8. Ensemble ROC–AUC

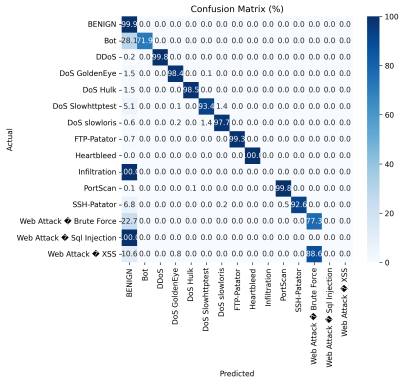

Fig. 9. Confusion Matrix of Decision Tree.

TABLE III
LIGHTGBM PERFORMANCE (OVERALL METRICS)

Accuracy	99.84%
Precision	89.88%
Recall	92.61%
F1-Score	91.05%

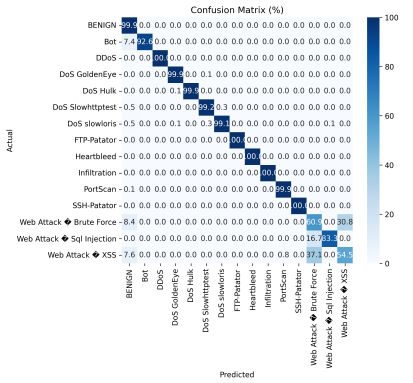

Fig. 10. Confusion Matrix of LightGBM.

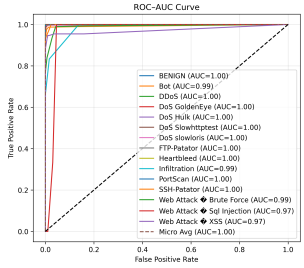

Fig. 11. Decision Tree ROC–AUC

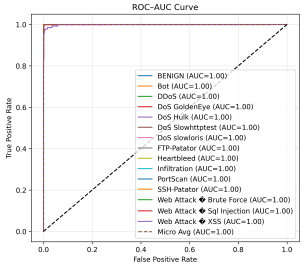

Fig. 12. LightGBM ROC–AUC

E. TCN for Resource Forecasting

The prediction metrics for CPU, RAM, and total load forecasts are provided in Table IV. The values show how well the TCN represents temporal behavior of system stress patterns, while Figure 13 shows how closely the predicted and actual resource usage curves follow the actual trend.

TABLE IV
TCN EVALUATION METRICS

Metric	CPU	RAM	Combined
MSE	208.14	212.73	210.43
MAE	9.06	10.02	9.54
RMSE	14.43	14.59	14.50
R^2	0.6289	0.6203	0.6246
Corr.	0.7934	0.7942	0.7913

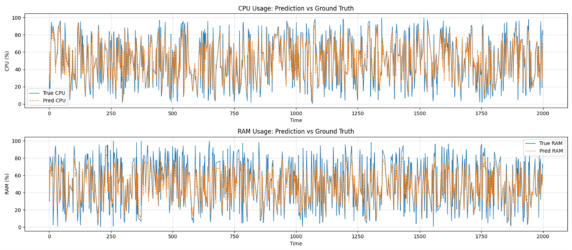

Fig. 13. Predicted vs actual CPU and RAM usage from TCN model.

F. SHAP-Based Interpretability

The SHAP generated outputs are displayed in Figure 14, Figure 15 and Figure 16; the force plot shows contributions to one prediction, the waterfall plot shows contributions of all predictions made for an entire data set, and the summary plot shows feature importance in terms of the entire data set.

Fig. 14. SHAP Force Plot

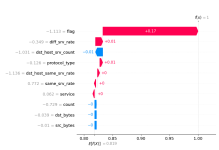

Fig. 15. SHAP Waterfall

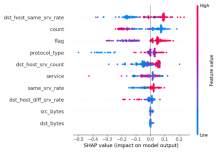

Fig. 16. SHAP Summary Plot

G. Summary

The framework produced consistently high detection rates, accuracy of temporal forecasts, and meaningful interpretability across all assessments. The framework demonstrated an ability to detect anomalies, classify attacks, forecast resource stress, and aid in self repairing autonomously within a simulated cloud environment.

VI. CONCLUSION

This project developed an automated self-healing system for cloud and virtualized networks that uses anomaly detection, predictive modeling and interpretable machine learning methods to find faults before they cause any real downtime or problems with network performance. From a technical perspective, the results clearly demonstrate that the combination of these three complementary techniques yields greater reliability and lowers mean time to recover from faults than do traditional reactive fault-management strategies. The MLP, LSTM and Ensemble algorithms all exhibited good performance in terms of detecting anomalies while also providing dependable results. The LightGBM classifier had improved capability to handle small amounts of anomalous data (minority class instances) compared to previous work due to the fact that previous works did not adequately address imbalance issues when evaluating their models. The TCN model produced very accurate predictions of CPU and RAM usage, which allowed for proactive remedies prior to experiencing resource saturation. By providing SHAP explanations for all models, we were able to provide insight into which features influenced each model's decision. CloudSim Plus was used to conduct workload simulations and create fault scenarios in a controlled manner. Ultimately, our results reinforce the belief that the combination of predictive analytics with interpretable automation increases the resiliency of cloud networks and enhances the scalability and usability of automatic self-healing of cloud networks.

