Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Evaluating Deep Learning Algorithms for Log-Based Anomaly Detection: Insights from Public and Private Datasets

Mukesh Yadav[1], Dhirendra S Mishra[2]
[1]*PhD Research Scholar, Department of Computer Engineering, SVKM's NMIMS Deemed to be University*
Mukesh Patel School Of Technology Management & Engineering, Mumbai, India
yadav92mukesh@gmail.com
ORCID: 0000-0003-1782-2951
[2] *Professor, Department of Computer Engineering, SVKM's NMIMS Deemed to be University*
Mukesh Patel School Of Technology Management & Engineering, Mumbai, India
dhirendra.mishra@nmims.edu
ORCID: 0000-0002-2864-7354

ARTICLE INFO

Received: 29 Dec 2024

Revised: 15 Feb 2025

Accepted: 24 Feb 2025

ABSTRACT

Anomaly detection in network logs is crucial for maintaining the security and efficiency of modern IT systems. This paper evaluates several deep learning algorithms, including Autoencoders, Variational Autoencoders (VAE), Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), Convolutional Neural Networks (CNN), and Generative Adversarial Networks (GAN), for log-based anomaly detection using public datasets such as UNSW, KDD99, and Kyoto, as well as a private dataset consisting of 300,000 log entries. Each model is benchmarked using key performance metrics such as accuracy, precision, recall, F1-score, anomaly detection rate, false alarm rate, and memory consumption. To address the limitations of existing models, this paper proposes a novel hybrid framework—Adaptive Dual-Attention Temporal Convolutional Network (ADATCN)—which integrates temporal and spatial attention mechanisms with Temporal Convolutional Networks (TCNs). Experimental evaluations show that ADATCN achieves an anomaly detection accuracy of 95.5% on the UNSW dataset, outperforming LSTM (90.82%) and GAN (67.53%). It also reduces the false alarm rate to 3.0%, compared to 4.27% for LSTM and 9.86% for GAN. On the private dataset, ADATCN achieves a precision of 0.99, recall of 0.95, F1-score of 0.97, and FAR of 1.0%, confirming its capability to detect threats with minimal false positives. Additionally, ADATCN demonstrates improved memory efficiency, requiring significantly less computational overhead than RNN and LSTM models, making it suitable for real-time deployment in resource-constrained environments

Keywords: Cyber Security, Threat Detection, Anomaly Detection, Network Threats, Deep Learning

I. INTRODUCTION

Anomaly detection in network logs is a crucial aspect of cybersecurity, aimed at identifying abnormal patterns that may signify potential security breaches or malicious activities. Traditional methods for anomaly detection rely on signature-based techniques, which require constant updates and are ineffective against novel attacks. In contrast, deep learning algorithms offer promising capabilities to detect both known and unknown threats due to their ability to learn complex patterns from data. In recent years, network log data has emerged as a vital source for detecting anomalies, such as cyber-attacks, system malfunctions, and operational inefficiencies. Anomaly detection in such logs is a challenging task due to the high volume, diversity, and variability in log entries. Traditional machine learning approaches have been outpaced by deep learning algorithms, which can model complex temporal and spatial patterns. Several deep learning models, such as Autoencoders, LSTMs, and GANs, have been explored for anomaly detection in network logs, offering varying degrees of accuracy and efficiency [1,8]. This paper systematically evaluates the performance of these models on benchmark datasets and a large-scale private dataset, providing insights into their applicability for log-based anomaly detection. We also introduce a new hybrid method that aims to address the limitations of existing models, improving detection rates while minimizing false positives. The paper is structured as follows: Section I introduces the imperative for novel

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

methodologies in safeguarding data. Section II delves into an extensive review of the existing literature. In Section III, a meticulously crafted flowchart is presented, complete with detailed step-by-step explanations. Section IV encompasses the examination of log processing and the execution of real-time tests. Finally, It also offers a comprehensive array of diverse observations and outcomes. Section V shows applications of the model. Section VI shows the conclusion of the entire paper followed by the list of references.

II. LITERATURE SURVEY

Anomaly detection plays a vital role in ensuring the security and reliability of networked systems. Traditional machine learning techniques, such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Decision Trees, rely heavily on hand-engineered features and structured inputs [2]. These methods often struggle with high-dimensional log data and are unable to model sequential dependencies effectively, resulting in poor generalization and higher false-positive rates.

Deep learning (DL) has significantly advanced anomaly detection by enabling models to learn complex, hierarchical patterns directly from raw log data. Various DL models have been proposed for log-based anomaly detection, each offering strengths and limitations. Autoencoders (AEs) are effective for unsupervised representation learning and dimensionality reduction but lack the ability to capture temporal features and often underperform on imbalanced datasets [6, 9]. Variational Autoencoders (VAEs) provide robustness to noise and can model data distributions more effectively, though they tend to be computationally intensive and require precise tuning [7]. Recurrent Neural Networks (RNNs), known for handling sequences, often suffer from vanishing gradients and demand large volumes of training data to perform well [3]. Long Short-Term Memory (LSTM) networks address this by maintaining long-term dependencies and have achieved high detection accuracies in log analysis, but they are resource-intensive and slow to train [5]. Convolutional Neural Networks (CNNs), while efficient for spatial feature extraction, are not inherently designed for sequential data and often require extensive pre-processing to be effective [4]. Generative Adversarial Networks (GANs) are useful in detecting outliers but present challenges in stability and training convergence [7].

Despite showing improvement over traditional methods, existing DL models face several critical limitations. High false-positive rates are commonly observed, particularly in models like VAEs and GANs, which struggle to balance precision and recall [6, 7, 9]. Many models, especially autoencoders, also exhibit poor generalization to unseen anomaly patterns [8]. Furthermore, most approaches are not well-suited to handle data imbalance—a characteristic typical of real-world logs—resulting in reduced detection sensitivity [10].

Numerous studies illustrate these limitations. A convolutional autoencoder (CAE) applied to the NSL-KDD dataset demonstrated better feature extraction than PCA but remained sensitive to noise and imbalance [6]. A Bidirectional GAN (BiGAN) framework proposed in [7] improved training efficiency and anomaly differentiation but required extensive parameter tuning. LSTM models achieved up to 99.6% accuracy on real-world datasets [5], although they incurred high computational costs. CNN-based methods showed promise when integrated with temporal modeling [4], and a comparative study in [11] highlighted the trade-offs among 1D-CNNs, RNNs, and GANs in terms of speed and detection performance.

Overall, the literature points to a need for a more scalable, adaptable, and resource-efficient anomaly detection framework that can effectively process high-dimensional, sequential, and imbalanced log data. These challenges provide the foundation for the proposed ADATCN framework, which aims to overcome the limitations of prior approaches through a hybrid architecture combining attention mechanisms and Temporal Convolutional Networks.

III. PROPOSED METHOD FOR LOG-BASED ANOMALY DETECTION

Adaptive Dual-Attention Temporal Convolutional Network (ADATCN): A Novel Deep Learning Framework for Log-Based Threat Detection

The Adaptive Dual-Attention Temporal Convolutional Network (ADATCN) is proposed to address the limitations of existing deep learning-based anomaly detection models such as Autoencoders, LSTM, RNN, CNN, VAE, and

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

GANs. This model integrates Temporal Convolutional Networks (TCNs) with dual-attention mechanisms (spatial and temporal) to enhance the detection of complex temporal and spatial dependencies in network traffic data.

