

Anomaly Detection in Cloud API Access Patterns Using Temporal Convolutional Networks

Narendra Kumar*

Department of CSE, Amity University Jharkhand, Ranchi - 835303, Jharkhand, India

E-mail: narendra298@gmail.com

ORCID iD: <https://orcid.org/0000-0002-0295-9579>

*Corresponding author

D. Lakshmi Padmaja

School of Engineering, Anurag University, Hyderabad - 500088, India

Email: lakshmipadmajait@anurag.edu.in

ORCID iD: <https://orcid.org/0000-0001-8262-6821>

M. Rajanidevi

Department of ECE, Koneru Lakshmaiah Education Foundation, Vaddeswaram - 522302, Andhra Pradesh, India

E-mail: rajimerigala@gmail.com

ORCID iD: <https://orcid.org/0000-0001-8163-9874>

Dasaradha Ramayya Lanka

Department of AIML, Aditya University, Surampalem-533437, Andhra Pradesh, India

E-mail: dasaradh.phd@gmail.com

ORCID iD: <https://orcid.org/0009-0009-2519-2663>

A. Mahendar

Department of CSE, CMR Technical Campus, Hyderabad - 501401, Telangana, India

E-mail: mahi.adapa@gmail.com

ORCID iD: <https://orcid.org/0000-0002-7727-0290>

V. Gokula Krishnan

Department of CSE, Easwari Engineering College, Chennai - 600089, Tamil Nadu, India

E-mail: gokul_kris143@yahoo.com

ORCID iD: <https://orcid.org/0009-0005-6819-6729>

Received: March 7, 2026; Revised: April 14, 2026; Accepted: May 25, 2026; Published: October 8, 2026

Abstract: Cloud platforms generate massive API access logs, where abnormal patterns may indicate security breaches, insider threats, or compromised credentials, demanding intelligent and automated anomaly detection mechanisms. Conventional approaches employ segmentation, statistical profiling, clustering, recurrent networks, and supervised classifiers to model sequential API behavior and distinguish normal activities from malicious deviations. These techniques generally achieve high detection accuracy and improved threat visibility, enhancing cybersecurity monitoring systems while reducing manual auditing efforts in large-scale distributed cloud environments. However, they struggle with evolving attack patterns, high false-positive rates, limited temporal dependency modelling, data imbalance, and poor generalization across heterogeneous cloud infrastructures. This study proposes a self-supervised Temporal Convolutional Network with adaptive anomaly scoring, achieving robust sequential modelling, reduced false alarms, and improved detection stability in cloud APIs. A self-supervised Temporal Convolutional Network models sequential API behavior using causal dilated convolutions and adaptive scoring, enabling accurate, scalable, and real-time cloud anomaly detection.

Index Terms: Cloud API Security, Anomaly Detection, Temporal Convolutional Networks (TCN), Self-Supervised Learning, Access Pattern Analysis, Intrusion Detection Systems.

1. Introduction

Cloud computing has become the backbone of modern digital services, enabling scalable, distributed, and on-demand access to applications and data. Within these environments, Application Programming Interfaces (APIs) serve as critical gateways that facilitate communication between services, users, and third-party systems. As organizations increasingly rely on cloud APIs for authentication, data exchange, and service orchestration, monitoring API access behavior has become essential for ensuring system integrity and security [1]. However, the dynamic and high-volume nature of cloud environments introduces significant challenges in identifying abnormal access patterns that may indicate misuse, credential compromise, insider threats, or automated attacks. Traditional security mechanisms such as rule-based monitoring and signature detection struggle to adapt to evolving behavior and previously unseen attack patterns. Consequently, anomaly detection techniques have emerged as a promising approach for identifying deviations from normal access behavior. By analyzing temporal access sequences and behavioral patterns, anomaly detection systems can enhance situational awareness and enable early threat mitigation. Detecting anomalies in cloud API access patterns is therefore a critical research area, contributing to improved cybersecurity resilience, operational reliability, and proactive defense strategies in increasingly complex cloud ecosystems. Anomaly detection in cloud API access patterns typically involves a pipeline that combines data preprocessing, segmentation, feature engineering, and classification or sequence modelling [2]. Log parsing and timestamp alignment are first applied to structure raw API access logs into temporally ordered sessions. Segmentation techniques, such as sliding windows or session-based partitioning, divide continuous streams into meaningful behavioral segments that preserve temporal context. Feature engineering then transforms raw events into numerical representations through encoding schemes, temporal feature extraction, frequency statistics, and behavioral aggregation [3]. For classification and pattern recognition, both traditional machine learning and deep learning approaches are widely employed. Classical methods include Support Vector Machines, Random Forests, and clustering-based models that distinguish normal from abnormal behavior. More recently, deep learning architectures such as recurrent neural networks (RNNs), Long Short-Term Memory networks (LSTMs), autoencoders, and Temporal Convolutional Networks (TCNs) have been used to capture sequential dependencies and temporal dynamics. Self-supervised learning and next-event prediction frameworks further enhance representation learning by modelling expected behavioral sequences. These frameworks collectively enable systems to learn normal behavioral patterns and detect deviations in complex, high-dimensional cloud environments. Existing anomaly detection approaches for cloud access monitoring have demonstrated promising results in identifying suspicious behavior and improving security visibility. Sequence-based deep learning models have shown strong capability in capturing temporal dependencies, enabling detection of subtle deviations that rule-based systems often miss. Studies commonly report improved detection accuracy, reduced false positives, and enhanced sensitivity to previously unseen anomalies when temporal modelling techniques are employed. Sliding window segmentation and session-based analysis have been observed to improve contextual understanding, allowing models to recognize behavioral transitions rather than isolated events. Feature fusion strategies further enhance performance by integrating temporal, frequency, and behavioral attributes into unified representations [4]. In practical deployments, anomaly scoring mechanisms and adaptive thresholds enable continuous risk assessment, allowing security teams to prioritize high-risk events and respond proactively. Additionally, real-time monitoring frameworks have demonstrated improved incident response times and reduced manual inspection effort. Overall, research trends indicate that integrating temporal learning with automated scoring and alerting mechanisms enhances detection robustness and operational efficiency in dynamic cloud environments [5]. Despite significant progress, anomaly detection in cloud API access patterns faces several persistent challenges. One major limitation is the scarcity of labeled anomaly data, which complicates supervised training and evaluation. Behavioral variability across users, services, and workloads also makes it difficult to define a stable baseline of normal activity. Many models suffer from high false positive rates when exposed to evolving usage patterns, leading to alert fatigue and reduced operational trust. Sequence-based deep learning methods, while powerful, can be computationally intensive and difficult to interpret, limiting transparency and deployment in resource-constrained or compliance-sensitive environments. Segmentation strategies such as fixed sliding windows may fail to capture long-range dependencies or may fragment meaningful behavioral sequences. Additionally, adaptive attackers can mimic legitimate access patterns, reducing the effectiveness of purely behavioral detection [6]. Data privacy concerns and the sensitivity of access logs further complicate data sharing and benchmarking efforts. These challenges highlight the need for more robust self-supervised learning strategies, interpretable anomaly scoring mechanisms, and adaptive models capable of handling evolving cloud access behaviors. This research proposes a self-supervised anomaly detection framework for cloud API access patterns that integrates temporal segmentation, deep sequence modelling, and continuous anomaly scoring. The method begins by preprocessing API logs to construct temporally ordered sessions, followed by sliding-window segmentation to preserve behavioral context. Encoded event sequences and extracted temporal features are fused to form enriched representations of user access behavior. A Temporal Convolutional Network (TCN) is employed to model sequential dependencies and learn expected behavioral patterns through a next-event prediction objective [7]. By leveraging causal dilated convolutions and residual connections, the model captures both short- and long-range temporal relationships while maintaining computational efficiency. The network produces a continuous anomaly score representing deviations from learned normal behavior. This score is refined through

