ICeTEA: Mixture of Detectors for Metric-Log Anomaly Detection

Junxiang Wang[1], Xu Zheng[1,2], Zhengzhang Chen[1], Masanao Natsumeda[3], Jun Nishioka[3], Dongsheng Luo[2], Haifeng Chen[1]
[1]NEC Laboratories America, [2]Florida International University, [3]NEC Corporation

Abstract

Anomaly detection is essential for identifying unusual system behaviors and has wide-ranging applications, from fraud detection to system monitoring. In web servers, anomalies are typically detected using two types of data: metrics (numerical indicators of performance) and logs (records of system events). While correlations between metrics and logs in real-world scenarios highlight the need for joint analysis, which is termed the "metric-log anomaly detection" problem, it has not been fully explored yet due to inherent differences between metrics and logs. In this paper, we propose ICeTEA, a novel system for metric-log anomaly detection that integrates three detectors: a metric-log detector based on a multimodal Variational Autoencoder (VAE), and two individual metric and log detectors. By leveraging the ensemble technique to combine outputs of these detectors, ICeTEA enhances the effectiveness and robustness of metric-log anomaly detection. Case studies demonstrate two key functionalities of ICeTEA: data visualization and rankings of contributions to anomaly scores. Experiments demonstrate that our proposed ICeTEA accurately detects true anomalies while significantly reducing false positives.

ACM Reference Format:
Junxiang Wang[1], Xu Zheng[1,2], Zhengzhang Chen[1], Masanao Natsumeda[3], Jun Nishioka[3], Dongsheng Luo[2], Haifeng Chen[1]. 2025. ICeTEA: Mixture of Detectors for Metric-Log Anomaly Detection. In . ACM, New York, NY, USA, 6 pages.

1 Introduction

Anomaly detection is a classic unsupervised learning problem investigated for decades with the goal of finding unusual patterns or behaviors that deviate from expected system performance [4, 6, 15–17, 23]. It encompasses a wide range of applications, including fraud detection in financial transactions, cyber intrusion detection, and machinery fault diagnosis [1–3, 5, 7, 8, 19, 21, 27, 28]. In the application of web servers, anomalies are usually detected by two data modalities: metrics and logs. Metrics are measurable indicators of system performance and health, like CPU usage, memory consumption, and network bandwidth [32]. They also include performance measures such as latency and throughput, as well as business metrics like customer satisfaction and revenue growth. Logs, in contrast, are records of system events and activities generated by the Internet of Things (IoT), such as operating systems, networks, and applications. Both metrics and logs are monitored over time to maintain server performance, troubleshoot issues, and ensure smooth operation [32].

Most existing studies focus on either metrics or logs for anomaly detection [1, 14, 18, 29]. However, their combinations have not been fully explored yet due to their inherently different natures: metrics are regularly generated, represented as numerical and continuous values, and can be multivariate; time-series anomalies are usually detected as outliers whose behaviors deviate from normal time-series patterns (*e.g.*, peaks or troughs). Logs, however, are event-driven (*i.e.*, a large volume of logs can be generated within milliseconds), recorded as text, and can be converted to discrete event types [7, 8, 29]. Log anomalies can be detected by either unexpected sequences (*e.g.*, unusual repetition of the same event) or error messages. Despite their structural differences, metrics and logs are often correlated, making their joint analysis important for accurate anomaly detection, defined as the so-called "metric-log anomaly detection" problem. For example, a spike in CPU usage might be normal during the launch of a large application but can be unusual if no corresponding log entry is recorded. As another example, an increase in the volume of router events could be normal if the network traffic rises but would indicate a hardware issue if the traffic remains stable. Therefore, the metric-log anomaly detection problem demands attention and extensive investigation from machine learning researchers.

To bridge this gap, we propose ICeTEA, a unified system for metric-log anomaly detection. ICeTEA adopts an ensemble architecture comprising three components: a metric-based detector, a log-based detector, and a metric-log detector. The metric-log detector extends the Variational Autoencoder (VAE) to a multimodal setting, capturing cross-modal interactions to detect joint anomalies. In contrast, the metric and log detectors specialize in unimodal anomalies. The final anomaly scores are derived by aggregating the outputs from all three detectors. Our key contributions are:

- We study the unexplored metric-log anomaly detection problem, which holds significant research and application value.
- We propose ICeTEA, a novel system for metric-log anomaly detection based on input metrics and logs.
- We develop a platform to provide detailed anomaly detection analysis for web server performance monitoring.
- We conduct experiments on real-world business data to show that the ICeTEA outperforms existing anomaly detection models.