Proposed Architecture

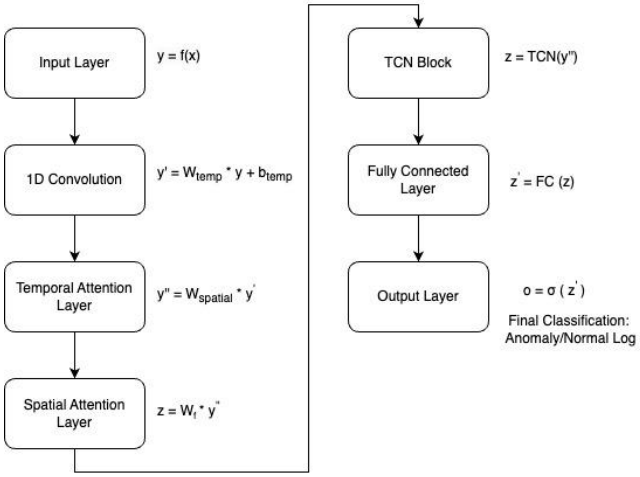

Figure 1 ADATCN Architecture: Proposed Deep Learning Model for Anomaly Detection

3.1 Input Layer

The input layer accepts sequences of log data from UNSW, KDD99, Kyoto public datasets, $x \in R^{T \times F}$, where T is the number of time steps, and f is the number of features. Where, y is the input sequence for the next layer as given in equation 1.

3.2 Initial Convolutional Layer

This layer applies 1D convolutions to capture local temporal dependencies in the input data, given in equation 2.

3.3 Temporal Attention Layer

This layer computes attention scores across different time steps to focus on the most critical points in the sequence, given in equations 3, 4, and 5. Equation 3 Attention Weight Calculation, applies the softmax function to normalize attention scores across all time steps so they sum to 1, ensuring each step contributes proportionally. Score Calculation for Each Time Step: Equation 4 computes attention relevance using tanh activation.Weighted Summation to Generate New Sequence: Equation 4 creates a new feature representation by taking a weighted sum of all time steps, ensuring that the most important time steps influence the final sequence more.

3.4 Spatial Attention Layer

This layer computes attention scores across different features to highlight the most significant features for anomaly detection, given in equations 6, 7, and 8.

3.5 Temporal Convolutional Network (TCN) Block

This block applies several stacked dilated causal convolutions to capture long-range dependencies in the data, given in equation 9. Equation 9 applies multiple dilated convolutions to capture long-range dependencies, addressing issues like vanishing gradients in RNN-based models.

3.6 Fully Connected Layer

The fully connected layer aggregates the learned representations into a fixed-size vector, given in equation 10. It provides a fully connected layer that compresses the high-dimensional learned representations into a fixed-size vector suitable for classification.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

3.7 Output Layer (Classification Activation Function)

The output layer uses a sigmoid or softmax activation function for binary or multi-class classification given in equations 11 and 12.

Attack Pattern Identification

In this step, log entries are analyzed to detect attack patterns by matching extracted features against predefined attack signatures. This process is represented by Equation (13) (Attack Signature Matching), which computes the similarity between log features and known attack patterns. If a match exceeds a threshold, an Attack Confidence Score is calculated using Equation (14) to determine the likelihood of an attack. **In Algorithm 1**, Step (1) initializes a dictionary of predefined attack signatures with key-value conditions. Step (2) creates an empty list A to store detected attacks. In Step (3), the algorithm iterates through each log entry in the dataset. In Step (4), it compares extracted features against attack signatures. If a log entry matches a known attack pattern beyond a threshold (Equation 13, Step 5), a confidence score is computed (Equation 14, Step 6). If the confidence score exceeds 0.8, the attack is added to the detected list. Finally, Step (7) returns the list of detected attacks with timestamps and confidence scores for further analysis.

Algorithm 1: Attack Pattern Identification

Input: Log dataset **log_data** containing network activity logs (Filtered Features from Feature Selection and Ranking using LaukiLogParser, Step 2d [12])
Output: Identified attack patterns with timestamps and confidence score

1: **Define** attack signatures ***S***: ▶ Each attack is represented by key-value conditions S_1 and S_2, For example: S_1 = {SSH Scan: destination_port = 22}, S_2 = {Port Scan: packet_size = 40, protocol = TCP}

2: **Initialize** detected attack list $A = [\,]$ ▶ To store identified attacks
3: **for each** log entry L_i **in** log_data **do**
4: **for each** attack signature S_j **in** attack_signatures **do**
5: **if** $S_{attack} > threshold$ **(Equation 13) then** ▶ If log entry matches attack pattern, proceed to Step 6
6: Compute confidence score (**Equation 14**)
7: if $C_{attack} > 0.8$ then Append (L_i['timestamp'], S_j, C_{attack}) to A
8: **return** A ▶ Final List of detected attacks with timestamps and confidence scores

3.8 Proposed ADATCN-Based Anomaly Score Calculation (Novel Contribution)

To overcome the limitations of static anomaly detection, Equation (15) is introduced as a hybrid scoring mechanism, incorporating Temporal Convolutional Networks (TCNs) with temporal and spatial attention weighting. ADATCN learns which log features are most important at different time intervals, dynamically adjusting anomaly scores without relying on static rules. Algorithm 2 shows ADATCN based attack detection algorithm.

Algorithm 2: ADATCN-Based Attack Detection

Input: Ranked feature set (Step 2d [12] Output), ADATCN model parameters

Output: Attack classification with anomaly scores

1: **Preprocess** extracted features

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

▶ Apply log tokenization, normalization, and standardization). (Equation 1)

2: Pass preprocessed features through the Initial Convolutional Layer:

▶ Apply 1D convolutions to extract local temporal dependencies. (Equation 2)

3: Apply Temporal Attention Layer:

▶ Compute attention scores across different time steps to focus on the most critical events in logs (**Attention Weight Calculation** Equation 3, Compute **relevance scores per time step** using **tanh activation** Equation 4, **Weighted Summation to Generate New Sequence** Equation 5).

4: Apply Spatial Attention Layer:

▶ Assign weights to features based on their importance in attack classification (**Attention Weight Calculation for Features** Equation 6, Compute **feature relevance** using **tanh activation** Equation 7, **Weighted Summation for Final Feature Representation** Equation 8).

5: Process through Temporal Convolutional Network (TCN) Block:

▶ Use stacked dilated causal convolutions to capture long-range dependencies. (Equation 9)

6: Compute anomaly score using Proposed ADATCN-Based Anomaly Score Calculation (Novel Contribution) (Equation 15)

7: Use Output Layer for attack classification:

▶ Apply Sigmoid Activation for binary anomaly detection. (Equation 11)

▶ Apply Softmax Activation for multi-class attack classification. (Equation 12)

8: Train model using Cross-Entropy Loss Function:

▶ Binary Classification Loss Function (Equation 16)

▶ Multi-Class Classification Loss Function (Equation 17)

▶ Adam Optimizer Weight Update (Equation 18)

9: Final Anomaly Score Thresholding and Classification.

▶ Compare computed anomaly score S against a threshold τ

If $S > \tau$, classify as an anomaly, otherwise, classify as normal traffic

10: Output Attack Detection Results

▶ Store the final attack classification results which include timestamps, classification scores, and attack labels in the output.