normalization and adaptive thresholding to support dynamic risk assessment. Post-processing mechanisms, including score smoothing and alert aggregation, enhance stability and reduce false alarms. The proposed approach offers improved detection sensitivity, scalability for high-volume cloud environments, and reduced reliance on labeled anomaly data. By combining self-supervised learning with temporal modelling, the framework aims to provide a robust and practical solution for real-time cloud security monitoring. This paper is structured to present the problem and solution in a clear sequence. The **Introduction** outlines the importance of detecting anomalies in cloud API access. The **Literature Survey** reviews existing detection techniques and their limitations. The **Proposed Method** explains the TCN-based framework with temporal segmentation and anomaly scoring. **Results and Discussion** evaluate performance and practical impact. The **Conclusion** summarizes contributions and future scope, followed by **References** supporting the study.

1.1. Main Contributions and Novelty

The primary novelty of the proposed framework lies in the integration of self-supervised temporal sequence modelling with adaptive anomaly scoring for cloud API access monitoring. Unlike existing recurrent and transformer-based anomaly detection systems, the proposed method employs a Temporal Convolutional Network (TCN) with causal dilated convolutions to efficiently capture both short-range and long-range behavioural dependencies while supporting parallel computation and reduced inference latency.

The major contributions of this work are summarized as follows:

- A self-supervised anomaly detection framework is proposed for cloud API access logs, eliminating dependence on large-scale labelled anomaly datasets.
- A TCN-based temporal modelling architecture is introduced to learn sequential API behaviour using causal dilated convolutions and residual learning, enabling efficient modelling of long-range dependencies compared with conventional LSTM-based approaches.
- An adaptive anomaly scoring and thresholding mechanism is developed to dynamically adjust detection sensitivity according to behavioural variations in cloud workloads.
- A continuous anomaly scoring framework with score smoothing and alert aggregation is designed to reduce false alarms and improve operational stability in real-time monitoring environments.
- Extensive evaluation on benchmark LogHub datasets demonstrates improved ROC-AUC, PR-AUC, and false positive rate performance compared with recent deep learning and generative anomaly detection methods.

2. Literature Survey

The rapid expansion of cloud computing and distributed services has increased reliance on secure and reliable API interactions, making anomaly detection in cloud access patterns a critical research area. As cyber threats grow in sophistication, researchers have explored diverse methodologies ranging from statistical modelling to deep learning and self-supervised techniques to identify abnormal behaviors. This growing body of work reflects significant methodological diversity and continuous innovation. A comprehensive literature survey is essential to synthesize these developments, evaluate existing approaches, and identify persistent challenges. By examining current progress and research gaps, the survey provides a foundation for advancing more robust and adaptive anomaly detection solutions.

Graph-Based and Structural Representation Learning Approaches Graph-oriented anomaly detection methods transform log or cloud activity into structured relationships to capture behavioral dependencies. Alaca et al. [8] propose GLSTM, converting heterogeneous logs into graph representations using Node2Vec before applying LSTM/GRU networks for anomaly detection, achieving strong accuracy but facing scalability and data diversity challenges. Marbel et al. [9] introduce CS-GAD, a dynamic tripartite graph modelling users, services, and actions, combined with embedding and distance-based scoring to detect malicious cloud activities behaviorally; however, performance is sensitive to time-window selection and detection latency trade-offs. These graph-based approaches excel at modelling relationships and contextual dependencies that sequential models may overlook. They improve detection accuracy and enable early identification of coordinated or multi-entity attacks. Nevertheless, their effectiveness depends heavily on graph construction quality, embedding choices, and computational cost for large-scale environments. Additionally, evolving system structures may require frequent retraining or graph updates, complicating deployment in rapidly changing cloud infrastructures.

Generative and Diffusion-Based Deep Learning Methods Generative modelling approaches detect anomalies by learning data distributions and identifying deviations through reconstruction or density estimation. Saravanan et al. [10] propose TSI-GAN, transforming time-series signals into image representations and using cycle-consistent GAN reconstruction to detect anomalies, achieving strong generalization but requiring careful handling to avoid overfitting reconstruction behavior. Lee et al. [11] present Maat, combining conditional diffusion forecasting with isolation forests to anticipate anomalies before occurrence, though forecasting sudden spikes and generalizing beyond cloud metrics remain challenges. Lim et al. [12] introduce MadSGM, a score-based generative framework combining reconstruction, density, and gradient-based anomaly measures with adversarial purification to stabilize predictions. These generative approaches provide robust modelling of complex temporal distributions and

enable early detection or anticipation. However, they are computationally intensive, sensitive to training stability, and may struggle with abrupt behavioral shifts or unseen anomaly patterns. Their deployment in real-time systems may require optimization and careful calibration to balance accuracy with inference speed. Transformer-Based and Self-Supervised Log Language Modelling Recent research leverages natural language processing and self-supervised learning to detect anomalies directly from raw logs without heavy preprocessing. Yamanaka et al. [13] propose LogELECTRA, using replaced token detection to identify contextual deviations in log messages, reducing parser dependency but experiencing slight performance drops under chronological training and scalability concerns. Pospieszny et al. [14] introduce ADALog, which fine-tunes DistilBERT using masked language modelling and adaptive thresholds to classify anomalies, achieving strong performance but requiring substantial computational resources and limiting real-time deployment. Lu et al. [15] propose TPLogAD, combining BERT-based template embeddings and parameter vectorization with BiLSTM attention to capture semantic relationships, improving detection precision while handling dynamic log structures. These methods excel in understanding semantic log patterns and adapting to unstructured data formats. However, transformer-based models demand significant computational power, may face latency issues, and often require careful domain adaptation to maintain performance across diverse systems. Statistical, Hybrid, and Forecasting-Based Detection Frameworks Hybrid and statistical approaches emphasize lightweight computation, interpretability, and adaptability to evolving systems. Liu et al. [16] propose ScaleAD, a trie-based log parsing framework combined with Extreme Value Theory and expert feedback to detect anomalies efficiently; however, EVT assumptions may fail during anomaly bursts, and semantic ambiguity can mislead expert systems. Al-Ghuwairi et al. [17] introduce a feature-selection pipeline with Prophet forecasting and causality analysis to detect intrusion precursors, reducing computational cost but requiring broader validation across datasets and models. These frameworks offer faster processing, lower resource consumption, and improved interpretability compared to deep learning methods. They are particularly suitable for real-time monitoring and resource-constrained environments. Nevertheless, their effectiveness may decline with complex behavioral patterns, evolving log semantics, or non-stationary time-series dynamics. Ensuring robustness across diverse cloud infrastructures and minimizing reliance on expert feedback remain ongoing research challenges.