2 Related Work

Existing works on anomaly detection in IT operations can be mainly categorized into metric-based and log-based approaches.

Metric Anomaly Detection: Metric anomaly detection is a classical and challenging task in time series analysis that has been studied

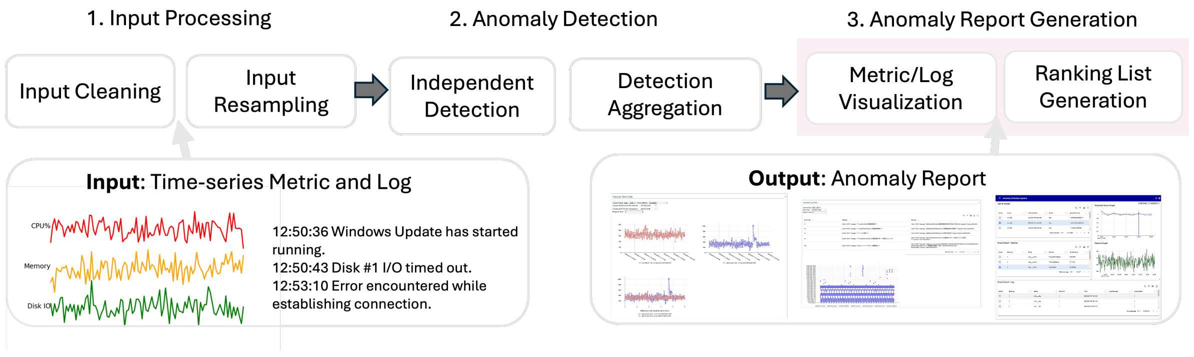


Figure 1: Pipeline of the ICeTEA system: it consists of input processing, anomaly detection, and anomaly report generation.

for decades [1, 24]. The challenges include unpredictable patterns due to the inherent dynamicity of metric data, distribution shifts caused by configuration changes, and the scarcity of labeled anomalies [28]. For multivariate metric data, previous anomaly detection techniques are classified into forecasting-based and reconstruction-based methods [31]. Forecasting-based methods detect anomalies based on forecasting errors [13]. In contrast, reconstruction-based methods learn representations of metric data by reconstructing the original metric input using latent variables [25]. In our proposed ICeTEA model, the metric detector is based on forecasting methods, while the metric-log detector employs reconstruction methods.

Log Anomaly Detection: Log files are crucial for monitoring computer systems, capturing both normal operations (*e.g.*, process starts/stops, VM restarts, and user access) and unexpected events (*e.g.*, process failures, availability issues, and security incidents). Consequently, log anomaly detection models aim to learn normal system behavior and identify deviations for human review. Previous techniques primarily used deep learning models to learn normal system behaviors such as DeepLog [9] and LogRobust [30].

While metric and log anomaly detection have been extensively studied individually, their integration remains underexplored. This gap stems from two key challenges: the inherently different modalities of metrics and logs, and the difficulty in modeling their interactions. To address these challenges, we propose ICeTEA, which adopts a multimodal VAE architecture. ICeTEA encodes metrics and logs into a shared latent space, where their interactions are captured through a dedicated fusion module.

3 The ICeTEA System

The diagram of our ICeTEA system is shown in Figure 1. First, the metric and log inputs are processed. Next, the processed inputs are fed into the anomaly detection model, which consists of three detectors, and their outputs are aggregated to make final detections using the ensemble technique. Finally, an anomaly report is generated based on the detection results.

3.1 Input Processing

3.1.1 Input Cleaning. For metric data, all metrics are normalized to ensure a consistent data range, and missing values are imputed with default values. For log data, the Drain parser is trained on the training log data to construct a parse tree [11]. The parse tree

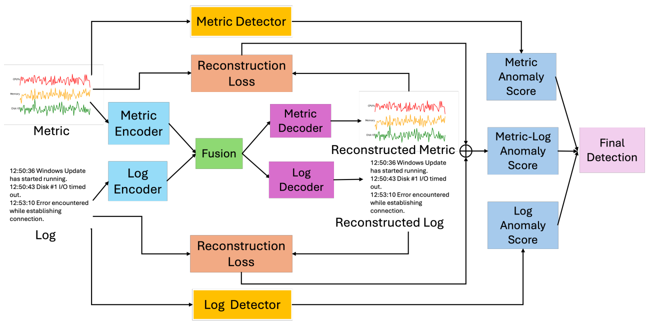


Figure 2: The overview of the anomaly detection model: final detections are aggregated by outputs from the metric-log detector, the metric detector, and the log detector.

maps each test log to a template, assigning the corresponding log template ID as an event type for the log. This event type forms one part of the log input for the anomaly detection model.