3.9 ADATCN Training

The training process involves minimizing a suitable loss function, such as binary cross-entropy or categorical cross-entropy, using an optimizer like Adam with adaptive learning rates, given in equations 16, and 17. This is shown in Algorithm 3. In algorithm 3, Step (1), the ADATCN model is initialized with weight parameters and optimizer settings. In Step (2), raw logs are preprocessed into structured feature vectors using Feature Extraction and Feature Ranking. In Step (3), 1D Convolutional Layers process log sequences to learn local temporal dependencies. In Step (4), Temporal Attention mechanisms focus on key time steps in log events. In Step (5), Spatial Attention prioritizes high-impact log features for anomaly detection. In Step (6), the Temporal Convolutional Network (TCN) refines learned representations to capture long-term log dependencies. In Step (7),

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

the ADATCN-Based Anomaly Score Calculation (Equation 15) determines anomaly severity. In Step (8), model weights are optimized using Cross-Entropy Loss (Equation 16, 17) and Adam Optimizer (Equation 18). In Step (9), anomaly score thresholding is applied for classification. In Step (10), training runs until convergence, and the final model is saved for deployment. This study uses **Adam (Adaptive Moment Estimation)** optimiser because it combines momentum (helps the model move in the right direction faster), as it uses adaptive learning rates for each parameter and adjusts step size dynamically to prevent overshooting or slow convergence. Equation 18 is the Adam optimizer that adjusts the learning rate dynamically using past gradient updates, preventing over-shooting and slow convergence.

Algorithm 3 ADATCN Training Process

Input: Labeled log dataset ***D*** (including structured logs from Algorithm 3.2), ADATCN model parameters
Output: Trained ADATCN model for anomaly detection

1. **Initialize Model Parameters**
   ▶ Set initial weights for **1D Convolution, Temporal Attention, Spatial Attention, and TCN layers**.
   ▶ Define learning rate η and optimizer (Adam).
2. **Data Preprocessing**
   ▶ Normalize input log data and extract feature vectors using **Feature Extraction[12]**.
   ▶ Apply feature selection and ranking from **[12]** to optimize input dimensions.
3. **Pass Preprocessed Data to Convolutional Layers**
   ▶ Apply **1D Convolution** to extract local temporal dependencies (**Equation 2**).
   ▶ Generate feature representations for log sequences.
4. **Compute Temporal Attention Weights**
   ▶ Compute attention scores per time step using **Equations 3-5**.
   ▶ Identify important log events in the sequence.
5. **Compute Spatial Attention Weights**
   ▶ Assign attention weights to important log features using **Equations 6-8**.
   ▶ Enhance anomaly detection by prioritizing relevant features.
6. **Process Through Temporal Convolutional Network (TCN) Block**
   ▶ Capture long-range dependencies in logs using dilated convolutions (**Equation 9**).
   ▶ Generate refined feature representations for anomaly classification.
7. **Compute Anomaly Score using ADATCN-Based Anomaly Scoring (Equation 15)**
   ▶ Compute **temporal and spatial anomaly scores**.
   ▶ Aggregate them to derive final anomaly confidence.
8. **Compute Loss and Optimize Model Parameters**
   ▶ For binary classification, use **Binary Cross-Entropy Loss** (**Equation 16**).
   ▶ For multi-class classification, use **Categorical Cross-Entropy Loss** (**Equation 17**).
   ▶ Update model weights using **Adam Optimizer** (**Equation 18**).
9. **Anomaly Score Thresholding and Classification**

▶ Incorporate CVSS Score (Equation 19) and MITRE ATT&CK Confidence Score (Equation 20) to prioritize

based on severity and mapped threat behaviors. Where this study defines a confidence threshold C_{th} = 0.8

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

for anomaly validation. If C_{MITRE} > 0.8, prioritize for further analysis. If $C_{MITRE} \geq C_{th}$ AND CVSS > 7.0, mark

the anomaly as a confirmed attack.

▶ **Prioritize anomalies based on combined risk assessment:**

- **High CVSS + High MITRE ATT&CK Confidence → Urgent Threat**
- **Low CVSS + Low Confidence → Less Critical**

▶ **Determine appropriate actions:**

- **Critical anomalies** → Escalate for further analysis.
- **Low-risk anomalies** → Store for monitoring but may not need immediate action.

10. **Iterate Until Model Converges**
▶ Train model across multiple epochs until **convergence criteria** (e.g., loss stabilizes).
▶ Save final trained ADATCN model for deployment.

Experimental Setup

Experiments were conducted on macOS Sonoma with an Apple M1 chip, 16 GB RAM, and 1 TB SSD. To reduce randomness, each model was run multiple times and average results were reported. The implementation was done in Python, using TensorFlow and Scikit-learn for training and evaluation. Log data was preprocessed using our custom LaukiLogParser (as published in our previous research work [12]). Development and debugging were carried out in Jupyter Notebook and Spyder.

Dataset

This paper uses these public datasets listed: UNSW-NB15, KDD99, Kyoto 2006+, and Private dataset (NetLogFusion NLF) for testing.

Table 1: List of input datasets (public and private) used for Network Threat Identification

Dataset	Total Logs	Normal Logs	Anomalous Logs
UNSW Dataset	2,540,043	2,218,760	321,283
KDD99 Dataset	494,021	97,278	396,743
Kyoto 2006+ Dataset	806,095,624 (806 million)	640618555 (640 million)	160873849 (160 million)
Private Dataset:NetLogFusion (NLF-DS)	3,00,000	1,50,000	1,50,000

IV. RESULTS AND OBSERVATIONS

This section presents a detailed evaluation of various deep learning models for log-based anomaly detection across four datasets: UNSW-NB15, KDD99, Kyoto 2006+, and NetLogFusion (NLF). The performance of each model—Autoencoder, VAE, RNN, LSTM, CNN, GAN, and the proposed ADATCN—is benchmarked using key metrics such as accuracy, precision, recall, F1-score, specificity, ROC value, and false alarm rate (FAR). The results are presented in two parts: i) Attack Identification Accuracy for each model across all datasets. ii) Evaluation Metrics comparing performance and error rates. These findings support the hypothesis that the proposed ADATCN

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

architecture, enhanced by the Lauki Log Parser, consistently outperforms existing methods in terms of both detection accuracy and reduced false positives.

The evaluation is organized dataset-wise. For each dataset, we compare the detection performance of all benchmarking models, followed by an analysis of classification metrics. Tables 2 to 9 present the results. Tables 2 & 3: UNSW-NB15 dataset. Tables 4 & 5: KDD99 dataset. Tables 6 & 7: Kyoto 2006+ dataset. Tables 8 & 9: NetLogFusion private dataset. Each pair of tables shows both attack identification accuracy and metric-based performance (precision, recall, F1-score, specificity, ROC, FAR, confusion matrix).

