JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 19, No. 2, June 2025, pp. 37-45

A Deep Learning-Based Abnormal Detection Study Using Cloud Time Series

Eunjung Jo and Jeongjin Lee*
School of Computer Science and Engineering, Soongsil University, Seoul, Korea
Eunjung.jo1@gmail, leejeongjin@ssu.ac.kr

Myunghwa Kim
Department of IT Policy Management, Soongsil University, Seoul, Korea
beauhwa1@naver.com

Jongsub Lee
Department of IT Policy Management, Soongsil University, Seoul, Korea
minieno@gmail.com

Abstract
With the increasing complexity of cloud-native environments, timely detection of anomalies in system performance has become crucial. Cloud infrastructures produce large volumes of multivariate time-series data across various metrics including CPU, memory, and network I/O. Traditional statistical methods struggle with the nonlinear dependencies in such data, leading to a shift toward unsupervised machine learning techniques. This study investigates the effectiveness of two deep learning-based anomaly detection models—LSTM Autoencoder and GMM-GRU-VAE (GRU-based Gaussian Mixture Variational Autoencoder)—using real-world cloud infrastructure data. This study conducts an in-depth analysis of the architectures of the two models and compares their strengths, limitations, and performance in light of the specific requirements of cloud environments. By doing so, it aims to provide both theoretical and practical foundations for selecting the most suitable model for various cloud anomaly detection scenarios.

Category: Cloud Computing / High Performance Computing

Keywords: Observability; Telemetry; Anomaly detection; Deep learning; Autoencoder; LSTM

I. INTRODUCTION

With the expansion of cloud utilization, failure factors in cloud-native environments have become more diverse and complex, encompassing hybrid and multi-cloud, distributed containerized edge infrastructures, and microservices. In order to prevent failures in cloud environments, it is important to quickly understand what is happening within the infrastructure and promptly analyze the root causes of problems. This study aims to train and compare the performance of state-of-the-art deep learning models based on key time-series data obtained from actual cloud operational environments, in order to research effective time-series-based anomaly detection methods.

Open Access http://dx.doi.org/10.5626/JCSE.2025.19.2.37 http://jcse.kiise.org

Received 19 July 2024; **Accepted** 14 September 2025
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

II. RELATED WORK

A. Cloud Computing

Cloud computing is a type of computer system based on internet technology that allows access to resources such as servers, storage, applications, and networks anytime and anywhere. Cloud computing has the advantage of low initial setup costs and ease of use without maintenance costs. It is particularly convenient because users can access the amount of resources they need at the time they need them and only pay for what they use [1]. However, hardware or network failures are managed by the cloud service providers, while application errors or data input errors are the responsibility of the users, highlighting the importance of cloud monitoring to collect and analyze these issues. Cloud monitoring involves quickly understanding what is happening within the IT infrastructure to prevent prolonged outages and swiftly conducting root cause analysis during incidents. To achieve this, three types of data are monitored: logs, metrics, and traces, which numerically represent the overall behavior of systems, services, or network components over time. Logs are text records of events within applications generated when specific parts of the code are executed. They provide information on errors, timestamps, and relevant system details. Metrics are numerical representations used to track the overall behavior of systems, services, or network components over time, consisting of a timestamp, name, labels, and value, and are used to measure system runtime performance. Traces represent the complete journey of a request or operation as it moves through various components of a distributed system. Operators analyze this trace data to identify bottlenecks and quickly resolve performance issues.

B. Anomaly Detection

Anomaly detection is the identification of observations, events, or data that deviate from the norm or expected behavior and do not match other data sets. Distinguishing between "normal" and "anomalous" can be challenging. First, the variability within normal data can be significant, leading to the misclassification of normal samples as anomalies (Type I error) or failing to identify actual anomalies (Type II error). Preprocessing, normalization, and feature selection are potential means to reduce this variability and enhance detection capability [2-4]. Second, since anomalous events are rare, the training data is often highly imbalanced, and labeling most data sets is difficult. Thus, anomaly detection is often reduced to an unsupervised learning task aimed at learning a valid model for the majority of data points [5]. Lastly, anomalies can be highly diverse, making it challenging to learn a comprehensive model for them. Therefore, the approach involves learning a model for normal data and treating deviations from this model as anomalies. To prevent and respond to various issues such as security breaches, abnormal network activities, and service disruptions in the cloud, it is crucial to identify and detect anomalous behaviors or events that may occur in the cloud environment. This paper specifically aims to promptly detect cloud infrastructure failures by utilizing time-series data of infrastructure resources. We will train various deep learning models and compare their performance to propose the optimal model.