In addition to event types, the other part of the log input consists of token IDs obtained from log messages. Specifically, log messages are tokenized using the HuggingFace tokenizer library[1], and the tokens are mapped to token IDs.

3.1.2 Input Resampling. Due to the inherently different frequencies of metrics and logs (*i.e.*, regular versus event-driven), resampling is performed to align the two data modalities. The entire time period is divided into evenly spaced overlapping time segments. Multiple metric records are averaged if they fall within a time segment. Thus, each time segment contains constant records for the metrics, while the log sequences vary in length.

3.2 Anomaly Detection

In the step of anomaly detection, the processed inputs are fed into the anomaly detection model, as shown in Figure 2. The anomaly detection model consists of three detectors: the metric-log detector, the metric detector, and the log detector.

[1]https://github.com/huggingface/tokenizers/blob/main/docs/source-doc-builder/index.mdx

3.2.1 *Metric-Log Detector.*

The metric-log detector adopts a multi-modal VAE architecture, comprising three key components: the encoder, the fusion module, and the decoder. The encoder includes both a metric encoder and a log encoder, which are responsible for encoding their respective inputs into latent representations. Similarly, the decoder consists of a metric decoder and a log decoder, which reconstruct the metric and log data from the fused representations. The metric-log detector is the core of the ICeTEA system, and its architecture will be illustrated in detail in the next section.

3.2.2 *Metric Detector.*

Aside from the metric-log detector, two individual metric and log detectors are also utilized to detect anomalies based on the single modality, providing further support for the metric-log detector. The thresholds of the two detectors are defined as the 95th percentile anomaly scores of all training samples.

Specifically, the metric detector is set to the Peak Over Threshold (POT) model [26], which is a statistical approach based on extreme value theory. The POT model learns the behavior of extreme events by fitting them into the generalized Pareto distribution, on which the anomaly score is based.

3.2.3 *Log Detector.*

For the log detector, a combination of a frequency-based detector and a simple Principal Component Analysis (PCA) model is employed: the frequency-based log detector checks whether log time-series exhibit periodic patterns, and the PCA model transforms a sequence of event types into continuous feature representations (*e.g.*, occurrence counts or Term Frequency-Inverse Document Frequencies (TF-IDFs) of event types). The anomaly score is the reconstruction error between the reconstructed representations and the actual representations. The anomaly score of the log detector is the average of the normalized anomaly scores from the frequency-based log detector and the PCA detector (*i.e.*, here the normalized anomaly score is the original anomaly score divided by the threshold).

3.2.4 *Detection Aggregation.*

Outputs from three independent detectors are aggregated to make final detections, and a simple majority-voting strategy is applied: a sample is flagged as an anomaly if two out of three detectors label it as an anomaly. We believe that our ICeTEA model is effective and robust not only because of the ensemble technique but also due to the diversity of our detectors: our metric-log detector is a nonlinear reconstruction-based model, our metric detector is a probabilistic approach, and our log detector is based on frequency and low-dimensionality. This means that they can potentially capture different types of anomalies (*e.g.*, abnormal frequencies of event types, unexpected log sequence patterns, and extreme metric values).

3.3 Anomaly Report Generation

In order to make our anomaly report instructive and insightful, the anomaly report will cover two parts: data visualization and ranking lists of contributions to anomaly scores. In the current version, we simply visualize time-series metric values and the patterns of log event types by selected time ranges, and the same time-series metrics with different time ranges can be compared. The anomaly scores for specific metrics and logs are also demonstrated respectively. Moreover, ranking lists of contributions to anomaly scores are also provided. In other words, these ranking lists will display the top metrics or logs that contribute the most to anomaly scores. To measure such contributions, gradient-based attribution methods [22] are utilized to calculate how metrics and logs contribute to the anomaly scores (*e.g.*, the gradients of the reconstruction loss with respect to metric/log input embeddings in the metric-log detector). The top ten metrics/logs will be shown in the ranking lists for further anomaly analysis and investigation.