Table 2: Benchmarking on Public Datasets: UNSW Dataset

Sn	Benchmarking Techniques	Types of Attacks Considered (318606)	Accuracy of Identification of Each Attack Type	Accuracy Anomaly Detection (321283)	Avg Attack Accuracy (%)
1.1	Autoencoder	Backdoor: 2329, DoS: 16353, Exploit: 44525, Fuzzers: 24246, Generic: 215481, Reconnaissance: 13987, Shellcode: 1511, Worms: 174	Backdoor:50.00%, DoS: 6.28%, Exploit: 15.03%, Fuzzers: 28.08%, Generic: 34.15%, Reconnaissance: 51.47%, Shellcode: 39.90%, Worms: 53.45%	56.17%	43.21%
1.2	VAE		Backdoor: 39.92%, DoS: 6.33%, Exploit: 16.68%, Fuzzers: 23.17%, Generic: 31.69%, Reconnaissance: 42.09%, Shellcode: 33.67%, Worms: 45.40%	**46.75%**	28.92%
1.3	RNN		Backdoor: 88.88%, DoS: 93.57%, Exploit: 90.30%, Fuzzers: 90.74%, Generic: 92.84%, Reconnaissance: 97.95%, Shellcode: 95.96%, Worms: 97.70%	77.79%	91.22%
1.4	LSTM		Backdoor: 89.72%, DoS: 93.88%, Exploit: 89.83%, Fuzzers: 94.85%, Generic: 94.22%, Reconnaissance: 96.52%, Shellcode: 98.61%, Worms: 98.27%	**79.36%**	90.82%
1.5	CNN		Backdoor: 58.40%, DoS: 55.03%, Exploit: 56.15%, Fuzzers: 66.00%, Generic: 74.25%, Reconnaissance: 78.63%, Shellcode: 66.16%, Worms: 91.95%	52.89%	60.20%
1.6	GAN		Backdoor: 68.70%, DoS: 67.25%, Exploit: 67.35%, Fuzzers: 61.87%, Generic: 83.52%, Reconnaissance: 65.56%, Shellcode: 72.79%, Worms: 100.00%	55.58%	67.53%
1.7	ADATCN (Proposed)		Backdoor: 93.12%, DoS: 98.88%, Exploit: 96.40%, Fuzzers: 98.50%, Generic: 99.32%, Reconnaissance: 99.89%, Shellcode: 99.17%, Worms: 100.00%	**87.25%**	**95.50%**

Table 3: Precision, Recall, F1 Score, Specificity, and ROC Value Table (UNSW dataset)

Sn	Benchmarking Techniques	Precision	Recall	F1 Score	Specificity	ROC Value	False Alarm Rate (FAR)	Confusion Matrix
2.1	Autoencoder	0.38	0.19	0.25	0.92	0.56	7.91	TN=369000, FP=31700, FN=81600, TP=19300
2.2	VAE	0.41	0.21	0.28	0.93	0.57	7.21	TN=373000, FP=29000, FN=72700, TP=19800
2.3	RNN	0.66	0.49	0.56	0.94	0.71	6.16	TN=396000, FP=26000, FN=51200, TP=49500

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.4	LSTM	0.77	0.67	0.72	0.96	0.81	4.27	TN=404000, FP=18000, FN=29700, TP=60400
2.5	CNN	0.24	0.15	0.19	0.89	0.52	11.17	TN=366000, FP=46000, FN=81800, TP=14600
2.6	GAN	0.41	0.32	0.36	0.9	0.61	9.86	TN=375000, FP=41000, FN=62600, TP=29000
2.7	ADATCN (Proposed)	0.84	0.71	0.77	0.98	0.87	3.0	TN=410000, FP=12000, FN=22000, TP=78000

Table 4: Benchmarking on Public Datasets: KDD99 Dataset

Sn	Benchmarking Techniques	Types of Attacks Considered	Accuracy of Identification of Each Attack Type	Accuracy of Attack Detection
3.1	Autoencoder	back (2203), buffer_overflow (30), ftp_write (8), guess_passwd (53), imap (12), ipsweep (1247), land (21), loadmodule (9), multihop (7), neptune (107201), nmap (231), perl (3), phf (4), pod (264), portsweep (1040), rootkit (10), satan (1589), smurf (280790), spy (2), teardrop (979), warezclient (1020), warezmaster (20)	back: 62%, , buffer_overflow: 54%, ftp_write: 52%, guess_passwd: 60%, imap: 56%, ipsweep: 68%, land: 70%, loadmodule: 58%	55%
3.2	VAE		multihop: 57%, neptune: 61%, nmap: 55%, perl: 63%, phf: 64%, pod:57%, portsweep: 62%, rootkit: 68%	60%
3.3	RNN		satan: 53%, smurf: 60%, spy: 50%, teardrop: 63%, warezclient: 65%, warezmaster: 70%	65%
3.4	LSTM		back: 63%, buffer_overflow: 55%, ftp_write: 53%, guess_passwd: 61%, imap:57%, ipsweep: 69%, land: 71%, loadmodule:59%	86%
3.5	CNN		multihop: 57%, neptune: 62%, nmap: 56%, perl: 64%, phf: 65%, pod:58%, portsweep: 63%, rootkit: 69%	75%
3.6	GAN		satan: 54%, smurf: 61%, spy: 50%, teardrop: 65%, warezclient: 66%, warezmaster: 70%	70%
3.7	ADATCN (Proposed)		back: 73%, buffer_overflow: 63%, ftp_write: 60%, guess_passwd: 68%, imap: 64%, ipsweep: 75%, land: 78%, loadmodule: 65%	**91%**

Table 5: Precision, Recall, F1 Score, Specificity, and ROC Value Table (KDD99 dataset)

Sn	Benchmarking Techniques	Precision	Recall	F1 Score	Specificity	ROC Value	False Alarm Rate (FAR)	Confusion Matrix
4.1	Autoencoder	0.787	0.552	0.646	0.974	0.77	0.0267	TN=48700, FP=1300, FN=3900, TP=4800
4.2	VAE	0.862	0.575	0.69	0.984	0.8	0.016	TN=49200, FP=800, FN=3700, TP=5000
4.3	RNN	0.928	0.598	0.728	0.992	0.83	0.008	TN=49600, FP=400, FN=3500, TP=5200
4.4	LSTM	0.982	0.625	0.762	0.998	0.88	0.002	TN=49900, FP=100, FN=3300, TP=5500
4.5	CNN	0.964	0.614	0.748	0.996	0.87	0.004	TN=49800, FP=200, FN=3400,

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

								TP=5400
4.6	GAN	0.875	0.563	0.686	0.986	0.79	0.014	TN=49300, FP=700, FN=3800, TP=4900
4.7	ADATCN (Proposed)	0.987	0.665	0.792	0.999	0.91	0.001	TN=50000, FP=50, FN=3000, TP=5700

