Towards Multi-System Log Anomaly Detection

Boyang Wang[1]*, **Runqiang Zang**[2]*, **Hongcheng Guo**[1]†,
Shun Zhang[1], **Shaosheng Cao**[3], **Donglin Di**[4], **Zhoujun Li**[1]†

[1]Beihang University, [2]Renmin University of China,
[3]Xiaohongshu Inc. [4]Tsinghua University
{wangboyang, hongchengguo}@buaa.edu.cn, caoshaosheng@xiaohongshu.com

Abstract

Despite advances in unsupervised log anomaly detection, current models require dataset-specific training, causing costly procedures, limited scalability, and performance bottlenecks. Furthermore, numerous models lack cognitive reasoning abilities, limiting their transferability to similar systems. Additionally, these models often encounter the **"identical shortcut"** predicament, erroneously predicting normal classes when confronted with rare anomaly logs due to reconstruction errors. To address these issues, we propose **MLAD**, a novel **M**ulti-system **L**og **A**nomaly **D**etection model incorporating semantic relational reasoning. Specifically, we extract cross-system semantic patterns and encode them as high-dimensional learnable vectors. Subsequently, we revamp attention formulas to discern keyword significance and model the overall distribution through vector space diffusion. Lastly, we employ a Gaussian mixture model to highlight rare word uncertainty, optimizing the vector space with maximum expectation. Experiments on real-world datasets demonstrate the superiority of MLAD [1].

1 Introduction

Logs play a vital role in system maintenance by recording operations and outcomes that can reveal abnormal behavior. Data-driven log analysis techniques have been widely used to automatically detect anomalies in system behavior (Du et al., 2017a; Chandola et al., 2009; Meng et al., 2019a; Guo et al., 2024). However, most log anomaly detection models are designed for a single system, following a **"one model for one system"** approach (Yu et al., 2024; Su et al., 2024; Guo et al., 2023b), as shown in Fig.1(a). This siloed training limits generalization and fails to capture patterns common across different systems.

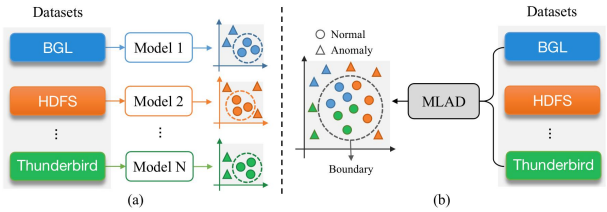


Figure 1: Multi-system log anomaly detection task. (a) Existing models learn separate decision bounds for different object logs. (b) We model the multi-system log distributions so that a single bound can detect anomalies.

Integrating log data from multiple systems offers the potential to uncover anomalous patterns hidden in isolated datasets. In practice, though, new systems often lack sufficient log data to train reliable models, leading to delayed deployment and missed anomalies (Landauer et al., 2024). Existing methods also tend to overlook deeper semantic features (Wang et al., 2017; Guo et al., 2023a) shared across systems. As a result, similar anomalies, such as repeated error or warning messages, occurring across different system logs may remain undetected.

To address these challenges, we introduce MLAD—a generalized log anomaly detection model designed for multiple systems, as illustrated in Fig.1(b). MLAD learns a unified decision boundary to classify normal and abnormal events across all systems, rather than maintaining separate models per system. Unlike reconstruction-based methods that can misclassify anomalies due to the "identical shortcut" (You et al., 2022) effect, where rare abnormal logs are reconstructed too well and thus labeled normal (Yao et al., 2024), MLAD avoids this pitfall. It employs a deflationary transformation of the vector space to amplify distinctions between normal and abnormal log samples. This transformation clusters similar log entries together

*Equal contribution.
†Corresponding author.
[1]We provide code and dataset: https://github.com/LolerPanda/Multi-System-Log-Anomaly-Detection

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 83–91
July 28-30, 2025

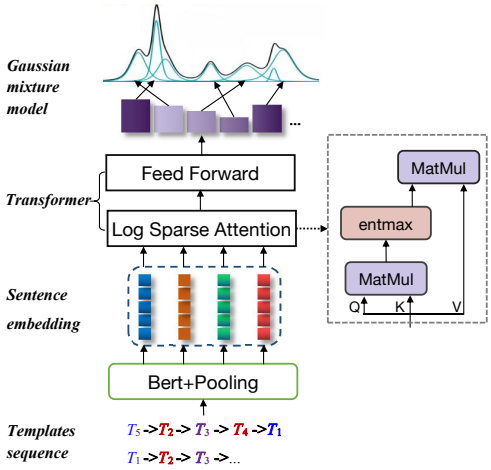


Figure 2: The architecture of our proposed MLAD.

while pushing normal and anomalous logs farther apart, making anomalies easier to isolate.