3.4 Architecture of the Metric-Log Detector

The metric-log detector is the focus of the ICeTEA system. In this section, we detail its multi-modal VAE architecture.

1. Encoder. Representations of a metric x_i and a log m_j are denoted as $\tilde{x}_i$ and $\tilde{m}_j$, respectively, as shown below:

$$\tilde{x}_i = g_1(\overline{t}_i + \overline{x}_i), \quad \tilde{m}_j = g_2(\overline{\tau}_j + \overline{u}_j + \overline{m}_j).$$

where $\overline{t}_i$ and $\overline{x}_i$ represent the metric time and value encodings from Equations (1) and (3), respectively. Similarly, $\overline{\tau}_j$, $\overline{u}_j$, and $\overline{m}_j$ represent the log time, event, and message encodings from Equations (2), (4), and (5), respectively. Both g_1 and g_2 are transformer encoders. The details are described below:

(a). Time Representation. The time representation is computed using sinusoidal functions that exhibit smooth periodic oscillations. For the metric timestamp t_i, we define

$$c^t_{i,k} = \cos\left(\frac{2\pi t_i}{2^i}\right), s^t_{i,k} = \sin\left(\frac{2\pi t_i}{2^i}\right).$$

and for the log timestamp τ_j, we have

$$c^\tau_{j,k} = \cos\left(\frac{2\pi \tau_j}{2^j}\right), s^\tau_{j,k} = \sin\left(\frac{2\pi \tau_j}{2^j}\right).$$

Then time representations are calculated as follows:

$$\overline{t}_i = f_t(t_i) = W_t[c^t_{i,1}, s^t_{i,1}, c^t_{i,2}, s^t_{i,2}, \dots, c^t_{i,K}, s^t_{i,K}]^\top, \tag{1}$$

$$\overline{\tau}_j = f_\tau(\tau_j) = W_\tau[c^\tau_{j,1}, s^\tau_{j,1}, c^\tau_{j,2}, s^\tau_{j,2}, \dots, c^\tau_{j,K}, s^\tau_{j,K}]^\top. \tag{2}$$

where $\overline{t}_i$ and $\overline{\tau}_j$ are the time representations of t_i and τ_j, respectively. W_t and W_τ are learned projection matrices for the metric and log timestamps, and K is the number of sinusoidal function pairs.

(b). Metric Representation. The metric representation $\overline{x}_i$ is:

$$\overline{x}_i = g_3(x_i). \tag{3}$$

where g_3 is a transformer encoder.

(c). Event and Message Representations. The event and message representations are tokenized and embedded using the token embedding function $\boldsymbol{e}(\bullet)$. These embeddings can be learned from scratch or initialized using a pretrained tokenizer:

$$\overline{u}_j = \boldsymbol{e}(u_j), \tag{4}$$

$$\overline{m}_j = g_4(\boldsymbol{e}(m_j)). \tag{5}$$

where $\overline{u}_j$ and $\overline{m}_j$ are the event and message representations of u_j and m_j, respectively. g_4 is a transformer encoder.

2. Fusion Module. The goal of the fusion component is to integrate the metric representation $\tilde{x}_i$ and the log representation $\tilde{m}_j$ into a joint context representation:

$$h = g_5\left(\{\tilde{x}_i\}_{i=1} \circ \{\tilde{m}_j\}_{j=1}\right).$$

where g_5 is a fusion transformer encoder, $\circ$ is the concatenation along the time dimension, and h is a contextual representation.

Here h is used to compute the mean μ and the standard deviation σ of the posterior distribution $q(z|X,M)$, which will be utilized to sample a latent representation z for metric and log reconstructions:

$$[\mu;\sigma] = Wh + b, \quad q(z|X,M) = \mathcal{N}(z;\mu,\sigma I).$$

where W and b are the weight and bias of the linear layer.