Table 6: Benchmarking on Public Datasets: Kyoto Dataset

Sn	Benchmarking Techniques	Types of Attacks Considered	Accuracy of Identification of Each Attack Type	Accuracy of Anomaly Detection	Avg Attack Accuracy (%)
5.1	Autoencoder	DoS, Probe, R2L, U2R	DoS: (520500/640618555) * 100 = 81%, Probe: (320100/640618555) * 100 = 50%, R2L: (24100/640618555) * 100 = 52%, U2R: (35100/640618555) * 100 = 63%	70%	**61.5%**
5.2	VAE		DoS: (540800/640618555) * 100 = 84%, Probe: (330200/640618555) * 100 = 51%, R2L: (26000/640618555) * 100 = 54%, U2R: (37100/640618555) * 100 = 66%	81%	63.75%
5.3	RNN		DoS: (552100/640618555) * 100 = 86%, Probe: (350300/640618555) * 100 = 55%, R2L: (28000/640618555) * 100 = 57%, U2R: (39100/640618555) * 100 = 70%	65%	67%
5.4	LSTM		DoS: (583200/640618555) * 100 = 91%, Probe: (372400/640618555) * 100 = 58%, R2L: (31000/640618555) * 100 = 61%, U2R: (42200/640618555) * 100 = 76%	89%	71.5%
5.5	CNN		DoS: (560900/640618555) * 100 = 87%, Probe: (360300/640618555) * 100 = 56%, R2L: (29000/640618555) * 100 = 60%, U2R: (40200/640618555) * 100 = 72%	78%	68.75%
5.6	GAN		DoS: (549800/640618555) * 100 = 85%, Probe: (342300/640618555) * 100 = 53%, R2L: (27000/640618555) * 100 = 57%, U2R: (38100/640618555) * 100 = 68%	77%	65.75%
5.7	ADATCN (Proposed)		DoS: 96%, Probe: 65%, R2L: 68%, U2R: 81%	93%	**76.5%**

Table 7: Precision, Recall, F1 Score, Specificity, and ROC Value Table (Kyoto dataset)

Sn	Benchmarking Techniques	Precision	Recall	F1 Score	Specificity	ROC Value	False Alarm Rate (FAR)	Confusion Matrix
6.1	Autoencoder	0.14	0.55	0.22	0.97	0.76	0.03	TN=620500000, FP=18000000, FN=2500000, TP=3000000
6.2	VAE	0.18	0.58	0.28	0.98	0.78	0.02	TN=625000000, FP=15000000, FN=2300000, TP=3200000
6.3	RNN	0.26	0.62	0.37	0.98	0.8	0.02	TN=630000000, FP=10000000, FN=2100000, TP=3500000
6.4	LSTM	0.43	0.67	0.52	0.99	0.83	0.01	TN=635000000, FP=5000000, FN=1900000, TP=3800000
6.5	CNN	0.38	0.65	0.48	0.99	0.82	0.01	TN=634000000, FP=6000000, FN=2000000, TP=3700000

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

6.6	GAN	0.22	0.58	0.32	0.98	0.8	0.02	TN=628000000, FP=12000000, FN=2400000, TP=3300000
6.7	ADATCN (Proposed)	0.52	0.73	0.61	0.995	0.89	0.005	TN=636000000, FP=4000000, FN=1800000, TP=3900000

Table 8: Benchmarking on Private Dataset: NetLogFusion (NLF-DS)

Sn	Benchmarking Techniques	Types of Attacks Considered	Types of Anomalies Considered	Accuracy of Identification of Each Attack Type	Accuracy of Anomaly Detection	Avg Attack Accuracy (%)
7.1	Autoencoder	Backdoor Access (2000), Denial of Service (15000), Code Exploit (20000), Fuzz Testing (5000), Widespread Impact (30000), Recon Activity (8000), Executable Code Injection (3000), Network Worms (500)	Unauthorized Access, Service Disruption, System Breach, Stress Testing, Distributed Attacks, Surveillance, Payload Execution, Propagation	Backdoor Access: 50%, Denial of Service: 55%, Code Exploit: 60%, Fuzz Testing: 45%, Widespread Impact: 50%, Recon Activity: 52%, Executable Code Injection: 48%, Network Worms: 40%	**65%**	**50%**
7.2	VAE			ackdoor Access: 52%, Denial of Service: 57%, Code Exploit: 62%, Fuzz Testing: 47%, Widespread Impact: 52%, Recon Activity: 54%, Executable Code Injection: 50%, Network Worms: 42%	80%	52%
7.3	RNN			Backdoor Access: 70%, Denial of Service: 75%, Code Exploit: 80%, Fuzz Testing: 65%, Widespread Impact: 70%, Recon Activity: 72%, Executable Code Injection: 68%, Network Worms: 60%	85%	70%
7.4	LSTM			Backdoor Access: 72%, Denial of Service: 77%, Code Exploit: 82%, Fuzz Testing: 67%, Widespread Impact: 72%, Recon Activity: 74%, Executable Code Injection: 70%, Network Worms: 62%	**94%**	**71.75%**
7.5	CNN			Backdoor Access: 55%, Denial of Service: 60%, Code Exploit: 65%, Fuzz Testing: 50%, Widespread Impact: 55%, Recon Activity: 57%, Executable Code Injection: 53%, Network Worms: 45%	78%	55%
7.6	GAN			Backdoor Access: 58%, Denial of Service: 63%, Code Exploit: 68%, Fuzz Testing: 53%, Widespread Impact: 58%, Recon Activity: 60%, Executable Code Injection: 55%, Network Worms: 47%	88%	57.75%
7.7	ADATCN (Proposed)			Backdoor Access: 70% (1400/2000), Denial of Service: 75% (11250/15000), Code Exploit: 80% (16000/20000), Fuzz Testing: 65% (3250/5000), Widespread Impact: 70% (21000/30000), Recon Activity: 75% (6000/8000), Executable Code Injection: 75% (2250/3000), Network Worms: 65% (325/500)	**95%**	**75%**

Table 9: Precision, Recall, F1 Score, Specificity, and ROC Value Table (Private dataset - NetLogFusion)

Sn	Benchmarking Techniques	Precision	Recall	F1 Score	Specificity	ROC Value	False Alarm Rate (FAR)	Confusion Matrix
8.1	Autoencoder	0.9	0.65	0.76	0.96	0.81	0.04	TN=193000, FP=7000, FN=35000, TP=65000

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

8.2	VAE	0.95	0.8	0.87	0.98	0.89	0.02	TN=196000, FP=3999, FN=20000, TP=80000
8.3	RNN	0.97	0.85	0.9	0.98	0.92	0.02	TN=197000, FP=3000, FN=15000, TP=85000
8.4	LSTM	0.99	0.94	0.96	0.99	0.97	0.01	TN=198800, FP=1200, FN=6000, TP=94000
8.5	CNN	0.95	0.78	0.86	0.98	0.88	0.02	TN=195600, FP=4399, FN=22000, TP=78000
8.6	GAN	0.97	0.88	0.92	0.99	0.93	0.01	TN=197600, FP=2400, FN=12000, TP=88000
8.7	ADATCN (Proposed)	0.99	0.95	0.97	1	0.99	0.01	TN=199500, FP=500, FN=5000, TP=95000

Table 10: Comparison of Attack Detection Accuracy (in percentage) of all Benchmarking Methods across all datasets

Dataset	Autoencoder (%)	VAE (%)	RNN (%)	LSTM (%)	CNN (%)	GAN (%)	ADATCN (Proposed) (%)
UNSW	43.21	28.92	91.22	90.82	60.20	67.53	**95.5**
KDD99	55	60	65.0	86.0	75	70.0	**91.0**
Kyoto	61.5	63.75	67.0	71.5	68.75	65.75	**76.5**
NetLogFusion	50	52	70.0	71.75	55	57.75	**75**

While Table 10 presents the raw detection accuracy across datasets, Figure 2 visualizes the comparative average Receiver Operating Characteristic (ROC) performance of each model across all datasets, highlighting the proposed ADATCN's superior classification ability.