C. Anomaly Detection in Time-Series Data

1) Time-Series Decomposition

Time-series data can be decomposed into Seasonality, Trend, and Remainder [6]. Seasonality refers to patterns that repeat over a short period across the entire dataset, while Trend indicates the overall increasing or decreasing tendencies in the time series [6]. The irregular factors that cannot be explained by these two components are referred to as the Remainder. Identifying the types of time-series data and dividing them into seasonality, trend, and remainder is known as time-series decomposition.

2) Univariate vs. Multivariate

Time-series data can be classified as univariate or multivariate depending on the number of variables. A univariate time series $X = [x_1, x_2, \ldots, x_t]$ is an ordered set of real values. The length of X is equal to the number of real values T. For example, weather data such as current and past temperatures or past wind speeds, represented as time-temperature or time-wind speed with a single variable, are considered univariate time series. However, weather is influenced by factors like temperature, wind speed, and precipitation. When each optimal variable is not only temporal but also influenced by other optimal variables, it is considered a multivariate time series. Analyzing multivariate time series is more complex and challenging than univariate time series, hence the exploration of their analysis using deep learning.

3) Types of Time-Series Anomalies

There are three methods to classify types of time-series anomalies. First, a point anomaly, also known as a global outlier, occurs when data significantly deviates from the rest of the signal group. Second, a contextual anomaly refers to unexpected patterns in the context of the time-series data, known as conditional anomalies. Lastly, a group anomaly involves analyzing and determining whether multiple entities or groups exhibit an anomalous pattern collectively [7, 8]. Detecting group anomalies is more challenging than detecting point or contextual anomalies.

4) Time-Series Anomaly Detection Techniques

Time-series anomaly detection involves identifying the time steps where anomalies may occur [9]. Traditionally, statistical process control (SPC) methods such as CUSUM,

EWMA, and Shewhart charts were widely used solutions for monitoring the quality of industrial processes by finding deviations from the expected operational state [10]. However, these methods have limitations in handling multivariate data streams. To overcome this and leverage large volumes of data, machine learning techniques have started to be used [11]. In practice, due to the scarcity of labeled data, anomaly detection is mostly implemented using unsupervised learning models. Unsupervised learning methods can be categorized into four types: (1) linear model-based methods, (2) distance-based methods, (3) probability and density estimation-based methods, and (4) deep learning-based methods.

a) Linear model-based methods: An example of a linear model-based unsupervised anomaly detection method is principal component analysis (PCA) [12]. PCA is fundamentally a multivariate data analysis method that preserves significant variability information extracted from process measurements while reducing dimensions due to massive correlations. However, it is effective only for data with high correlations and assumes that the data follows a multivariate Gaussian distribution [13].

b) Distance-based methods: A widely used distance-based method is the k-nearest neighbor (KNN) algorithm, which calculates the average distance to the k-nearest neighbors and derives anomaly scores based on this distance [14]. The clustering-based local outlier factor (LOF) method is another example of distance-based approaches. This method uses a pre-defined anomaly score function to identify anomalies based on clustering, which is an enhanced version of the LOF method [15]. These distance-based methods perform better when there is prior knowledge about anomaly periods and the number of anomalies.

c) Probability and density estimation-based methods: Probability model-based and density estimation-based methods are improvements on distance-based approaches with a focus on data distribution. For instance, methods like angle-based outlier detection (ABOD) [16] and feature bagged (FB) [17] consider variable correlations in their data processing. However, these methods do not account for temporal correlations over time steps, thus performing poorly on multivariate time-series data.