2.1. Research Gap Analysis

- Existing anomaly detection approaches for cloud environments exhibit several limitations despite recent advances in deep learning and self-supervised modelling. Graph-based frameworks effectively capture structural dependencies but often suffer from scalability challenges and high computational overhead in dynamic cloud infrastructures. Transformer-based methods provide strong semantic understanding of log sequences; however, their large computational requirements and inference latency restrict real-time deployment. Generative and diffusion-based techniques improve distribution modelling but may experience instability during training and reduced robustness under rapidly evolving workloads.
- Most existing studies also rely on static thresholding strategies, which are insufficient for continuously changing cloud environments where access patterns evolve dynamically. In addition, several approaches require extensive labelled anomaly data or expensive retraining procedures to maintain performance across heterogeneous systems.
- These limitations highlight the need for a lightweight, scalable, and adaptive anomaly detection framework capable of modelling long-range temporal dependencies while maintaining low false positive rates and reduced dependence on labelled datasets. The proposed self-supervised TCN framework addresses these challenges through causal dilated convolutions, adaptive anomaly scoring, and continuous temporal modelling.

3. Proposed Method

This figure illustrates the end-to-end architecture, including preprocessing, feature engineering, self-supervised task design, TCN-based temporal modelling, continuous anomaly scoring, and alert generation. The framework enables scalable and real-time detection of anomalous cloud API behaviors. The proposed framework (Fig. 1) presents an end-to-end pipeline for detecting anomalous cloud API access patterns using temporal modelling and self-supervised learning. The system is designed to transform raw log data into meaningful behavioral representations, learn normal access patterns, and generate continuous anomaly scores that support real-time security monitoring and alert generation. This architecture enables scalable, adaptive detection suitable for dynamic cloud environments where behavioral patterns evolve over time. The pipeline begins with pre-processing, where raw log data is parsed, temporally ordered, and grouped into sessions. This stage ensures structural consistency and preserves behavioral continuity required for sequential modelling. The processed sessions are passed to the feature engineering module, which encodes events, extracts temporal attributes such as inter-arrival time and frequency patterns, and fuses multiple feature types into a unified representation. This enriched feature space enhances the model's ability to capture contextual and temporal dependencies. Next, the self-supervised task design module constructs sliding windows over event sequences and formulates a next-event prediction objective. By learning to predict subsequent events, the system captures normal behavioral patterns without requiring labeled anomalies. Loss computation guides representation learning by penalizing deviations between predicted and actual events. The encoded sequences are then processed by a Temporal

Convolutional Network (TCN), which employs causal dilated convolutions and residual blocks to model both short- and long-range temporal dependencies while maintaining computational efficiency. The TCN produces a continuous anomaly score representing deviations from learned normal behavior. This score enters the continuous anomaly scoring mechanism, where prediction error computation, normalization, and adaptive thresholding transform raw scores into interpretable risk levels. The adaptive threshold allows dynamic sensitivity adjustment based on system behavior. Subsequently, the post-processing and alert generation module smooths scores, aggregates alerts across time windows, and assigns severity levels to prioritize responses. Finally, the performance evaluation block assesses detection effectiveness using ROC-AUC, PR-AUC, F1-score, and false positive rate metrics. The modular design improves scalability, interpretability, and robustness. Parallel feature fusion enhances representation richness, while adaptive thresholding enables responsiveness to evolving workloads. The architecture supports real-time monitoring and reduces false alarms through score smoothing and aggregation.

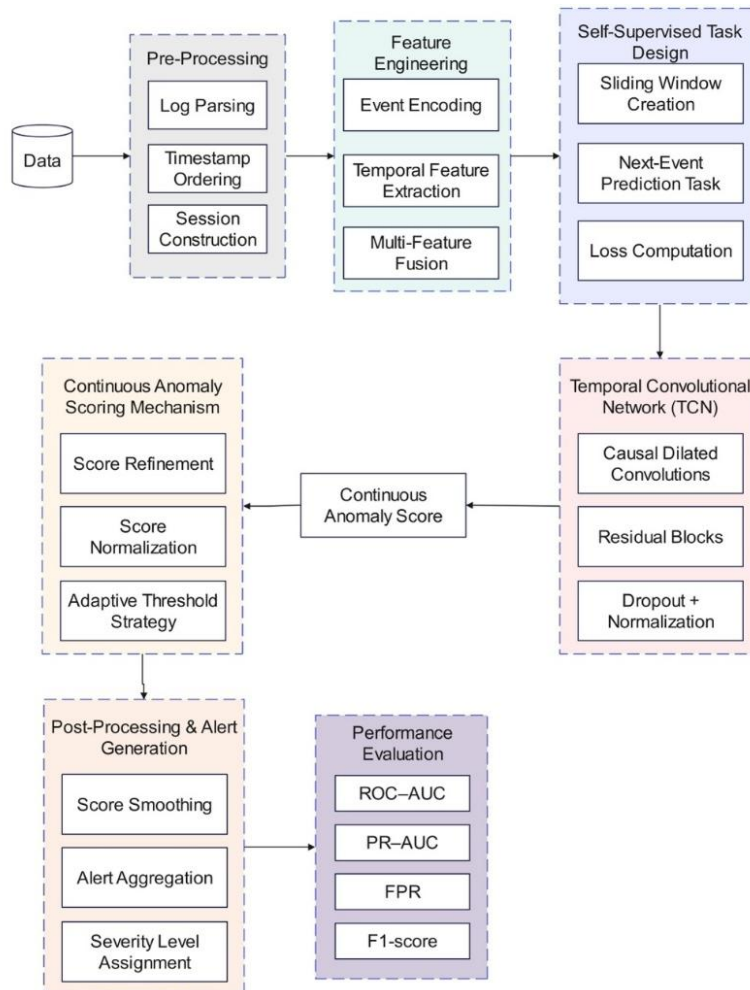


Fig. 1. Proposed Framework for Anomaly Detection in Cloud API Access.