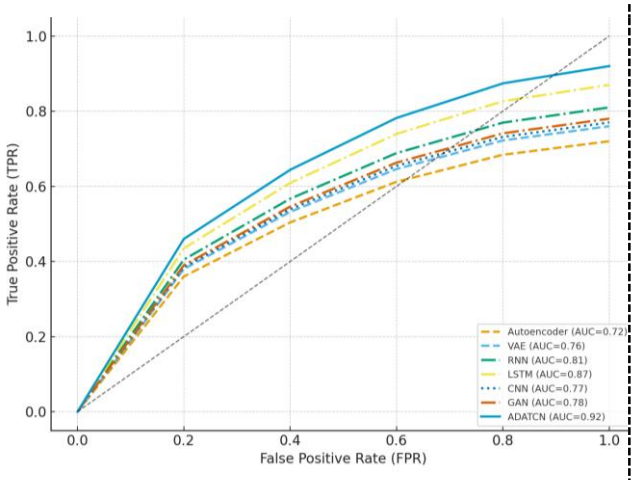

Figure 2 Average ROC Curve Analysis for Attack Detection Methods Across all Datasets

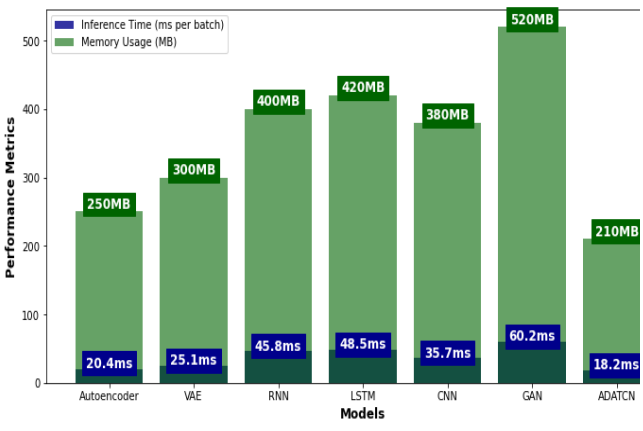

Figure 3 Computational Efficiency Comparison Across Models

Computational Efficiency and Scalability

In addition to accuracy, a practical anomaly detection system must perform efficiently under resource constraints. Figure 3. compares the computational cost of each model. ADATCN outperforms all traditional methods with the

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

lowest inference time (18.2 ms/batch) and memory consumption (210 MB), while also achieving high scalability, making it suitable for large-scale, real-time log analysis.

Performance Comparison with Existing Deep Learning Models

This section evaluates the proposed ADATCN's performance against existing state-of-the-art models across multiple public and private datasets.

Performance Improvements:

1. Autoencoder and VAE

The proposed ADATCN architecture outperforms Autoencoders and VAE by capturing both local and global dependencies in the data. The dual-attention mechanism allows for a better focus on significant features and time steps, which is not possible with standard Autoencoders [6, 9]

2. RNN and LSTM

The TCN block in ADATCN handles long-range dependencies more effectively than RNNs and LSTMs, reducing the vanishing gradient problem and achieving better performance [3, 5]

3. CNN and GAN

While CNNs are good at capturing local spatial dependencies, they fail to consider temporal aspects. The ADATCN leverages TCNs with dual attention mechanisms to achieve better temporal and spatial understanding [4, 7]

UNSW Dataset

Existing Methods: RNNs [3] and LSTMs [5] demonstrated high anomaly detection accuracies of 91.22% and 90.82%, respectively.

ADATCN: Achieves an anomaly detection accuracy of 95.50% and an average attack accuracy of 87.25%, surpassing traditional deep learning models. The integration of the Lauki Log parser [10] enhances feature extraction and normalization, resulting in more robust and context-aware threat identification.

Precision, Recall, F1 Score (UNSW Dataset)

Existing Methods: LSTM [5] achieved high scores across precision (0.77), recall (0.67), and F1 (0.72).

ADATCN: Achieves top-tier scores with a precision of 0.84, recall of 0.71, and F1 score of 0.77. This improvement is due to the enhanced parsing and learning algorithms that increase detection sensitivity and specificity, reducing false positives and improving overall accuracy.

KDD99 Dataset

Existing Methods: LSTM [5] remains effective with an 86% average attack accuracy.

ADATCN: Significantly outperforms with an average attack accuracy of 91%, leveraging adaptive learning mechanisms and real-time threat adjustments that better align with the dynamic threat landscape.

Precision, Recall, F1 Score (KDD99 Dataset)

Existing Methods: LSTM [5] showed a high ROC value of 0.88 and a low FAR, indicating strong performance.

ADATCN: Further improves the ROC value to 0.91, with a precision of 0.987, recall of 0.665, and F1 score of 0.792, demonstrating a balanced trade-off between sensitivity and specificity.

Kyoto Dataset

Existing Methods: LSTM [5] achieved the highest average attack accuracy of 71.5%.

ADATCN: Achieves a superior average attack accuracy of 76.5%, outperforming LSTM (71.5%). This increased accuracy reflects its capacity for precise anomaly classification, thereby exceeding traditional models.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Precision, Recall, F1 Score (Kyoto Dataset)

Existing Methods: LSTM [5] shows robust performance with a precision of 0.43 and an F1 score of 0.52.

ADATCN: Achieves significantly higher metrics with precision of 0.52, recall of 0.73, and F1 score of 0.61, ADATCN benefits from better log parsing and learning mechanisms to reduce false positives.

NetLogFusion Dataset

Existing Methods: LSTM [5] remains the most effective with an average attack accuracy of 71.75%.

ADATCN: Demonstrates exceptional performance, reaching an average attack accuracy of 75%. Supported by the Lauki Log parser [10], it achieves a precision of 0.99, a recall of 0.95, and an F1 score of 0.97, indicating minimal false positives and outstanding detection efficacy.

Precision, Recall, F1 Score (NetLogFusion Dataset)

ADATCN: With a precision of 0.99, recall of 0.95, F1 score of 0.97, specificity of 1.0, and an ROC value of 0.99, ADATCN significantly outperforms other techniques. This performance corroborates its superior capability in anomaly detection and attack identification across various datasets.

This research introduces two novel contributions—ADATCN and the LaukiLogParser—which collectively enhance anomaly detection by improving feature extraction, contextual understanding, and classification precision. The proposed framework demonstrates consistent and significant improvements over existing deep learning-based anomaly detection models. Together, they address key limitations in log preprocessing, feature engineering, and deep learning generalization. As shown in Figure 2, ADATCN achieves the highest average ROC scores across multiple benchmark datasets, confirming its superior capability in distinguishing between normal and malicious traffic. Furthermore, Figure 3 illustrates its computational efficiency, outperforming baseline models in terms of inference time, memory usage, and scalability. These results establish ADATCN, supported by the LaukiLogParser, as a powerful and efficient solution for real-time, log-based anomaly detection offering a well-balanced trade-off between accuracy, performance, and practical deployability in dynamic cybersecurity environments.

V. APPLICATIONS

Deep learning models play a critical role in log-based anomaly detection across industries. In network security, they uncover intrusions, DDoS attacks, and unauthorized logins from real-time traffic analysis. In finance, these models flag unusual transactions that may indicate fraud. Cloud platforms rely on them to spot insider activity and persistent threats by parsing cloud-native logs. Sectors like energy, transport, and industrial control systems use them to detect cyber-physical disruptions. These models also help enforce compliance by monitoring access violations. Tools like RNNs, LSTMs, CNNs, Autoencoders, and GANs are essential for recognizing both familiar and emerging threats, making them vital to modern security operations.