d) Deep learning-based methods: This study focuses its analysis on two core architectures among reconstruction-based unsupervised anomaly detection models capable of effectively modeling temporal dependencies in time-series data. The first model is the LSTM-AE (long short-term memory autoencoder), which represents a standard approach in RNN-based anomaly detection. It employs long short-term memory networks as the fundamental components of both the encoder and decoder. This model is known to effectively learn temporal patterns in time-series data and reconstruct the unimodal normal state. The second model is the GMM-GRU-VAE (GRU-based Gaussian mixture variational autoencoder), an advanced architecture that extends the capabilities of the LSTM-AE in multiple aspects. First, it utilizes the gated recurrent unit (GRU), which is more computationally efficient than LSTM, to achieve model compactness and faster training. Furthermore, instead of the deterministic autoencoder, it is based on the probabilistic variational autoencoder (VAE), thereby modeling uncertainty and distributional properties of the data. Most importantly, unlike the conventional single Gaussian prior in the latent space, it adopts a Gaussian mixture model (GMM) prior, enabling the effective representation of complex multimodal normal states inherent in cloud data. This study provides an in-depth architectural analysis of these two models and compares their strengths, limitations, and performance in the context of the unique requirements of cloud environments. Through this comparative evaluation, it aims to establish both theoretical and practical foundations for selecting the optimal model in diverse cloud anomaly detection scenarios.

(i) LSTM-AE: The LSTM-AE is one of the most widely recognized and utilized reconstruction-based models for anomaly detection in time-series data. The core idea of this model is to learn the temporal patterns and dependencies inherent in normal time-series data, and to identify anomalous patterns that deviate from these by means of high reconstruction errors.

- **Comprehensive architectural analysis:** The LSTM-AE has a symmetric architecture composed of two primary components: an encoder and a decoder [18].
  - Encoder–decoder structure: The encoder processes the multivariate time-series sequence $X = \{x^{(1)}, x^{(2)}, ..., x^{(l)}\}$ and converts it into a fixed-size latent vector or context vector $h_E^{(l)}$, which summarizes the temporal context of the sequence [19]. This vector serves as a low-dimensional representation of the key features. The decoder initializes with this latent vector and reconstructs the input sequence in reverse order, $X = \{x^{(l)}, x^{(l-1)}, ..., x^{(1)}\}$[19]. Reverse reconstruction improves the model's ability to retain information from the beginning of the sequence [19].
  - LSTM cell mechanism: Both encoder and decoder are built upon LSTM cells. Standard RNNs suffer from the long-term dependency problem, especially vanishing gradients, as sequence length increases [20]. LSTM addresses this issue by introducing a cell state, which carries information across the sequence with minimal modification. Information is regulated by three gates [20]—Forget gate, selects information to discard from the previous cell state; Input gate, determines new information to add based on the current input and hidden state; and Output gate, controls the hidden state (output) using the updated cell state. These gates enable LSTM to capture long-term temporal dependencies effectively [21].

- Training process: The model is trained in an unsupervised manner using only normal time-series data [19]. The objective is to minimize reconstruction error between the input sequence $\check{X}$ and the reconstructed sequence $\acute{X}$. Mean squared error (MSE) or mean absolute error (MAE) is typically used as the loss function [18]. Through training, the model learns to encode and decode complex spatiotemporal patterns of normal data.

- **Anomaly scoring and detection:** After training, the LSTM-AE evaluates previously unseen data to identify anomalies.
  - Reconstruction error vector: When test time-series data is input to the model, the LSTM-AE outputs its reconstruction. For each time step i the reconstruction error $e^{(i)} = |x^{(i)} - x'^{(i)}|$ is computed, where $x^{(i)}$ and $x'^{(i)}$ denote the original and reconstructed data points, respectively [19]. The sequence of these errors forms the reconstruction error vector. Normal data, being similar to the patterns learned during training, exhibit low errors, whereas anomalous data deviate from learned patterns and are expected to produce high errors [19].
  - Threshold setting: To determine anomalies, a threshold must be established. A common approach is to compute reconstruction errors on a separate validation set of normal data and analyze their distribution. For instance, the threshold can be set at the 95th or 99th percentile of the error distribution. Alternatively, assuming the errors follow a normal distribution, the mean and standard deviation can be used to define a threshold at a certain distance from the mean (e.g., 3σ) [19].
  - Anomaly detection: If the reconstruction error at any time step of the test data exceeds the predefined threshold, that point is classified as an anomaly [22]. This approach can detect not only point anomalies but also collective anomalies, where contiguous segments exhibit consistently high errors [21].