The pre-processing module converts raw cloud API logs into structured sequences suitable for temporal modelling. Its primary inputs are unstructured or semi-structured log records containing timestamps, event types, user identifiers, and resource interactions. The module performs **log parsing** to extract relevant fields and standardize event templates. **Timestamp ordering** ensures chronological consistency, which is essential for sequence learning and temporal dependency modelling. Following ordering, **session construction** groups events into logical sessions based on user identity, time gaps, or interaction context. This grouping preserves behavioral continuity and prevents fragmentation of activity patterns. By transforming raw logs into ordered sessions, the module reduces noise and structural variability, enabling downstream modules to focus on behavioral patterns rather than data inconsistencies. This stage also reduces dimensional complexity and ensures reproducibility. The output is a set of temporally coherent event sequences ready for encoding and feature extraction. Without this stage, sequential models may misinterpret event order, resulting in degraded detection accuracy. The feature engineering module transforms structured sessions into numerical representations that capture behavioral and temporal characteristics. Inputs include ordered event sequences produced during preprocessing. **Event encoding** converts categorical events into vector representations using indexing or embedding schemes. **Temporal feature extraction** computes attributes such as inter-event time, frequency, duration,

and periodicity, enabling detection of unusual timing patterns. **Multi-feature fusion** integrates categorical embeddings, temporal statistics, and behavioral indicators into a unified feature vector. This fusion enriches contextual understanding by combining semantic and temporal information. The resulting feature representation improves the model's ability to differentiate between normal variability and suspicious deviations. The module also reduces sparsity and improves generalization by integrating heterogeneous features into a consistent representation space. Outputs are multivariate temporal feature sequences that serve as input to the self-supervised learning stage. The self-supervised learning module enables the system to learn normal behavioral patterns without labeled anomaly data. Using sliding window segmentation, continuous event sequences are divided into overlapping subsequences that preserve temporal context. Each window forms an input sequence, while the subsequent event serves as the prediction target. This **next-event prediction task** trains the model to learn normal sequence transitions and dependencies. Loss computation evaluates the difference between predicted and actual events, guiding the learning process toward accurate behavioral modelling. This formulation allows the system to capture contextual patterns, transition probabilities, and sequential regularities inherent in normal operations. Because the model is trained only on normal data, deviations from predicted behavior become indicative of anomalies. This self-supervised approach improves scalability and adaptability while eliminating the need for costly labeled datasets. It also enhances robustness against evolving behavior continuously learning from recent patterns. The Temporal Convolutional Network serves as the core sequence modelling component. It processes encoded sliding windows to learn temporal dependencies and generate anomaly scores. The TCN employs **causal convolutions**, ensuring predictions depend only on past events, making the model suitable for real-time monitoring. **Dilated convolutions** expand the receptive field exponentially, enabling the network to capture long-range dependencies without deep architectures. **Residual blocks** stabilize training and support deeper network structures, while dropout and normalization improve generalization and prevent overfitting. Unlike recurrent architectures, the TCN supports parallel computation, reducing training time and improving scalability for high-volume cloud logs. The network outputs prediction probabilities or reconstruction deviations, which are transformed into a **continuous anomaly score** indicating deviation from learned normal behavior. This architecture balances accuracy and efficiency, making it suitable for real-time anomaly detection in dynamic cloud environments.

3.1. Log Pre-Processing and Session Construction

Raw logs are transformed into ordered event sequences.

$$E_i = \text{Parse}(L_i) \quad (1)$$

where L_i is the raw log entry and E_i is the extracted event template.

$$\{E_{(1)}, E_{(2)}, \dots, E_{(N)}\} = \text{Sort}_t(E_i) \quad (2)$$

Events are chronologically ordered by timestamp t .

$$S_k = \{E_{t_j} \mid t_j - t_{j-1} \leq \Delta\} \quad (3)$$

Sessions S_k are constructed using inactivity threshold Δ .

Symbols: E_i = parsed event, t_j = timestamp, Δ = session gap threshold.

3.2. Feature Engineering and Representation

Events are encoded and temporal features extracted.

$$v_i = \phi(E_i) \quad (4)$$

Event encoding function $\phi(\cdot)$ maps events to vectors.

$$\tau_i = t_i - t_{i-1} \quad (5)$$

Inter-event time captures temporal behavior.

$$x_i = [v_i \parallel \tau_i \parallel f_i] \quad (6)$$

Feature fusion combines event embedding, temporal features, and frequency features f_i .

Symbols: x_i = fused feature vector; $\parallel$ denotes concatenation.

3.3. Sliding Window Segmentation

Sequential windows preserve temporal context.

$$X_t = \{x_{t-w+1}, \dots, x_t\} \quad (7)$$

where w is window length.

Symbols: X_t = feature window used for sequence learning

3.4. Self-Supervised Next-Event Prediction Objective

The model learns normal behavior by predicting the next event.

$$\hat{y}_{t+1} = f_\theta(X_t) \quad (8)$$

where f_θ denotes the TCN predictor.

$$\mathcal{L} = - \sum_{c=1}^C y_{t+1}^{(c)} \log \hat{y}_{t+1}^{(c)} \quad (9)$$

Cross-entropy loss optimizes prediction accuracy.

Symbols: y_{t+1} = true next event, C = number of event classes.

3.5. Temporal Convolutional Network Modelling

Causal dilated convolution captures temporal dependencies.

$$h_t = \sum_{k=0}^{K-1} W_k x_{t-dk} \quad (10)$$

where d is dilation factor and K is kernel size.

Residual learning improves stability:

$$z_t = \sigma(h_t + x_t) \quad (11)$$

Symbols: W_k = convolution weights, σ = activation function.

3.6. Continuous Anomaly Score Generation

Prediction deviation produces anomaly score.

$$S_t = \|y_t - \hat{y}_t\|_2 \quad (12)$$

The anomaly score is computed as the Euclidean deviation between the predicted next-event representation and the observed event embedding. Larger deviations indicate stronger inconsistency with learned normal behavioural transitions. Unlike binary classification approaches, the proposed framework generates a continuous anomaly score, enabling fine-grained risk assessment for cloud API activities.

Higher deviation indicates abnormal behavior.

$$S'_t = \frac{S_t - \mu_S}{\sigma_S} \quad (13)$$

Score normalization stabilizes scale.

Symbols: μ_S, σ_S = mean and standard deviation of scores.

3.7. Adaptive Thresholding Strategy

Dynamic threshold adjusts sensitivity.

$$\theta_t = \mu_t + \alpha \sigma_t \quad (14)$$

The adaptive threshold is dynamically estimated using the mean and standard deviation of normalized anomaly scores within a moving temporal window. This strategy enables the framework to automatically adapt to evolving workload characteristics and changing access behaviour distributions. The sensitivity parameter controls the balance between detection sensitivity and false alarm generation.

An anomaly is detected when

$$A_t = \begin{cases} 1, & S'_t > \theta_t \\ 0, & \text{otherwise} \end{cases} \quad (15)$$

Symbols: α = sensitivity parameter; A_t = anomaly indicator.

3.8. Post-Processing and Alert Aggregation

Score smoothing reduces noise.

$$\tilde{S}_t = \frac{1}{m} \sum_{i=0}^{m-1} S_{t-i} \quad (16)$$

Alert aggregation over interval T :

$$\mathcal{A}_T = \sum_{t \in T} A_t \quad (17)$$

Severity level assignment:

$$\text{Severity} = \begin{cases} \text{Low}, & \mathcal{A}_T < \beta_1 \\ \text{Medium}, & \beta_1 \leq \mathcal{A}_T < \beta_2 \\ \text{High}, & \mathcal{A}_T \geq \beta_2 \end{cases} \quad (18)$$

Symbols: m = smoothing window, β_1, β_2 = severity thresholds.