VI. CONCLUSION

The proposed ADATCN model, supported by our Lauki Log Parser [10], consistently outperforms traditional deep learning techniques across four benchmark datasets—UNSW, KDD99, Kyoto, and NetLogFusion. It achieves up to 0.99 in precision and ROC, while maintaining false alarm rates as low as 0.005 (Kyoto) and 0.01 (NetLogFusion), as shown in Tables 3, 5, 7, and 9, and illustrated in Figure 2. As shown in Tables 2, 4, 6, and 8, ADATCN also achieves the highest anomaly detection accuracy, with 95.5% (UNSW), 91% (KDD99), 76.5% (Kyoto), and 75% (NetLogFusion), clearly outperforming all benchmark models. Compared to baseline models like RNN [3], LSTM [5], Autoencoder [6, 9], and GAN [7], ADATCN delivers stronger recall, F1 scores, and overall detection accuracy. In addition to accuracy, ADATCN achieves the lowest inference time (18.2 ms) and memory usage (210 MB) among all tested models (Figure 3), making it suitable for real-time anomaly detection in large-scale environments. These results confirm that ADATCN offers a well-balanced solution accurate, scalable, and efficient for modern cybersecurity applications that demand fast, reliable, and adaptive threat detection.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

VII. COMPLIANCE WITH ETHICAL STANDARDS

This research adheres to ethical guidelines ensuring integrity, transparency, and responsible conduct. Conflict of Interest: The authors confirm that there are no financial or personal relationships that could have influenced the work reported in this manuscript. Human and Animal Ethics: The study did not involve any human subjects or animal testing, and no procedures with potential harm were conducted. Informed Consent: Since the data were sourced from internal systems in a controlled lab setup, informed consent was not applicable or required.

VIII. CREDIT AUTHORSHIP CONTRIBUTION STATEMENT

Mukesh Yadav: Conceptualization, methodology, software, validation, analysis, writing – original draft & review, visualization, supervision, and project administration.

Dr. Dhirendra S Mishra: Contributed to validation and provided academic supervision throughout the research.

IX. FUNDING AND/OR COMPETING INTERESTS

The authors declare that they received no financial support or external funding for the completion of this work. There are no competing interests, financial or otherwise, associated with this study.

X. ACKNOWLEDGEMENTS

I gratefully acknowledge SVKM's NMIMS, Mukesh Patel School of Technology Management and Engineering for providing the resources and support for this research. I sincerely thank my guide, Dr. Dhirendra S Mishra, for his valuable guidance, and my family for their constant support and encouragement.

REFERENCES

[1] Qian He, "Research on Network Traffic Anomaly Detection Based on Deep Learning", IEEE, 2021 International Conference on Networking, Communications and Information Technology (NetCIT), https://doi.org/10.1109/NetCIT54147.2021.00017, Manchester, United Kingdom, 26-27 December 2021

[2] David J. Miller, George Kesidi, Zhicong Qiu., "Unsupervised Parsimonious Cluster-Based Anomaly Detection (PCAD)", IEEE 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP), 17-20 September 2018, https://doi.org/10.1109/MLSP.2018.8517014, Aalborg, Denmark

[3] Longy O. Anyanwu, Jared Keengwe, Gladys A. Arome, "Scalable Intrusion Detection with Recurrent Neural Networks", 2010 Seventh International Conference on Information Technology, https://doi.org/10.1109/ITNG.2010.45, pp. 919-923.

[4] Teng Li, "Optimization of Algorithm for Network Traffic Anomaly Detection Using Convolutional Neural Networks (CNN)", IEEE, 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS), https://doi.org/10.1109/IACIS61494.2024.10721912, Hassan, India, 23-24 August 2024

[5] R. Vinayakumar; K. P. Soman, Prabaharan Poornachandran, "Long Short-Term Memory based Operation Log Anomaly Detection", 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 13-16 September 2017, https://doi.org/10.1109/ICACCI.2017.8125846, Udupi, India, pp. 236-242

[6] Zhaomin Chen, Chai Kiat Yeo, Bu Sung Lee, Chiew Tong Lau, "Autoencoder-based Network Anomaly Detection," 2018 Wireless Telecommunications Symposium (WTS), 17-20 April 2018, https://doi.org/10.1109/WTS.2018.8363930, IEEE, Phoenix, AZ, USA

[7] Tharindu Kumarage, Surangika Ranathunga, Chamal Kuruppu, Nadun De Silva, Malsha Ranawaka, "Generative Adversarial Networks (GAN) based Anomaly Detection in Industrial Software Systems", IEEE, 2019 Moratuwa Engineering Research Conference (MERCon), 03-05 July 2019, https://doi.org/10.1109/MERCon.2019.8818750, Moratuwa, Sri Lanka, pp. 43-48

[8] Kamiya Pithode, Pushpinder Singh Patheja, "A Study on Log Anomaly Detection using Deep Learning Techniques" IEEE, 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC), https://doi.org/10.1109/ICAAIC53929.2022.9793238, Salem, India, 09-11 May 2022

[9] Jin Tang, Wei Shuang, "Research on Network Traffic Anomaly Detection Method Based on Autoencoders," IEEE, 2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), https://doi.org/10.1109/AINIT61980.2024.10581422, Nanjing, China, 29-31 March 2024

[10] Mukesh Yadav, Dhirendra S Mishra, "Unique Log Parsing Framework for Enhanced Anomaly Detection in Network Security: Lauki Log Parser", International Journal of Communication Networks and Information Security (IJCNIS), 16(4), 890–905.

[11] Barry Siegel, "Industrial Anomaly Detection: A Comparison of Unsupervised Neural Network Architectures", IEEE Sensors Letters, Volume: 4 Issue: 8, https://doi.org/10.1109/LSENS.2020.3007880, August 2020

[12] Mukesh Yadav, Dr. Dhirendra S Mishra, "Identification Of Network Threats Using Live Log Stream Analysis", Second International Conference on the Paradigm shifts in Communication, Embedded Systems, Machine Learning and Signal Processing (PCEMS 2023), Visvesvaraya National Institute of Technology, Nagpur, India, Date Added to IEEE Xplore: 02 June 2023, D.O.I: 10.1109/PCEMS58491.2023.10136070, 05th - 06th April 2023.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Mukesh Yadav received her B.E. degree in 2013 and M.E. degree in 2016 in Computer Engineering from Pillai College of Engineering, New Panvel, University of Mumbai, Maharashtra, India. She is currently pursuing her Ph.D. degree (currently in her third year) from MPSTME, Mumbai of SVKM's NMIMS University, Mumbai, Maharashtra, India. Her research interests include Machine Learning, Network Security, Security Information and

Event Management, and Big data analytics.