- **Strengths and limitations of LSTM-AE:** The LSTM-AE has established itself as a robust baseline model for anomaly detection across various industrial domains due to its structural clarity and capability to process time-series data.
  - Strengths:
    (1) Temporal pattern learning: Thanks to the LSTM cell architecture, the model can effectively capture the temporal order and long-term dependencies in the data [21]. This is particularly advantageous for analyzing time-series with complex patterns such as seasonality and trends; and
    (2) Unsupervised learning: The model can be directly applied to large unlabeled datasets, reducing data preparation costs and providing potential to detect previously unseen types of anomalies [19].
  - Limitations:
    (1) single-mode assumption: The LSTM-AE inherently attempts to map all normal data into a single continuous and smooth latent space. However, cloud systems may have multiple statistically distinct normal states, such as weekday vs. weekend patterns or peak service hours vs. batch processing periods. The LSTM-AE does not explicitly model the multimodal nature of the data, which can lead to learning overly broad "normal" boundaries and reduced detection precision [23];
    (2) Paradox of strong generalization: While deep learning models offer powerful representation and generalization capabilities, these can sometimes be detrimental to anomaly detection. If the model is overly flexible, it may reconstruct unseen anomalous patterns relatively well, resulting in over-generalization. In this case, the reconstruction error difference between normal and anomalous data is reduced, making detection more difficult [23];
    (3) Dependence on normal data definition and quality: The performance of an LSTM-AE is ultimately determined by how "normal" data is defined and curated. In dynamic cloud environments, where normal behavior can change due to new service deployments, user inflows, or external events, the composition and management of the training dataset are as critical as the model architecture itself [21]; and
    (4) Sensitivity to anomaly type: The sequential reconstruction mechanism of LSTM may react differently to various types of anomalies. Gradual degradation in system states may lead the model to adapt to slowly changing patterns, maintaining low reconstruction errors. Consequently, slightly abnormal states from yesterday could be considered "normal" today. In contrast, abrupt anomalies, such as sudden CPU spikes or network delays, sharply deviate from prior normal patterns, resulting in high reconstruction errors and easier detection [21]. Therefore, when applying this model, it is essential to consider the characteristics of the anomalies to be detected and acknowledge the model's inherent limitations.

(ii) GMM-GRU-VAE

While the LSTM-AE is effective in modeling a single-mode normal state, the GMM-GRU-VAE represents a more advanced architecture designed to capture the complex and multimodal normal states commonly observed in cloud environments. This model combines computational efficiency, probabilistic representation, and multimodal modeling capability to enhance the precision of anomaly detection.

- Need for probabilistic and multimodal modeling:

Operational data in cloud systems do not follow a single statistical distribution. For example, CPU and network usage patterns during peak user interaction hours and patterns observed during overnight large-scale data backups or batch processing both fall within the "normal" category, yet their statistical characteristics are entirely different [23]. Data containing multiple distinct operational modes like this is referred to as multimodal data. Traditional reconstruction-based models, such as the LSTM-AE, attempt to represent such multimodal distributions within a single continuous latent space. This is analogous to approximating a mountain range with multiple peaks using a single gentle hill, which fails to capture the unique characteristics of each normal state. Consequently, the boundaries between states become ambiguous, making subtle anomalies difficult to detect. To address this issue, it is necessary to adopt probabilistic generative models that directly model the complex underlying distributions of the data. VAEs learn the data distribution, and GMMs explicitly assume that this distribution is composed of multiple sub-distributions, enabling precise modeling of multi-modal data.