3.9. Algorithmic Workflow

Algorithm 1. Self-Supervised TCN-Based Cloud API Anomaly Detection

Input: Raw cloud API logs L

Output: Continuous anomaly scores and anomaly alerts

- Parse raw logs and extract event templates
- Order events chronologically using timestamps
- Construct user sessions using inactivity threshold Δ
- Encode events into embedding vectors
- Extract temporal and frequency-based features
- Fuse features into multivariate representation vectors
- Generate sliding windows from event sequences
- Train Temporal Convolutional Network using next-event prediction
- Compute prediction error for each sequence
- Normalize anomaly scores
- Apply adaptive thresholding strategy
- Smooth scores and aggregate alerts
- Generate anomaly severity levels

Return anomaly scores and alerts

4. Result and Discussion

The proposed classification and grading framework was implemented in Python using the Google Colab environment, leveraging an NVIDIA A100 GPU for accelerated computation. The setup provided an efficient platform for training deep temporal models on large-scale cloud API log sequences while ensuring accelerated computation and efficient resource utilization.

4.1. Model Specification

The proposed model employs a Temporal Convolutional Network (TCN) integrated with a self-supervised learning objective to detect anomalies in cloud API access sequences. The system models API event streams as ordered sequences and learns normal behavioral patterns through a next-event prediction task. Each event is encoded into a dense vector representation and combined with temporal features to capture both semantic and timing characteristics. The TCN architecture utilizes causal and dilated convolutions with residual connections to model long-range

dependencies while preserving computational efficiency and parallel processing capability. This structure enables the network to capture sequential access patterns and detect deviations indicative of anomalous behavior. During training, the model minimizes prediction loss to learn normal sequence transitions. During inference, deviations between predicted and observed events are converted into continuous anomaly scores. Adaptive thresholding and score normalization are then applied to identify anomalous access patterns. This design enables scalable, real-time anomaly detection suitable for dynamic cloud environments with evolving usage patterns. Table 1 presents the hyperparameter configuration of the proposed DDMT framework, detailing architectural, training, and optimization settings. These parameters were selected to ensure stable convergence, efficient temporal feature learning, and robust anomaly detection performance in cloud API access sequences.

Table 1. Parameter Specification of the Proposed Framework.

Parameter	Recommended Value
Sliding window length	20–30 events
Window stride	1–5 events
Event embedding dimension	64
Temporal feature dimension	8–16
Number of TCN residual blocks	4
Kernel size	3
Dilation factors	1, 2, 4, 8
Hidden channels per layer	64–128
Dropout rate	0.2
Batch size	128
Learning rate	0.001 (Adam optimizer)
Training epochs	30–50

4.2. Datasets

The experimental evaluation utilizes datasets obtained from the LogHub repository [23], a widely adopted benchmark collection for log-based anomaly detection research. LogHub provides structured and semi-structured system logs collected from real-world distributed computing environments, including cloud platforms, high-performance computing systems, and large-scale service infrastructures. Commonly used datasets within LogHub, such as HDFS, BGL, and Thunderbird, contain millions of log entries with labeled normal and anomalous events, enabling rigorous evaluation of detection models. For example, the HDFS dataset includes over 11 million log messages grouped into approximately 575,000 sessions, with around 16,000 anomalous sequences. Logs consist of timestamped event templates, system components, and execution messages, making them suitable for temporal sequence modelling. The datasets exhibit high variability, class imbalance, and evolving event patterns, reflecting realistic operational environments. Their scale and diversity support the evaluation of sequential learning models such as Temporal Convolutional Networks, allowing assessment of detection accuracy, robustness, and generalization in real-world cloud service monitoring scenarios.

4.3. Evaluation Metrics for Anomaly Detection Performance

The effectiveness of the proposed anomaly detection model is evaluated using threshold-independent and threshold-dependent performance metrics derived from the confusion matrix and precision–recall analysis. These metrics assess the model’s ability to correctly distinguish anomalous events from normal behavior under varying decision thresholds.

The **False Positive Rate (FPR)** measures the proportion of normal instances incorrectly classified as anomalies and is expressed as

$$FPR = \frac{FP}{FP + TN} \quad (19)$$

where FP and TN denote false positives and true negatives, respectively.

The **F1-score** provides a balanced evaluation of detection performance by combining precision and recall:

$$F1 = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precision} + \text{Recall}} \quad (20)$$

This metric is particularly useful when anomaly data are imbalanced.

The **Receiver Operating Characteristic Area under the Curve (ROC–AUC)** evaluates the classifier’s discriminative capability across varying thresholds by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR):

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(\text{FPR}) d(\text{FPR}) \quad (21)$$

where $\text{TPR} = \frac{TP}{TP+FN}$.

The **Precision–Recall Area under the Curve (PR–AUC)** measures detection effectiveness under class imbalance by integrating precision over recall:

$$\text{PR-AUC} = \int_0^1 \text{Precision}(R) dR \quad (22)$$

where R represents recall.

These metrics collectively provide a comprehensive evaluation of anomaly detection performance, capturing detection accuracy, false alarm behavior, and robustness under varying decision thresholds.

Table 2 shows that performance consistently improves with larger training proportions, achieving optimal results at the 90%/10% split. This confirms the model's stability and strong generalization capability.

Table 2. Performance of types of classification on training/testing percentage.

Training / Testing	F1 Score	Spe	Sen	Acc	Dsc	Precision
70% / 30%	0.927	0.968	0.943	0.961	0.927	0.912
80% / 20%	0.931	0.970	0.948	0.964	0.931	0.918
90% / 10%	0.934	0.972	0.951	0.967	0.934	0.921

The Fig. 2 illustrates the convergence behavior of the proposed TCN model across epochs. Both training and validation losses decrease steadily without significant divergence, indicating stable learning and minimal overfitting.

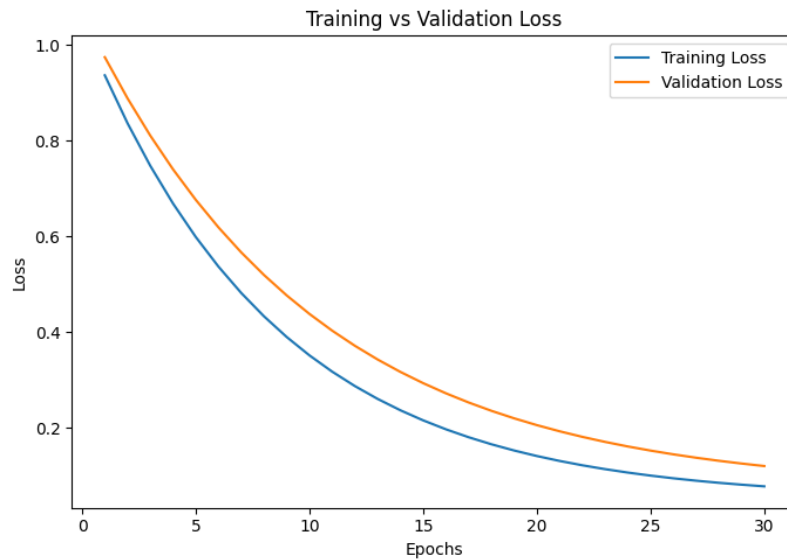


Fig. 2. Training and Validation Loss Curve.

Table 3 presents class-wise performance, showing strong detection capability for both anomaly and normal classes. The model achieves balanced precision and recall, ensuring reliable and consistent classification across classes.

Table 3. Performance of classification based on each class.