3. Decoder. Given a sampled latent $z \sim q(z|X,M)$, the goal of the decoder component is to reconstruct the metric and the log from z, which are achieved by transformer decoders G_1 and G_2. Specifically, the reconstructed metric is shown as follows:

$$\hat{x}_i = G_1(z, \{u_j\}_{j=1}).$$

where z and u_j are aligned by the cross-attention in G_1 to match the metric representation in z. Similarly, the reconstructed log is

$$\hat{u}_j^{'} = G_2(z, \{x_i\}_{i=1}) \in [0,1], \quad \hat{u}_j = \arg\max \hat{u}_j^{'}.$$

where $\hat{u}_j^{'}$ is the probability distribution of the reconstructed event type, and $\hat{u}_j$ is the reconstructed event type. Noticeably, we do not reconstruct message m_j due to the posterior collapse problem [20].

4. Objective and Anomaly Score. We denote $\hat{X} = \{t_i, \hat{x}_i\}_{i=1}^{T}$ and $\hat{M} = \{\tau_j, \hat{u}_j, m_j\}_{j=1}^{N}$ as the reconstructed metric and log sequences, respectively. Then the objective is formulated mathematically as follows:

$$\mathcal{L} = \mathcal{L}_{\text{met}}(X,\hat{X}) + \alpha\mathcal{L}_{\log}(M,\hat{M}) + \beta\mathcal{L}_{\text{reg}}(X,M).$$

where $\mathcal{L}_{\text{met}}(X,\hat{X})$ is the reconstruction loss of the metric, which is achieved by the Mean Squared Error (MSE). $\mathcal{L}_{\log}(M,\hat{M})$ is the reconstruction loss of the log, which is achieved by the Cross-Entropy loss. $\mathcal{L}_{\text{reg}}(X,M)$ is the regularization loss, which is achieved by the Kullback–Leibler (KL) divergence. $\alpha > 0$ and $\beta > 0$ are two hyperparameters to balance three terms.

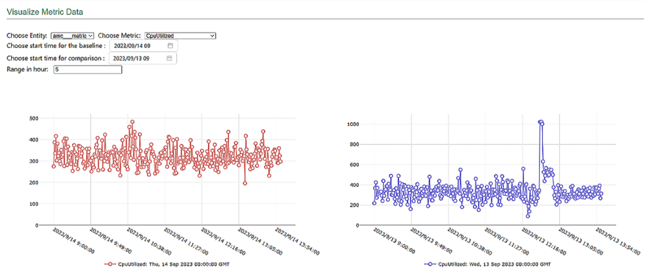
The anomaly score is defined as the sum of two reconstruction losses. The threshold is defined as the mean plus three standard deviations of anomaly scores of all training samples.

4 Demonstration Scenarios

This section demonstrates two major functionalities of the ICeTEA system: visualization of metrics and logs (i.e. Figure 3) and rankings of contributions to anomalies scores (i.e. Figure 4).

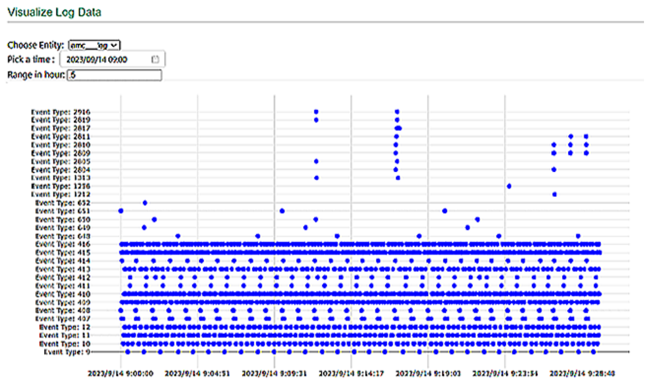
1. Data Visualization. Figure 3 demonstrates visualizations of metrics and logs. Specifically, for the metric data, different entities (*e.g.*, webservers 1 and 2), metrics, and data ranges can be freely chosen. Moreover, metric visualization supports comparisons between different start times. As Figure 3(a) shows, red and purple curves represent plots of CPU utilization starting from 13:09 and 14:09 on September 13th, respectively. The comparison shows a peak in the purple curve, while no obvious peaks are observed in the red curve, suggesting that the peak in the purple curve may indicate a potential anomaly. The visualization of logs is similar to that of metrics, where log visualization explores occurrence patterns of different event types.