- Comprehensive architectural analysis: The GMM-GRU-VAE architecture is formed through the synergistic combination of three key components: GMM, GRU, and VAE.
  - GRU: The GRU is a variant of the LSTM that improves computational efficiency through a simpler recurrent unit structure [20]. Unlike LSTM, which uses three gates (forget, input, and output), the GRU merges the forget and input gates into a single update gate and introduces a reset gate to control how much of the previous hidden state is incorporated [23]. This structural simplification reduces the number of parameters compared to LSTM, accelerating model training and reducing memory usage [24]. In cloud environments, where large-scale time-series data must be periodically retrained, this efficiency gain directly contributes to TCO reduction. Several studies report that GRU achieves performance comparable to LSTM while enabling faster training. Therefore, replacing LSTM with GRU is not merely a technical substitution but a strategic architectural response to the key cloud constraint of cost efficiency.
  - VAE: The VAE is a generative model that learns the probabilistic distribution of data. Unlike conventional autoencoders, whose encoders compress input data into a single point in the latent space, a VAE encoder outputs the parameters of a probability distribution (typically mean μ and variance σ^2) that the data is assumed to follow in the latent space [25]. The decoder samples latent vectors z from this estimated distribution to reconstruct the original data. This sampling enables probabilistic reconstruction rather than deterministic reconstruction, producing a smooth and continuous latent space and endowing the model with generative capabilities [25]. The VAE loss function consists of two components—Reconstruction loss, measures the difference between the original and reconstructed data, encouraging accurate reconstruction; and Regularization term, the Kullback-Leibler divergence is computed between the encoder-estimated latent distribution and the prior distribution, which prevents overly complex latent space structures and enforces well-structured representations. The sum of these terms forms the evidence lower bound (ELBO), and the VAE is trained to maximize the ELBO [26].
  - GMM prior: The most innovative aspect of the GMM-GRU-VAE is its assumption about the prior distribution in the VAE latent space. Standard VAEs assume a single standard Gaussian prior ($N(0, I)$), implying that all data should form one cluster near the origin. GMM-GRU-VAE relaxes this assumption by modeling the latent space prior as a mixture of K Gaussian distributions (GMM) [23]. Each Gaussian component has its own mean and variance, forming K clusters in the latent space. This approach explicitly enables the model to learn the inherent multimodality of the data. For instance, "high-load normal" states in a cloud system may map to one Gaussian cluster, while "low-load normal" states map to another. The model can thus learn the statistical characteristics of each normal state individually, allowing much more precise and refined definitions of the boundary between normal and anomalous states [23].
- Integration and anomaly detection: These three components operate synergistically in an integrated manner.
  - Operation of the integrated model: Input time-series data first passes through the GRU-based encoder, which extracts temporal features. Based on these features, the encoder estimates the probability that the data belongs to each of the K Gaussian clusters and the corresponding latent distribution (μ_k, σ_k^2) within the assigned cluster. A latent vector z is then sampled from this multimodal latent space and fed into the GRU-based decoder, which probabilistically reconstructs the original time series [23].
  - Anomaly scoring and detection: GMM-GRU-VAE uses reconstruction probability instead of reconstruction error as the anomaly score. When the model reconstructs a specific input data x, it calculates how high the probability is that the reconstruction result is similar to x.
  - Normal data: Since normal data exists within the learned multimodal distribution, it has a high reconstruction probability.
  - Anomalous data: Since anomalous data does not belong to any of the learned normal distributions, it

has a low reconstruction probability. This approach enables more statistically robust anomaly detection by defining anomalies as 'low probability of occurrence' rather than 'large error'. The detection process, similar to LSTM-AEs, involves setting a threshold α based on the reconstruction probability distribution of the training data, and classifying data as anomalous if its reconstruction probability is lower than α. In conclusion, GMM-GRU-VAE is an advanced architecture that directly addresses the complexity and multimodality of cloud time series data. By securing computational efficiency through GRU, acquiring probabilistic representation through VAE, and capturing multimodality through GMM, it can expect more precise and robust anomaly detection performance than LSTM-AEs, which assume a unimodal distribution.

III. EXPERIMENTS AND RESULTS

This study is based on the results of a deep learning benchmark for multivariate time series data published by Dennis Wagner in April 2023 [27]. The TimeSeAD [27] library provides a general training and evaluation framework tailored for deep learning-based methods, several analysis tools for datasets and methods, and a large collection of architectural components, methods, and baselines. The architecture is implemented on top of PyTorch, offering reusable building blocks that allow for extensive customization. This setup enables the rapid prototyping of multiple models and the adaptation of individual components to any configuration. Using these elements, we compared model-specific performance through a fast (linear-time) implementation of time series precision and recall. The SMD [28] and Exathlon [29] datasets were used as benchmark data. We then validated the performance of the high-performing LSTM-AE and GMM-GRU-VAE deep learning models, identified through the anomaly detection library TimeSeAD, using time series data collected from a real-world cloud operating environment.