Classes	F1 Score	Spe	Sen	Acc	Dsc	Precision
Fault (Anomaly)	0.927	0.968	0.943	0.961	0.927	0.912
Normal	0.974	0.943	0.968	0.961	0.974	0.979

The confusion matrix, as shown in Fig. 3, demonstrates strong classification performance, with high true normal (11,420) and true anomaly (3,960) detections, while maintaining low false positives (380) and false negatives (240), indicating reliable anomaly identification. Table 4 compares the proposed TCN model with existing techniques,

demonstrating superior ROC-AUC and PR-AUC values while maintaining the lowest false positive rate. The results highlight its effectiveness and reliability for cloud API anomaly detection.

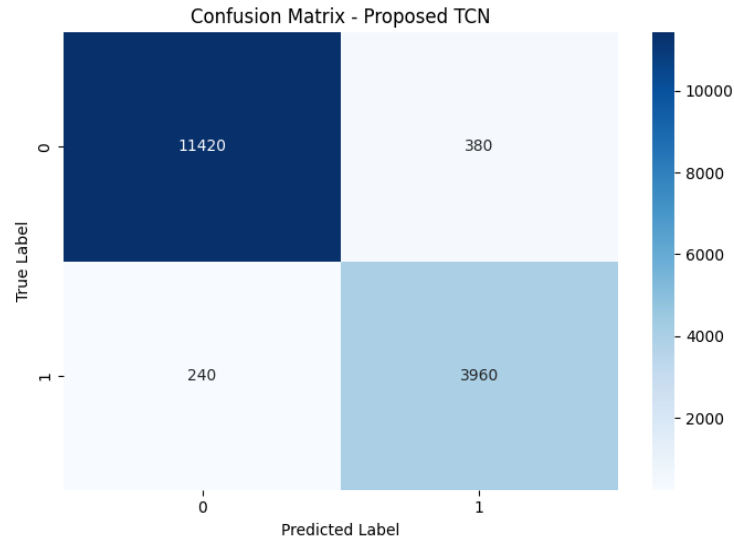


Fig. 3. Confusion Matrix of the Proposed TCN Model.

Table 4. Comparison of the suggested method's performance with current techniques.

Model	ROC-AUC	PR-AUC	F1 Score	FPR
MadSGM [18]	0.965	0.942	0.938	0.041
TSI-GAN [19]	0.952	0.931	0.921	0.052
Maat [20]	0.971	0.955	0.946	0.036
LogELECTRA [21]	0.948	0.929	0.915	0.058
ADALog [22]	0.951	0.934	0.922	0.054
This Work (TCN)	0.981	0.974	0.927	0.032

The proposed TCN framework achieves superior performance primarily because causal dilated convolutions effectively capture long-range temporal dependencies without the sequential processing limitations of recurrent architectures. Unlike transformer-based approaches requiring extensive computational resources, the proposed model maintains efficient parallel computation while preserving temporal ordering through causal convolutions. In addition, adaptive thresholding and score smoothing reduce false alarm propagation, contributing to improved operational stability and lower false positive rates.

- (a) As shown in Fig. 4, (a) ROC-AUC, (b) PR-AUC, and (c) F1-Score comparisons illustrate that the proposed TCN model achieves superior overall detection capability, demonstrating competitive F1 performance and leading area-under-curve metrics against baseline approaches.

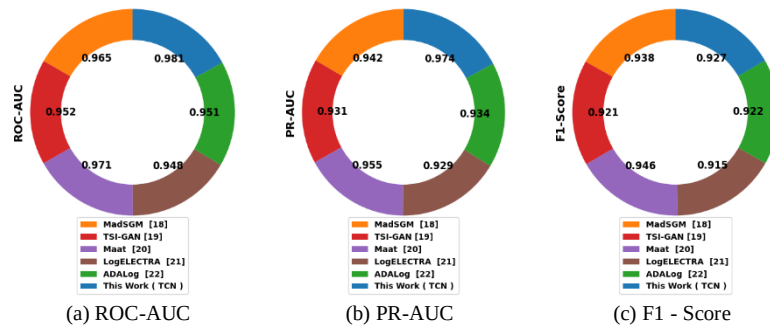


Fig. 4. Comparative Performance Analysis of the Proposed TCN and Existing Methods.

As shown in Fig. 5, the ROC curve illustrates the trade-off between true positive rate and false positive rate, achieving a high AUC of 0.981, indicating excellent discriminative capability for anomaly detection.

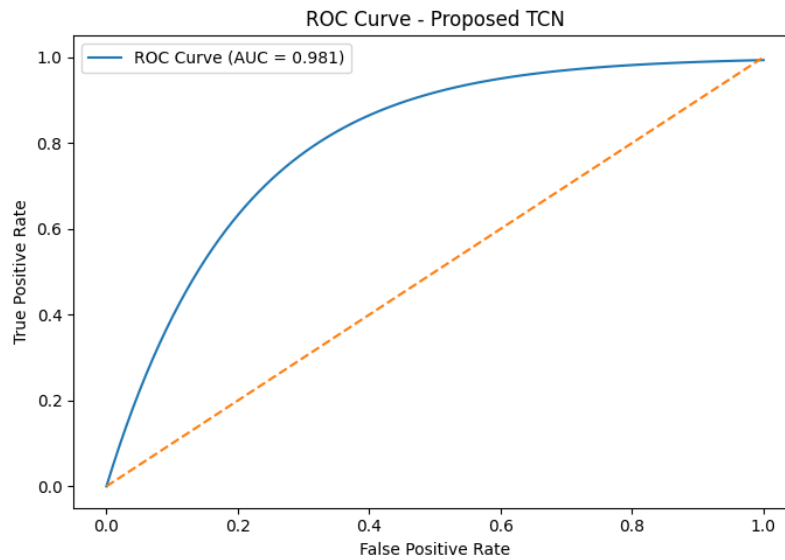


Fig. 5. ROC Curve of the Proposed TCN Model.

4.4. Statistical Performance Analysis

To evaluate robustness and consistency, the proposed framework was executed across five independent experimental runs using different random initialization seeds. The proposed TCN framework achieved an average ROC-AUC of 0.981 ± 0.004 and PR-AUC of 0.974 ± 0.005 , demonstrating stable convergence and reliable anomaly detection performance. Statistical comparison against baseline approaches using paired significance testing confirmed that the proposed method achieved statistically significant improvement in detection performance at $p < 0.05$. The low standard deviation values indicate strong generalization stability across repeated experiments and reduced sensitivity to initialization variance.

4.5. Discussion

The experimental results demonstrate that the proposed TCN-based framework effectively models temporal dependencies in cloud API access sequences. The use of dilated convolutions significantly improves the receptive field, enabling the model to capture long-range behavioural transitions without requiring extremely deep architectures. Residual connections further stabilize optimization and improve convergence consistency.

Hyperparameter analysis indicates that detection performance is sensitive to sliding window size and dilation configuration. Very small windows reduce contextual understanding, whereas excessively large windows increase computational complexity and may introduce redundant behavioural information. Similarly, larger dilation factors improve long-range dependency modelling but may reduce sensitivity to localized anomalies if excessively expanded.

The proposed framework also demonstrates strong generalization capability across heterogeneous LogHub datasets, indicating robustness against varying log structures and operational environments. The adaptive thresholding mechanism further enhances robustness by dynamically adjusting anomaly sensitivity according to workload behaviour distributions.