2. Rankings of Contributions to Anomaly Scores. Figure 4 demonstrates ranking lists of all anomalies, anomaly metrics, and anomaly logs. The tab "List of events" shows all anomalies above the thresholds. When different anomalies are selected, the blue vertical line in the right-hand figure, "Anomaly Score Graph," changes correspondingly. Similarly, when different metrics in the tab "Event Detail - Metrics" are selected, the metric curves in the right-hand "Metric Graph" figure are adjusted accordingly. The tabs "Event Detail - Metrics" and "Event Detail - Log" rank anomaly metrics and logs based on their contributions to anomaly scores, respectively. The tab "Event Detail - Metrics" lists rankings, entities, metrics, and contributions to anomaly scores, whereas the tab "Event Detail - Log" lists rankings, entities, event types, times, log messages, and contributions to anomaly scores.



Choose Entity: amc___metric Choose Metric: CpuUtilized
Choose start time for the baseline : 2023/09/14 09
Choose start time for comparison : 2023/09/13 09
Range in hour: 5

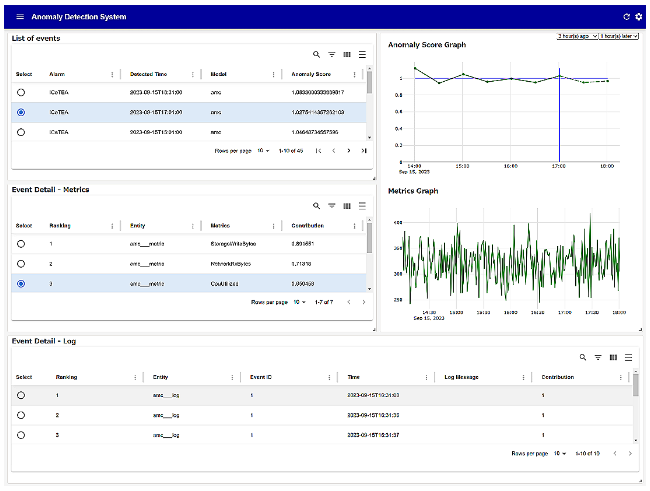
(a). Metrics.



Choose Entity: amc___log
Pick a time: 2023/09/14 09:00
Range in hour: 5

(b). Logs.

Figure 3: Visualizations of metrics and logs.



List of events

Select	Alarm	Detected Time	Model	Anomaly Score
○	ICeTEA	2023-09-15T18:31:00	amc	1.0833000333889817
◉	ICeTEA	2023-09-15T17:01:00	amc	1.0275414357262103
○	ICeTEA	2023-09-15T15:01:00	amc	1.04648734557596

Rows per page 10 1-10 of 45

Event Detail - Metrics

Select	Ranking	Entity	Metrics	Contribution
○	1	amc___metric	StorageWriteBytes	0.891551
○	2	amc___metric	NetworkRxBytes	0.71318
◉	3	amc___metric	CpuUtilized	0.650456

Rows per page 10 1-7 of 7

Event Detail - Log

Select	Ranking	Entity	Event ID	Time	Log Message	Contribution
○	1	amc___log	1	2023-09-15T16:31:00		1
○	2	amc___log	1	2023-09-15T16:31:36		1
○	3	amc___log	1	2023-09-15T16:31:37		1

Rows per page 10 1-10 of 10

Figure 4: All anomalies and ranking lists of top metrics and logs contributing the most to anomaly scores.

5 Experiments

Settings	#Metric(train/test)	#Log(train/test)
A	7,781/1,943	67,417/7,798
B	8,334/3,439	69,691/12,692

Table 1: Statistics of two settings in the dataset.

5.1 Experiment Setup

5.1.1 Dataset Description: We evaluate our proposed ICeTEA on a business dataset collected from a cloud-service platform.[2] The dataset is divided into two settings due to configuration changes. The statistics of the two settings are shown in Table 1. The seven metrics include CPU utilization, memory utilization, network metrics for bytes transmitted and received, the number of bytes written to and read from storage, and utilization of ephemeral storage. Log files include event types, log templates, and log messages. The total number of log event types is 3,416. Only one anomaly was detected in both settings.

5.1.2 Hyperparameter Settings: Two tuning parameters, α and β, were set to 1. The Adam optimizer was used to optimize the cross-joint VAE. The learning rate was set to 0.001. The batch size was set to 8. The dimension of the latent variable, d, was set to 32. The sequence length T was set to 30 minutes. Majority voting was used to ensemble the results.