A. Research Procedure

The methodology of this study is organized into three primary stages: data collection, data preprocessing, and model training and evaluation. First, the data collection phase involved acquiring time-series data from a production cloud operating environment. To construct a representative dataset, both normal operational data and anomalous data were gathered, specifically capturing the periods surrounding system failures. Key infrastructure-level performance metrics—including those related to CPU, memory, storage, and network—were collected using the New Relic monitoring platform. Second, a data preprocessing step was conducted to standardize the feature scales across the dataset. For this purpose, min-max normalization was applied. This technique rescales all feature values to a uniform range of [0, 1], ensuring that each metric contributes proportionally to the model's training process without bias from differing magnitudes. Finally, the preprocessed data was utilized to train and evaluate the LSTM-AE and GMM-GRU-VAE deep learning models. The comparative performance of these models was assessed using standard evaluation metrics. These included Precision, defined as the ratio of true anomalies among the instances identified as anomalous by the model; Recall, the ratio of true anomalies correctly identified by the model; and the F1-score, which serves as a balanced measure of a model's overall performance by calculating the harmonic mean of precision and recall.

B. Data Collection

In this study, data were collected from a real AWS cloud infrastructure environment during the deployment and operation of a mobile application. Following the deployment of a specific application, a service failure occurred, and key performance metrics of the cloud infrastructure were recorded around this event. The collected metrics include system-level indicators such as CPU utilization and memory usage, network-level indicators such as network traffic measured in bytes per second, and storage-level indicators such as data read latency and the number of data read operations. Data was gathered at 5-minute intervals over a period of 7 days, totaling 1,859 samples for each metric. Data from 6 days of normal operation, totaling 1,577 records, were used for training, while 287 records, which included anomalous data with a significant increase in resource usage on the 7th day, were used as test data. The collected data is visualized as shown in Fig. 1.

In addition, an example of five sampled data records is presented in Table 1.

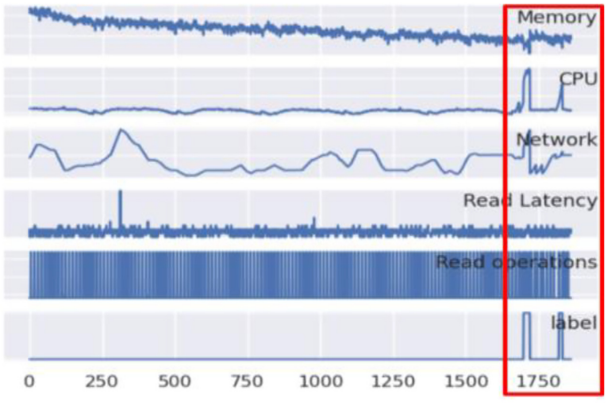

Fig. 1. Collected time-series data.

Table 1. Sampling of the 5 values

Time	Memory	CPU	Network	Read latency	# Read operations	Label
6/30/22 12:40	7031483904	6.168	14.506	0.000000	69.996	0
6/30/22 12:45	7031730176	6.154	14.594	0.000160	70.004	0
6/30/22 12:50	7034144768	6.168	14.682	0.000080	70.001	0
6/30/22 12:55	7033368064	6.134	14.771	0.000080	69.998	0
6/30/22 13:00	7031554048	6.198	14.859	0.000000	133.993	0
6/30/22 13:05	7031991808	6.205	14.953	0.000160	70.002	0
6/30/22 13:10	7033629184	6.218	15.047	0.000160	69.990	0
6/30/22 13:15	7032757248	6.196	15.135	0.000080	70.009	0
6/30/22 13:20	7032702976	6.156	15.224	0.000080	69.990	0

Table 2. Experimental environment and model components

Component	Value
Training data	Normal data (1577, 5)
Test data	Normal data (253, 5) Abnormal data (34, 5)
Batch size	10
Epochs	100
Latent vector	10
Optimizer	Adam
Loss function	Mean absolute error
CPU & GPU	NVIDIA Tesla K80 GPU, 12 GB
ML API	Keras 2.10.0

C. Experimental Environment

In this study, the implemented models were run in a Google Colab notebook environment using NVIDIA Tesla K80 GPU with 12 GB RAM, and the development language used was Python 3. The experimental environment and model-related elements are detailed in Table 2.

D. Comparison of AE and GMM-GRU-VAE Model Performance

The performance of the AE and GMM-GRU-VAE models was evaluated and is summarized in Table 3.