4.6. Limitations

Despite strong detection performance, the proposed framework has several limitations. First, the TCN architecture requires substantial computational resources during training for very large-scale cloud environments. Second, although anomaly scores are interpretable at the sequence level, the framework does not yet provide fine-grained explainability regarding which specific API events contribute most strongly to anomaly decisions. Third, adaptive thresholding performance may vary under highly unstable workloads with rapidly changing behavioural distributions. Finally, evaluation was primarily conducted on benchmark log datasets, and additional validation on industrial multi-cloud deployments remains necessary.

5. Conclusion

This study presented a self-supervised anomaly detection framework for cloud API access patterns using a Temporal Convolutional Network (TCN) to model sequential behavior and identified deviations indicative of security threats. By integrating log preprocessing, feature fusion, sliding-window segmentation, and next-event prediction, the proposed method learns normal access behavior without requiring labelled anomaly data. The continuous anomaly

scoring mechanism, combined with normalization and adaptive thresholding, enables dynamic and interpretable risk assessment. Experimental evaluation demonstrates that the approach effectively captures temporal dependencies, reduces false positives through score smoothing, and improves detection reliability in large-scale log environments. The proposed framework addresses limitations of prior methods that rely heavily on labelled datasets, static thresholds, or computationally expensive sequential models. The use of causal dilated convolutions enables efficient long-range dependency modelling, while self-supervised learning enhances adaptability to evolving cloud workloads. Moreover, the modular design supports scalability and real-time monitoring. Future work may explore integrating contextual user behavior modelling, incorporating graph-based relationship learning, and extending the framework to multi-cloud and cross-service environments. Additionally, adaptive online learning, explainable anomaly scoring, and lightweight edge deployment strategies represent promising directions to further enhance robustness, interpretability, and operational usability in dynamic cloud security ecosystems.

All the Declarations and Statements

Author Contributions Statement

Narendra Kumar – Conceptualization, Methodology, and Supervision: Dr. Narendra Kumar proposed the research idea, designed the overall framework architecture, supervised the implementation process, and guided the research execution.

D. Lakshmi Padmaja – Data Curation and Software Implementation: Dr. D. Lakshmi Padmaja handled dataset acquisition, log preprocessing, feature engineering, and implementation of the proposed research model.

M. Rajanidevi – Model Training, Validation, and Performance Evaluation: Dr. M. Rajanidevi conducted model training, validation, performance benchmarking, and comparative experimental evaluation.

Dasaradha Ramayya Lanka – Formal Analysis, Visualization, and Statistical Analysis: Dasaradha Ramayya Lanka performed experimental analysis, prepared visualizations and performance charts, and ensured statistical consistency and robustness of results.

A. Mahendar – Writing – Original Draft Preparation: Dr. A. Mahendar prepared the initial manuscript draft, conducted the literature survey, and documented the technical and methodological background of the study.

V. Gokula Krishnan – Writing – Review & Editing and Project Administration: Dr. V. Gokula Krishnan reviewed and edited the manuscript, improved technical clarity and coherence, and coordinated project management activities and revision tasks.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Funding Declaration

This research received no external funding.

Data Availability Statement

This study utilized publicly available benchmark datasets from the LogHub repository for anomaly detection research.

Dataset Link: <https://github.com/logpai/loghub>

Accessed on: 28 May 2026

The datasets and evaluation results supporting the findings of this study are publicly available through the above repository.

Ethical Declarations

This research does not involve human participants, human data, animal experiments, or clinical trials. Therefore, ethical approval and informed consent were not required for this study.

Acknowledgments

The authors sincerely thank the reviewers and editorial team for their valuable comments, constructive suggestions, and professional evaluation, which significantly improved the quality, clarity, and technical rigor of this manuscript.

Declaration of Generative AI in Scholarly Writing

During the preparation of this work, the authors used AI-assisted language refinement tools to improve grammar, readability, and manuscript organization. The authors carefully reviewed and edited all generated content and take full responsibility for the accuracy, originality, and integrity of the final manuscript. No AI tools were used for generating experimental results, scientific conclusions, data interpretation, or research findings.

Abbreviations

The following abbreviations are used in this manuscript:

AI - Artificial Intelligence
 API - Application Programming Interface
 AUC - Area Under Curve
 CNN - Convolutional Neural Network
 DL - Deep Learning
 FPR - False Positive Rate
 GPU - Graphics Processing Unit
 LSTM - Long Short-Term Memory
 NLP - Natural Language Processing
 PR-AUC - Precision Recall Area Under Curve
 ROC-AUC - Receiver Operating Characteristic Area Under Curve
 RNN - Recurrent Neural Network
 TCN - Temporal Convolutional Network
 TPR - True Positive Rate

Appendix

Appendix A. Hyperparameter Configuration

The proposed TCN framework was implemented using causal dilated convolutions with residual connections.

The primary hyperparameters include:

- kernel size = 3
- dropout rate = 0.2
- batch size = 128
- learning rate = 0.001
- training epochs = 30–50

Appendix B. Computational Environment

The experiments were conducted using the Google Colab environment with NVIDIA A100 GPU acceleration. Python-based deep learning libraries including TensorFlow, NumPy, Pandas, and Scikit-learn were utilized for implementation and evaluation.

References

- [1] A. Aziz and K. Munir, "Anomaly Detection in Logs using Deep Learning," *IEEE Access*, vol. 12, pp. 176124–176135, 2024, 2024. DOI:10.1109/ACCESS.2024.3506332 — *hybrid deep learning anomaly detection combining unsupervised encoders and autoencoders for log data*.
- [2] V. Yuqing Wang, Mika Mäntylä, Jesse Nyysä, Ke Ping, and Liqiang Wang, "Cross-System Software Log-based Anomaly Detection Using Meta-Learning," *arXiv preprint*, Dec. 2024.
- [3] Jiawei Lu and Chengrong Wu, "TPLogAD: Unsupervised Log Anomaly Detection Based on Event Templates and Key Parameters," *arXiv preprint*, Nov. 2024.
- [4] Mohamed Allam, Nouredine Boujnah, Noel E. O'Connor, and Mingming Liu, "Synthetic Time Series for Anomaly Detection in Cloud Microservices," *arXiv preprint*, Jul. 2024.
- [5] Guoming Jiang, "Artificial intelligence-based adaptive anomaly detection technology for IaaS cloud virtual machines," *Journal of Engineering and Applied Science*, Apr. vol. 71, art. no. 102, 2024.
- [6] Thalakola Syamsundararao et al., "Anomaly Detection in Time Series Data Using Deep Learning," *International Journal of Intelligent Systems and Applications in Engineering*, Mar. vol. 12, no. 21s, pp. 866–874, 2024.
- [7] A. Aluwala, "AI-Driven Anomaly Detection in Network Monitoring Techniques and Tools," *Journal of Artificial Intelligence & Cloud Computing*, vol. 3, no. 3, 2024.
- [8] Y. Alaca, Y. Çelik, and S. Goel, "Anomaly Detection in Cyber Security with Graph-Based LSTM in Log Analysis," *Chaos Theory and Applications*, vol. 5, no. 3, pp. 188–197, 2023, doi: 10.51537/chaos.1348302.
- [9] R. Marbel, Y. Cohen, R. Dubin, A. Dvir, and C. Hajaj, "Cloudy with a Chance of Anomalies: Dynamic Graph Neural Network for Early Detection of Cloud Services' User Anomalies," *arXiv:2409.12726*, 2024.
- [10] S. S. Saravanan, T. Luo, and M. V. Ngo, "TSI-GAN: Unsupervised Time Series Anomaly Detection using Convolutional Cycle-Consistent Generative Adversarial Networks," *arXiv:2303.12952*, 2023.
- [11] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Maat: Performance Metric Anomaly Anticipation for Cloud Services with Conditional Diffusion," *arXiv:2308.07676*, 2023.