5.1.3 Comparison Methods and Evaluation Metrics: Two time series detectors, LSTM [12] and POT [26], and two log detectors, PCA [10] and DeepLog [9], were used for performance comparison. Three evaluation metrics were used to assess model performance: True Alarm (*i.e.*, True Positive), which occurs when a model correctly detects an actual anomaly; False Alarm (*i.e.*, False Positive), which occurs when a model incorrectly classifies a normal event as an anomaly; and Missing Alarm (*i.e.*, False Negative), which occurs when a model fails to detect an actual anomaly.

Setting A	Metrics		Log		Ours
Methods	LSTM	POT	DeepLog	PCA	ICeTEA
True Alarm ↑	0	1	1	1	1
False Alarm ↓	3	4	13	17	5
Missing Alarm ↓	1	0	0	0	0
Setting B	**Metrics**		**Log**		**Ours**
Methods	LSTM	POT	DeepLog	PCA	ICeTEA
True Alarm ↑	0	0	0	1	1
False Alarm ↓	2	6	5	19	9
Missing Alarm ↓	1	1	1	0	0

Table 2: Performance of all methods: the ICeTEA captures true anomalies while minimizing false alarms.

[2]Some details are omitted due to commercial confidentiality here and below.

5.2 Experimental Results

Table 2 illustrates that our proposed ICeTEA framework consistently detects all true anomalies in both settings A and B, while maintaining a moderate false alarm rate and zero missed anomalies. In contrast, LSTM has the lowest false alarm rate but fails to detect any true anomalies, making it unreliable despite its low false alarm rate. POT shows a good balance with relatively low false alarms but misses one anomaly in setting B. DeepLog performs well in setting A with one true alarm and 13 false alarms, but it performs poorly in setting B, missing one true anomaly while generating five false alarms. PCA detects true anomalies in both settings, but its high false alarm rate significantly reduces its effectiveness.

6 Conclusion

Leveraging cross-modal information from both metrics and logs is crucial for enhancing the accuracy and robustness of web server anomaly detection. In this paper, we propose ICeTEA, a novel ensemble-based system that combines a multimodal VAE to model metric-log interactions with standalone metric-only and log-only detectors. A case study demonstrates two core capabilities of ICeTEA: anomaly visualization and attribution of anomaly scores to contributing factors. Experimental results show that ICeTEA accurately detects true anomalies while significantly reducing false positives.

For future work, we plan to enhance ICeTEA by incorporating Large Language Models (LLMs) for automated log analysis and conducting more comprehensive evaluations.

References

[1] Blázquez-García, A., Conde, A., Mori, U., and Lozano, J. A. A review on outlier/anomaly detection in time series data. *ACM computing surveys (CSUR) 54*, 3 (2021), 1–33.

[2] Cai, L., Chen, Z., Luo, C., Gui, J., Ni, J., Li, D., and Chen, H. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In *Proceedings of the 30th ACM international conference on Information & Knowledge Management* (2021), pp. 3747–3756.

[3] Cao, C., Chen, Z., Caverlee, J., Tang, L.-A., Luo, C., and Li, Z. Behavior-based community detection: Application to host assessment in enterprise information networks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1977–1985.

[4] Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. *ACM computing surveys (CSUR) 41*, 3 (2009), 1–58.

[5] Chen, T., Tang, L.-A., Sun, Y., Chen, Z., and Zhang, K. Entity embedding-based anomaly detection for heterogeneous categorical events. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), IJCAI'16, AAAI Press, p. 1396–1403.

[6] Chen, Z., Hendrix, W., and Samatova, N. F. Community-based anomaly detection in evolutionary networks. *Journal of Intelligent Information Systems 39*, 1 (2012), 59–85.

[7] Dong, B., Chen, Z., Tang, L.-A., Chen, H., Wang, H., Zhang, K., Lin, Y., and Li, Z. Anomalous event sequence detection. *IEEE Intelligent Systems 36*, 3 (2020), 5–13.

[8] Dong, B., Chen, Z., Wang, H., Tang, L.-A., Zhang, K., Lin, Y., Li, Z., and Chen, H. Efficient discovery of abnormal event sequences in enterprise security systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), pp. 707–715.

[9] Du, M., Li, F., Zheng, G., and Srikumar, V. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security* (2017), pp. 1285–1298.

[10] Dunia, R., Qin, S. J., Edgar, T. F., and McAvoy, T. J. Identification of faulty sensors using principal component analysis. *AIChE Journal 42*, 10 (1996), 2797–2812.