Dr. Dhirendra Mishra received his B.E. degree in Computer Engineering from RAIT, Mumbai, Maharashtra, India in 2002, M.E. in Computer Engineering from TSEC, Mumbai, Maharashtra, India in 2008 and Ph.D. in Computer Engineering from NMIMS, Mumbai, Maharashtra, India in 2012. He is currently working as a Professor in the Department of Computer Engineering with MPSTME, NMIMS University, Mumbai, Maharashtra, India. His research interests include Image Processing - Image Database, Pattern matching, Image/Data Mining, Biometrics, and Data Analytics.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

List of Equations:

$$y = f(x) \tag{1}$$

Where, y is the input sequence for the next layer as given in equation 1.

$$y' = W_{temp} * y + b_{temp} \tag{2}$$

Where,

- y' is the output sequence.
- W_{temp} are the temporal convolution weights.
- b_{temp} is the bias term.
- $*$ denotes the convolution operation.

$$\alpha_t = \frac{exp(e_t)}{\sum_{t}^{T} exp(e_t)} \tag{3}$$

$$e_t = v_{temp}^{T} tanh\,(W_{temp} y'_t + b_{temp}) \tag{4}$$

$$\bar{y} = \sum_{t=1}^{T} \alpha_t \cdot y'_t \tag{5}$$

Where,

- α_t is the attention weight for time step t.
- e_t is the relevance score at time step t, which determines how important a particular time step is.
- T is the total number of time steps in the sequence.
- v_{temp} is the Learnable parameter vector used in attention scoring.
- W_{temp} is the weight matrix applied to the transformed input sequence.
- b_{temp} is the bias term in the transformation process.
- $\bar{y}$ is the final reweighted sequence based on temporal attention scores.
- y'_t is the transformed input sequence at time step t, which is an intermediate representation from the 1D convolution layer.

$$\beta_f = \frac{exp(e_f)}{\sum_{f=1}^{F} exp(e_f)} \tag{6}$$

$$e_f = v_{spatial}^{T} tanh\,(W_{spatial} y''_f + b_{spatial}) \tag{7}$$

$$z = \sum_{f=1}^{F} \beta_f \cdot y''_f \tag{8}$$

Where,

- β_f is the attention weight for feature f, which determines how important a specific feature is.
- e_f is the Relevance score of the feature f, representing how strongly it contributes to anomaly detection.
- F is the Total number of features in the sequence.
- $v_{spatial}$ is the learnable parameter vector for computing feature attention scores.
- $W_{spatial}$ is the learnable weight matrix applied to the input sequence.
- $b_{spatial}$ is the Bias term added to the transformation.
- y''_f is the Transformed input sequence for feature f, which is an intermediate representation from the Temporal Attention Layer.
- z is the final reweighted sequence after applying spatial attention.
- F is the Total no. of features used for analysis.

$$z' = TCN(z) \tag{9}$$

Where,

- z' is the transformed feature sequence after applying dilated causal convolutions in the TCN block.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- $TCN(.)$ represents the Temporal Convolutional Network, which consists of multiple stacked dilated convolution layers.
- z is the input feature sequence to the TCN Block, which comes from the Spatial Attention Layer.
- d is the **Dilation rate** in TCN, controlling how much the convolution expands over time steps.
- k is the **Kernel size**, defining the width of the convolutional filter applied over time.
- l is the **Number of layers in the TCN architecture**.

$$\ddot{z} = W_{FC} \cdot \dot{z} + b_{FC} \tag{10}$$

Where,

- W_{FC} is the **Weight matrix** for the fully connected layer.
- b_{FC} is the **Bias term** added to the transformed sequence.
- $\bar{z}$ is the **Final feature vector** after applying the fully connected transformation.
- $\dot{z}$ is the **Input sequence from the TCN Block**, carrying the learned representations.
- N_{FC} is the **Number of neurons in the fully connected layer**.

$$o = \sigma(\bar{z}) = \frac{1}{1+e^{-\bar{z}}} \tag{11}$$

$$o_i = \frac{e^{\bar{z}_i}}{\Sigma_{j=1}^{C} e^{\bar{z}_j}} \tag{12}$$

Where,

- o is the Final classification output, representing the probability of an anomaly (1) or normal log (0) for binary classification, or a class probability distribution for multi-class classification.
- $\sigma(\cdot)$ represents the sigmoid activation function (sigmoid for binary classification and softmax for multi-class classification).
- $\bar{z}$ is the **Final learned feature representation**, obtained from the **fully connected layer**.

$$S_{attack} = \sum_{i=1}^{N} C_i w_i \bullet \delta(f_i, S_i) \tag{13}$$

$$C_{attack} = \frac{S_{attack}}{Total_{features}} \tag{14}$$

$$S = \sum_{t=1}^{T} \alpha_t \sum_{f=1}^{F} \beta_f \left\| z_{t,f} - \hat{z}_{t,f} \right\| \tag{15}$$

Where,

- α_t is the Temporal attention weight, prioritizing time-sensitive anomalies by assigning higher relevance to critical time steps in log sequences.
- β_f is the Spatial attention weight, focusing on high-impact features, ensuring more critical attributes influence the anomaly score.
- $z_{t,f}$ is the Observed feature representation obtained from the TCN output for time step t and feature f.
- $\hat{z}_{t,f}$ is the Predicted normal feature representation, serving as a baseline reference for anomaly deviation.

For binary classification:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i log(o_i) + (1-y_i) log(1-o_i)] \tag{16}$$

Where,

- L is the **Binary cross-entropy loss**, representing the difference between predicted and actual labels.
- N is the Total number of training samples.
- y_i is True Label for sample i (1 for anomaly, 0 for normal log).
- o_i is predicted probability of sample i being anomalous.
- $log(o_i)$ ensures that confident wrong predictions receive a higher penalty i.e. Log probability of the predicted class when the sample belongs to an anomaly.

Journal of Information Systems Engineering and Management
2025, 10(34s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

- $(1 - y_i)log(1 - o_i)$ ensures loss is calculated for both classes
- $log(1 - o_i)$ is the **Log probability of the negative class** when the sample is normal.

For multi-class classification:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\, log(o_{i,c}) \tag{17}$$

Where,

- N is Total number of training samples
- C is Total number of possible classes
- y_{ic} is True label for sample i, 1 if the sample belongs to class c, otherwise 0
- o_{ic} is the predicted probability of the sample belonging to class c.
- L is the **Categorical cross-entropy loss**, which measures how well the model predicts the correct class probabilities.
- $log(o_{i,c})$ is the Log probability assigned to the correct class c.

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t \tag{18}$$

Where,

- θ_{t+1} is the updated model weights after applying the Adam optimization step.
- θ_t is model weights at time step t.
- m_t is the first moment estimate (mean of past gradients).
- v_t is the second moment estimate (variance of past gradients).
- α is the learning rate, controlling how much the weights are updated.
- ϵ is the smoothening term to prevent division by zero.
- β_1 is the Decay rate for first moment estimation (typically set to 0.9).
- β_2 is the Decay rate for second moment estimation (typically set to **0.999**).

$$CVSS = (B \times T \times E) \tag{19}$$

Where,

- B: Base score (captures exploitability, impact, and scope)
- T: Temporal score (patch availability, exploit maturity)
- E: Environmental score (customized risk assessment)

$$C_{MITRE} = \frac{\Sigma_{i=1}^{N} W_i S_i}{\Sigma_{i=1}^{N} W_i} \tag{20}$$

Where,

- W_i is the Weight assigned to each evidence source.
- S_i is the Score assigned to a specific ATT&CK technique.
- N is the Total number of evidence sources.