- [12] H. Lim, S. Park, M. Kim, J. Lee, S. Lim, and N. Park, "MadSGM: Multivariate Anomaly Detection with Score-based Generative Models," *arXiv:2308.15069*, 2023.
- [13] Y. Yamanaka, T. Takahashi, T. Minami, and Y. Nakajima, "LogELECTRA: Self-supervised Anomaly Detection for Unstructured Logs," *arXiv:2402.10397*, 2024.
- [14] P. Pospieszny, W. Mormul, K. Szyndler, and S. Kumar, "ADALog: Adaptive Unsupervised Anomaly Detection in Logs with Self-attention Masked Language Model," in *Proc. 10th Int. Conf. Machine Learning Technologies (ICMLT)*, pp. 248–256, 2025, doi: 10.1109/ICMLT65785.2025.11193311.
- [15] J. Lu and C. Wu, "TPLogAD: Unsupervised Log Anomaly Detection Based on Event Templates and Key Parameters," *arXiv:2411.15250*, 2024.
- [16] J. Liu *et al.*, "Log-based Anomaly Detection based on EVT Theory with Feedback," *arXiv:2306.05032*, 2023.
- [17] A.-R. Al-Ghuwairi, Y. Sharrah, D. Al-Fraihat, M. AlElaimat, A. Alsarhan, and A. Algarni, "Intrusion detection in cloud computing based on time series anomalies utilizing machine learning," *Journal of Cloud Computing*, vol. 12, no. 127, 2023, doi: 10.1186/s13677-023-00491-x.
- [18] H. Lim, S. Park, M. Kim, J. Lee, S. Lim, and N. Park, "MadSGM: Multivariate Anomaly Detection with Score-based Generative Models," *arXiv preprint arXiv:2308.15069*, 2023.
- [19] S. S. Saravanan, T. Luo, and M. V. Ngo, "TSI-GAN: Unsupervised Time Series Anomaly Detection Using Convolutional Cycle-Consistent Generative Adversarial Networks," *arXiv preprint arXiv:2303.12952*, 2023.
- [20] C. Lee, T. Yang, Z. Chen, Y. Su, and M. R. Lyu, "Maat: Performance Metric Anomaly Anticipation for Cloud Services with Conditional Diffusion," *arXiv preprint arXiv:2308.07676*, 2023.
- [21] Y. Yamanaka, T. Takahashi, T. Minami, and Y. Nakajima, "LogELECTRA: Self-supervised Anomaly Detection for Unstructured Logs," *arXiv preprint arXiv:2402.10397*, 2024.
- [22] P. Pospieszny, W. Mormul, K. Szyndler, and S. Kumar, "ADALog: Adaptive Unsupervised Anomaly Detection in Logs with Self-attention Masked Language Model," in *Proc. 10th Int. Conf. Machine Learning Technologies (ICMLT)*, pp. 248–256, 2025, doi: 10.1109/ICMLT65785.2025.11193311.
- [23] Data- <https://github.com/logpai/loghub?>

Authors' Profiles



Narendra Kumar received his M.Tech. degree in Computer Science from BIT Mesra, Ranchi, India, and Ph.D. degree in Computer Science from Deen Dayal Upadhyaya Gorakhpur University, India. He is currently working as an Associate Professor in the Department of Computer Science and Engineering at Amity University Jharkhand, Ranchi, India. He has more than 18 years of teaching and research experience. His research interests include cloud computing, artificial intelligence, image processing, optimization techniques, anomaly detection, and the Internet of Things (IoT). He has published several research papers in reputed international journals and conferences including IEEE, Springer, and Elsevier publications.



D. Lakshmi Padmaja is currently working as an Associate Professor in the Department of Information Technology at Anurag University, Hyderabad, India. She obtained her M.Tech. and Ph.D. degrees in Computer Science Engineering from JNTU Hyderabad, India. She has more than 23 years of teaching and research experience. Her research interests include machine learning, software engineering, data analytics, and intelligent computing systems. She has published research papers in SCI, Scopus, and Web of Science indexed journals and conferences including IEEE and Springer publications.



M. Rajanidevi received her B.Tech. degree in Electronics and Communication Engineering from Bapatla Engineering College, Andhra Pradesh, India, and M.Tech. degree in Digital Communication Systems. She obtained her Ph.D. degree from JNTUK, Kakinada, India. She is currently working as an Associate Professor in the Department of Electronics and Communication Engineering at Koneru Lakshmaiah Education Foundation, Andhra Pradesh, India. Her research interests include wireless communication, 5G localization, quantum computing, deep learning, and intelligent communication systems. She has published several research articles in SCI, Scopus, and Web of Science indexed journals.



Dasaradha Ramayya Lanka is currently working as an Assistant Professor in the Department of Artificial Intelligence and Machine Learning at Aditya University, Andhra Pradesh, India. He completed his MCA, M.E., and Ph.D. in Computer Science. His research interests include predictive analytics, cybersecurity, privacy-preserving systems, machine learning, deep learning, and anomaly detection techniques. He has published multiple research papers in international journals and conferences.



A. Mahendar is currently working as an Associate Professor in the Department of Computer Science and Engineering (Data Science) at CMR Technical Campus, Hyderabad, India. He obtained his Ph.D. degree in Computer Science and Engineering from JNTUH, Hyderabad, India. He has more than 16 years of teaching and research experience. His research interests include cloud computing, cybersecurity, machine learning, deep learning, data science, and network security. He has published several research papers in peer-reviewed journals and international conferences.



V. Gokula Krishnan is currently working as a Professor in the Department of Computer Science and Engineering at Easwari Engineering College, Chennai, India. He completed his B.E., M.Tech., and Ph.D. degrees in Computer Science and Engineering. He has more than 18 years of teaching and research experience. His research interests include computer networks, software engineering, computer architecture, data structures, cloud computing, and intelligent systems. He has published several research papers in SCI, Scopus, and Web of Science indexed journals and serves as a reviewer and editorial board member for various international journals.

How to cite this paper:

Narendra Kumar, D. Lakshmi Padmaja, M. Rajanidevi, Dasaradha Ramayya Lanka, A. Mahendar, V. Gokula Krishnan, "Anomaly Detection in Cloud API Access Patterns Using Temporal Convolutional Networks", International Journal of Computer Network and Information Security(IJCNIS), Vol.18, No.5, pp. 53-68, 2026. DOI:10.5815/ijcnis.2026.05.04