[11] He, P., Zhu, J., Zheng, Z., and Lyu, M. R. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)* (2017), IEEE, pp. 33–40.

[12] Hochreiter, S. Long short-term memory. *Neural Computation MIT-Press* (1997).

[13] Hundman, K., Constantinou, V., Laporte, C., Colwell, I., and Soderstrom, T. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (2018), pp. 387–395.

[14] Landauer, M., Onder, S., Skopik, F., and Wurzenberger, M. Deep learning for anomaly detection in log data: A survey. *Machine Learning with Applications 12* (2023), 100470.

[15] Li, Y., Chen, Z., Zha, D., Zhou, K., Jin, H., Chen, H., and Hu, X. Autood: Automated outlier detection via curiosity-guided search and self-imitation learning. *arXiv preprint arXiv:2006.11321* (2020).

[16] Li, Y., Chen, Z., Zha, D., Zhou, K., Jin, H., Chen, H., and Hu, X. Automated anomaly detection via curiosity-guided search and self-imitation learning. *IEEE Transactions on Neural Networks and Learning Systems 33*, 6 (2021), 2365–2377.

[17] Li, Y., Chen, Z., Zha, D., Zhou, K., Jin, H., Chen, H., and Hu, X. Autood: Neural architecture search for outlier detection. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2021), IEEE, pp. 2117–2122.

[18] Li, Y., Liu, Y., Wang, H., Chen, Z., Cheng, W., Chen, Y., Yu, W., Chen, H., and Liu, C. Glad: Content-aware dynamic graphs for log anomaly detection. In *2023 IEEE International Conference on Knowledge Graph (ICKG)* (2023), IEEE, pp. 9–18.

[19] Lin, Y., Chen, Z., Cao, C., Tang, L.-A., Zhang, K., Cheng, W., and Li, Z. Collaborative alert ranking for anomaly detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), pp. 1987–1995.

[20] Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. Understanding posterior collapse in generative latent variable models, 2019.

[21] Luo, C., Chen, Z., Tang, L.-A., Shrivastava, A., Li, Z., Chen, H., and Ye, J. Tinet: learning invariant networks via knowledge transfer. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 1890–1899.

[22] Nguyen, Q. P., Lim, K. W., Divakaran, D. M., Low, K. H., and Chan, M. C. Gee: A gradient-based explainable variational autoencoder for network anomaly detection. In *2019 IEEE Conference on Communications and Network Security (CNS)* (2019), IEEE, pp. 91–99.

[23] Padmanabhan, K., Chen, Z., Lakshminarasimhan, S., Ramaswamy, S. S., Richardson, B. T., et al. Graph-based anomaly detection. *Practical Graph Mining with R* (2013), 311.

[24] Pang, G., Cao, L., and Aggarwal, C. Deep learning for anomaly detection: Challenges, methods, and opportunities. In *Proceedings of the 14th ACM international conference on web search and data mining* (2021), pp. 1127–1130.

[25] Park, D., Hoshi, Y., and Kemp, C. C. A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder. *IEEE Robotics and Automation Letters 3*, 3 (2018), 1544–1551.

[26] Siffer, A., Fouque, P.-A., Termier, A., and Largouet, C. Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (2017), pp. 1067–1075.

[27] Wang, D., Chen, Z., Fu, Y., Liu, Y., and Chen, H. Incremental causal graph learning for online root cause analysis. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining* (2023), pp. 2269–2278.

[28] Wang, J., Bai, G., Cheng, W., Chen, Z., Zhao, L., and Chen, H. Pond: Multi-source time series domain adaptation with information-aware prompt tuning. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2024).

[29] Wang, Z., Chen, Z., Ni, J., Liu, H., Chen, H., and Tang, J. Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (2021), pp. 3726–3734.

[30] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2019), pp. 807–817.

[31] Zhao, H., Wang, Y., Duan, J., Huang, C., Cao, D., Tong, Y., Xu, B., Bai, J., Tong, J., and Zhang, Q. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE international conference on data mining (ICDM)* (2020), IEEE, pp. 841–850.

[32] Zheng, L., Chen, Z., He, J., and Chen, H. Mulan: multi-modal causal structure learning and root cause analysis for microservice systems. In *Proceedings of the ACM Web Conference 2024* (2024), pp. 4107–4116.