VII. FUTURE WORKS

Future improvements can enhance the ability of the proposed framework to adapt, scale, and work in real-world settings. CloudSim Plus provided a testing environment that was tightly controlled, but deploying the proposed system onto real-world platforms (OpenStack, Kubernetes and Software Defined Networking testbeds) will give us the ability to validate the proposed system against real-world workloads, multi-tenant interference and containerized architecture. Reinforcement learning can also be leveraged to extend the rule-based self-healling mechanism for optimizing autoscaling, migration and routeing strategies through long-term performance improvement rules. Continual learning and drift detection may help maintain the accuracy of the introspective model in response to changing cloud-based workloads. The addition of storage I/O activity, micro-service traces and hardware performance counters sources will also help improve the accuracy of anomaly predictions and thus improve the accuracy of healing actions taken. As system size increases, there are opportunities

to use a hierarchical or distributed controller to coordinate self-healing decisions made across nodes. The explainability layer will also be improved with temporal attribution analysis and operator-centred visualisation dashboards. Increased robustness against adversarial attacks such as model poisoning and evasion attacks is another important area for future research. Finally, multi-objective optimisation approaches may be used in future versions of the proposed framework) to balance the energy consumed, cost to operate, and performance provided by those platforms and help enable sustainable self-healing of large-scale cloud computing platforms.

REFERENCES

[1] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1), 23–50.

[2] Elhabbash, A. S., Zahid, M. A. H., & Elmoghazy, A. M. (2022). Simulation-based network fault injection in the CloudSim Plus cloud simulation environment. *Applied Journal of High-Performance Computing*, 2(1), 67–75.

[3] Perumallaplli, R. (2015). Self-healing networks: An AI approach to network fault management. *SSRN Electronic Journal*. Retrieved from https://papers.ssrn.com/abstract=5228591

[4] Panigrahi, R., & Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology*, 7, 479–482.

[5] Ribeiro, M., Singh, S., & Guestrin, C. (2016). Why Should I Trust You? Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144).

[6] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 30.

[7] Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

[8] Kumar, N., Groenewald, E., Kulkarni, S., & Km, A. (2023). Self-healing networks: AI-based approaches for fault detection and recovery. *Power System Technology*, 47, 371–386.

[9] Farmani, J., & Zadeh, A. K. (2023). AI-based self-healing solutions applied to cellular networks: An overview. *arXiv preprint arXiv:2311.02390*.

[10] Katragadda, S., & G., D. (2023). Self-Healing Networks: Implementing AI-Powered Mechanisms to Automatically Detect and Resolve Network Issues with Minimal Human Intervention. *International Journal of Scientific Research and Engineering Development*, 6(6), 977–987.

[11] Khraisat, M., Alazab, A., & Gondal, S. (2023). A hybrid intrusion detection model using ensemble learning. *Complex & Intelligent Systems*, 9, 1235–1248.

[12] Rajesh, M., Kumar, T. G., & Kumar, V. (2023). A self-healing framework for cloud network resilience using AI and fault modeling. *Migration Letters*, 20(5), 45–56.

[13] Wang, T., Zhang, Q., & Liu, W. (2021). Fault prediction and prevention in cloud systems using machine learning. *Future Generation Computer Systems*, 125, 732–743.

[14] Raziq, A. R. A. (2024). Cloud computing performance metrics dataset. Kaggle Datasets. Retrieved from https://www.kaggle.com/datasets/abdurraziq01/cloud-computing-performance-metrics

[15] Tuli, M., Tuli, S., Buyya, R., & Dustdar, S. (2021). EdgeLens: Deep Learning-based anomaly detection in Edge Computing. *IEEE Internet of Things Journal*, 8(13), 10359–10368.

[16] Fahim, F. A., Alam, S., & Iftekhar, K. M. (2022). An explainable AI approach for cloud failure prediction using deep learning. *IEEE Access*, 10, 87510–87523.

[17] Nobre, J., Granville, L. Z., & Clemm, A. (2019). Self-healing and SDN: Bridging the gap. *ICT Express*, 5(1), 3–14.

[18] Sujatha, D., Raj, M. T. F., Ramesh, G., Agoramoorthy, M., & A. A. S. (2024). Neural Networks-Based Predictive Models for Self-Healing in Cloud Computing Environments. In *International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, 1(1), 1–4.