The evaluation results indicate that the LSTM-AE model achieved an area under the precision-recall curve (AUPRC) of 0.7378 and a BestTS F1-score of 0.9694 with a testing time of 12 minutes. The GMM-GRU-VAE model exhibited a competitive F1 performance with a BestTS F1-score of 0.9648, although its AUPRC was slightly lower at 0.7229. Additionally, it required the longest testing time of 13 minutes. Overall, both models were found to be suitable for cloud environ- ments, particularly for time-series anomaly detection in cloud infrastructure where high predictive accuracy and rapid processing are essential.

Table 3. Comparison of AE and GMM-GRU-VAE model performance

Model	Precision	BestTS F1-score	Duration (min)
LSTM-AE	0.7378	0.9594	12
GMM-GRU-VAE	0.7229	0.9648	13

IV. DISCUSSION

In conclusion, while both the LSTM-AE and GMM-GRU-VAE models demonstrated high performance with BestTS F1-scores exceeding 0.95, their practical application in a cloud environment hinges on a strategic trade-off between operational efficiency and detection sophistication. The LSTM-AE serves as a highly efficient and robust baseline, proving most effective for systems with unimodal, consistent operational patterns where rapid deployment and minimal computational overhead are critical. Its architectural simplicity facilitates faster prototyping and makes it a pragmatic choice under resource constraints. Conversely, the GMM-GRU-VAE is superior in complex, multimodal environments, such as cloud services with distinct usage patterns (e.g., weekday vs. weekend). By modeling multiple normal states within its latent space, it excels at identifying subtle, ambiguous anomalies that might otherwise be missed. This enhanced precision, however, comes at the cost of greater computational complexity and longer training times. Therefore, the ultimate model selection should be guided by the specific characteristics of the target data and operational demands. The decision framework rests on balancing the need for detection granularity against the available computational and temporal resources, ensuring the chosen model aligns with both the data's complexity and the organization's

AIOps maturity.

CONFLICT OF INTEREST

The authors have declared that no competing interests exist.

REFERENCES

1. A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, pp. 1045-1051, 2010. https://doi.org/10.1093/comjnl/bxp080
2. C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham, Switzerland: Springer, 2017. https://doi.org/10.1007/978-3-319-47578-3
3. I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, 2003.
4. S. Garcia, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Cham, Switzerland: Springer, 2015. https://doi.org/10.1007/978-3-319-10247-4
5. L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K. R. Muller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756-795, 2021. https://doi.org/10.1109/JPROC.2021.3052449
6. Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, and M. S. Hossain, "Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6348-6358, 2020.
7. V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, article no. 15, 2009. https://doi.org/10.1145/1541880.1541882
8. O. A. Ekle and W. Eberle, "Anomaly Detection in Dynamic Graphs: A Comprehensive Survey," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 8, article no. 192, 2024. https://doi.org/10.1145/3669906
9. V. Chandola, V. Mithal, and V. Kumar, "Comparative evaluation of anomaly detection techniques for sequence data," in *Proceedings of 2008 8th IEEE International Conference on Data Mining*, Pisa, Italy, 2008, pp. 743-748. https://doi.org/10.1109/ICDM.2008.151
10. B. Sun, P. B. Luh, Q. S. Jia, Z. O'Neill, and F. Song, "Building energy doctors: an SPC and Kalman filter-based method for system-level fault detection in HVAC systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 1, pp. 215-229, 2014. https://doi.org/10.1109/TASE.2012.2226155
11. D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, "A survey of deep learning-based network anomaly detection," *Cluster Computing*, vol. 22(Suppl 1), pp. 949-961, 2019. https://doi.org/10.1007/s10586-017-1117-8
12. S. Li and J. Wen, "A model-based fault detection and diagnostic methodology based on PCA method and wavelet transform," *Energy and Buildings*, vol. 68, pp. 63-71, 2014. https://doi.org/10.1016/j.enbuild.2013.08.044
13. X. Dai and Z. Gao, "From model, signal to knowledge: a data-driven perspective of fault detection and diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 4, pp. 2226-2238, 2013. https://doi.org/10.1109/TII.2013.2243743
14. F. Angiulli and C. Pizzuti, "Fast outlier detection in high dimensional spaces," in *Principles of Data Mining and Knowledge Discovery*. Heidelberg, Germany: Springer, 2002, pp. 15-27. https://doi.org/10.1007/3-540-45681-3_2
15. M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, Dallas, TX, USA, 2000, pp. 93-104. https://doi.org/10.1145/342009.335388
16. H. P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Las Vegas, NV, USA, 2008, pp. 444-452. https://doi.org/10.1145/1401890.1401946
17. A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, Chicago, IL, USA, 2005, pp. 157-166. https://doi.org/10.1145/1081870.1081891
18. W. Skaf and T. Horvath, "Denoising architecture for unsupervised anomaly detection in time-series," in *New Trends in Databases and Information Systems*. Cham, Switzerland: Springer, 2022, pp. 178-187. https://doi.org/10.1007/978-3-031-15743-1_17
19. P. Malhotra, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff, "LSTM-based encoder-decoder for multi-sensor anomaly detection," 2016 [Online]. Available: https://arxiv.org/abs/1607.00148.
20. S. Kaddu, "LSTM vs GRU: understanding recurrent neural networks for sequence data," 2025 [Online]. Available: https://medium.com/@sushantkadduwork/lstm-vs-gru-understanding-recurrent-neural-networks-for-sequence-data-bdfb1f89756a.
21. Z. Hong, "Anomaly detection in time series data using LSTM autoencoders," 2024 [Online]. Available: https://medium.com/@zhonghong9998/anomaly-detection-in-time-series-data-using-lstm-autoencoders-51fd14946fa3.
22. M. K. Hossain and A. Barua, "Comparative study of transformer & LSTM architectures for anomaly detection in multivariate time series data," M.S. thesis, University of Stavanger, Norway, 2025.
23. Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, "Multidimensional time series anomaly detection: a GRU-based Gaussian mixture variational autoencoder approach," *Proceedings of Machine Learning Research*, vol. 95, pp. 97-112, 2018.
24. R. Bansal, "When to use GRUs over LSTMs?," 2025 [Online]. Available: https://www.analyticsvidhya.com/blog/2025/03/lstms-and-grus/.
25. D. Kyle, "VAE for time series: generate realistic sequential data with this easy-to-train model," 2024 [Online]. Available: https://towardsdatascience.com/vae-for-time-series-1dc0fef4bffa.
26. H. Wang and H. Zhang, "An anomaly detection method for multivariate time series data based on variational autoencoders and association discrepancy," *Mathematics*, vol. 13, no. 7, article no. 1209, 2025. https://doi.org/10.3390/math13071209

27. D. Wagner, T. Michels, F. C. Schulz, A. Nair, M. Rudolph, and M. Kloft, "TimeSeAD: benchmarking deep multivariate time-series anomaly detection," 2023 [Online]. Available: https://openreview.net/forum?id=iMmsCI0JsS.
28. Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage, AK, USA, 2019, pp. 2828-2837. https://doi.org/10.1145/3292500.3330672
29. V. Jacob, F. Song, A. Stiegler, B. Rad, Y. Diao, and N. Tatbul, "Exathlon: a benchmark for explainable anomaly detection over time series," 2020 [Online]. Available: https://arxiv.org/abs/2010.05073.

Eunjung Jo https://orcid.org/0000-0002-2950-374X

Eunjung Jo is a Ph.D. student in Computer Science and Engineering from the Soongsil University, Seoul, Korea. Her research interests are abnormal detection, Cloud observability, deep learning.

Jeongjin Lee https://orcid.org/0000-0001-9676-271X

Jeongjin Lee is currently a professor with the School of Computer Science and Engineering and the director of Deep Imaging Laboratory at Soongsil University, South Korea. His research interests include deep learning, AI-based mobile healthcare, interactive vision techniques in augmented reality, and biomedical image processing.

Myunghwa Kim

Myunghwa Kim received Ph.D. degree in Department of IT Policy Management from the Soongsil University, Seoul, South Korea in 2021. She works for Hanwha Systems, Seoul, Korea. Her research interests are Data Science, deep learning.

Jongsup Lee

Jongsup Lee received Ph.D. degree in Department of IT Policy Management from the Soongsil University, Seoul, South Korea in 2022. He works for AIBiz Artificial Intelligence R&D Center, Seoul, Korea. His research interests include cloud, bigdata, deep learning.