MLAD combines a Transformer (Vaswani et al., 2017) and a Gaussian Mixture Model (GMM) (Zong et al., 2018; Vilnis and McCallum, 2015) in a unified architecture. The Transformer component learns rich semantic representations of log sequences, capturing context and reducing reconstruction error (Ma et al., 2024). The GMM component functions as a robust probabilistic classifier for distinguishing normal from anomalous log instances. We train the Transformer and GMM jointly, which minimizes encoding errors and yields more precise anomaly detection. Our contributions are:

- **Multi-System Anomaly Detection.** Introduces a new model for detecting anomalies across multiple systems, overcoming the limitations of traditional one-model-per-system methods.
- **Hybrid Transformer–GMM Architecture.** Integrates Transformers with GMMs, jointly learning semantic log representations while preserving clear separation between normal and abnormal events.
- **Addressing "Identical Shortcut".** Mitigates the identical shortcut problem by transforming the vector space, which effectively separates abnormal samples from normal ones based on learned distance relationships.
- **Improved Performance.** Extensive experiments on real-world log datasets show that MLAD outperforms state-of-the-art anomaly detection approaches.

2 Related Work

Traditional log anomaly detection methods use manual rules or statistical approaches like SVD (Mahimkar et al., 2011), ARIMA (Zhang et al., 2005), and variants. While effective to some extent, these models are noise-sensitive and parameter-sensitive (Chen et al., 2023a), limiting practical applications. Recent models leverage deep learning networks (Du et al., 2017b; Han and Yuan, 2021; Zhang et al., 2022). Du et al. proposed DeepLog (Du et al., 2017b), an LSTM architecture for identifying anomalous log message sequences. LogAnomaly (Meng et al., 2019b) improves on DeepLog by using log sequence embedding rather than template sequences. Zhang et al. introduced LogRobust (Zhang et al., 2019), an attention-based Bi-LSTM model for anomaly detection. Huang et al. (Huang et al., 2020) employed hierarchical transformers to model both log template sequences and parameter values. LogBERT (Guo et al., 2021) predicts masked log keys, positioning normal logs close together in embedding space.

3 MLAD

We introduce MLAD, as depicted in Figure 2, a hybrid model trained on log sequences using unsupervised tasks to automatically detect anomalies.

3.1 Problem Definition

System logs contain unstructured messages with fields like timestamp and severity, exhibiting sequential patterns and semantic relationships. We extract templates using the Drain parser (He et al., 2017), as shown in Figure 3. For example, the BGL log template "exception syndrome register: <>" comes from "exception syndrome register: 0x008000", where <> indicates variable parameters. We map each template to a key, creating sequences $T = [T_1, T_2, \ldots, T_i, \ldots, T_N]$, where $T_i \in \mathbb{T}$ is the template key at position i, and $\mathbb{T}$ is the set of N template sequences from system logs. Our model identifies abnormal template sequences by training only on normal log sequences.

3.2 Feature Extractor

For semantic template relation learning, we use pre-trained Sentence-Bert (Reimers and Gurevych, 2019) to obtain template sequence representations and MEAN pooling (Reimers and Gurevych, 2019) to compress vectors into fixed dimension d embeddings. This prevents information loss from log parsing errors and facilitates single- or multi-system log

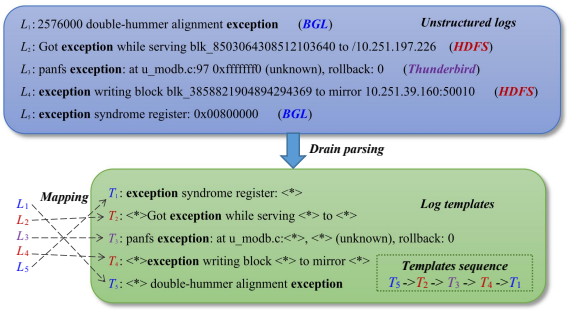


Figure 3: Log processing flow.

fusion. Each sequence $T \in \mathbb{R}^{l \times d}$ forms template vectors in high-dimensional vector spaces.

3.3 Sparse Log Self-attention

Self-attention encodes template sequence vectors by associating words based on pairwise similarity function $f(\cdot,\cdot)$. We use linear projection T to acquire query Q, key K, and value V, and adopt Scaled DotProduct Attention (Vaswani et al., 2017) with sparse transformation:

$$\begin{aligned} Q,K,V &= TW_q, TW_k, TW_v, \\ h &= \text{Attention}(Q,K,V) = \alpha\text{-entmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \end{aligned} \tag{1}$$

where learnable weights $\{W_q, W_k, W_v\} \in \mathbb{R}^{d \times d}$, $\sqrt{d_k}$ is a scaling factor, Q of Eq. 1 is the query representation matrix, K is the key matrix, and V is the values matrix. The sparse transformation (Peters et al., 2019) increases attention weight differences to accurately learn keyword embedding vectors. Weight values follow the function:

$$\begin{aligned} \alpha\text{-entmax}(x) &= \underset{p \in \Delta^{d-1}}{\text{argmax}} \left(p^\top x + H_\alpha^\top(p)\right), \\ H_\alpha^\top(p) &= \begin{cases} \frac{1}{\alpha(\alpha-1)} \sum_j \left(p_j - p_j^\alpha\right), & \alpha \neq 1, \\ H^\top(p), & \alpha = 1, \end{cases} \end{aligned} \tag{2}$$

where $H_\alpha^\top(p)$ is Tsallis α-entropies (Tsallis, 1988), parameterized by scalar $\alpha > 1$. From Eq. 2, the softmax function equals 1-entmax, with Shannon and Gini entropy as regularizers. Parameter α controls shape and sparsity, as shown in Figure 4. When $1 < \alpha < 2$, the function produces sparse probability distribution with smooth corners. Traditional softmax (Bridle, 1989) has small slope at 0.5, making weight values dense around 0.5 when word count is high, reducing word differentiation and hindering keyword identification.

3.4 Feed-Forward Network

We apply a fully connected Feed-Forward Network (FFN) to each position to add nonlinearity and consider latent dimension interactions. FFN includes

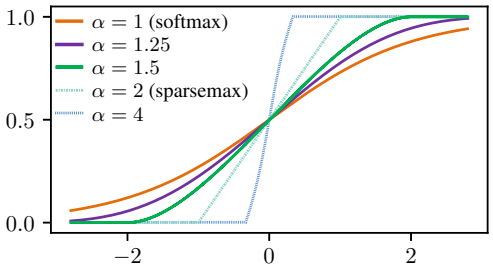


Figure 4: An illustration of the α-entmax function in a two-dimensional space.

two linear transformations with Continuously Differentiable Exponential Linear Unit (CeLU) (Barron, 2017) activation:

$$\begin{aligned} \text{FFN}(h) &= \text{CeLU}(hW_1 + b_1)W_2 + b_2, \\ \text{CeLU}(x) &= \max(0,x) + \min(0, \alpha * (\exp(x/\alpha) - 1), \end{aligned} \tag{3}$$

where W_1, W_2, b_1, and b_2 are parameters. CeLU$(\cdot)$ provides smoother transition than ReLU$(\cdot)$, improving generalization. We use normalization and dropout to prevent overfitting.

3.5 Gaussian Mixture Model

For anomaly detection, we use GMM with Expectation-Maximization (EM) algorithm (Huber and PeterJ, 2009). GMM excels in label-free learning but struggles with large-scale data (Zong et al., 2018). Transformers encode large-scale data and learn high-dimensional features effectively. By adjusting Multi-head Attention layers, we reduce vector space dimensions, addressing the big data limitations of GMM. Transformers face binary classification challenges when loss approaches zero. The α-entmax function maps normal log words to an identity matrix, potentially misclassifying similar abnormal logs. Replacing the decoder of Transformer with GMM enhances vector space differentiation through iterative sample reconstruction, improving normal/abnormal sample distinction. In the EM algorithm's E-step, GMM prior defines distributions on reconstruction function $f(h)$ using Gaussian distributions K. We compute probability $\hat{\phi}_k$ that hidden vector h_i belongs to the k-th Gaussian:

$$\begin{aligned} \hat{y} &= \text{entmax}(hWh + b), \\ \hat{\phi}_k &= \textstyle\sum_{i=1}^{N} \frac{\hat{y}_{ik}}{N}, \end{aligned} \tag{4}$$

where $\hat{y}_i$ indicates anomaly class probability and adjusts the attenuation parameter. Each Gaussian has mean μ (sample location) and covariance Σ. Sentence-BERT uses cosine similarity but overlooks uncertainty (Reimers and Gurevych, 2019) from low-frequency words. In multi-system log detection, imbalance between normal/abnormal sam-

ples exacerbates this issue. We integrate covariance matrix into the loss function to capture uncertainty differences, calculating mean μ and covariance Σ as:

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{y}_{ik} h_i}{\sum_{i=1}^{N} \hat{y}_{ik}}, \quad \hat{\Sigma}_k = \frac{\sum_{i=1}^{N} \hat{y}_{ik} (h_i - \hat{\mu}_k)(h_i - \hat{\mu}_k)^\top}{\sum_{i=1}^{N} \hat{y}_{ik}}. \tag{5}$$

In the M-step, we substitute estimated parameters to find the extreme value of the lower bound function, updating parameter values when the derivative equals 0. Sample energy is inferred as:

$$E(h_i) = -\log\left(\sum_{k=1}^{K} \hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(h_i - \hat{\mu}_k)^\top \hat{\Sigma}_k^{-1}(h_i - \hat{\mu}_k)\right)}{\sqrt{|2\pi\hat{\Sigma}_k|}}\right). \tag{6}$$

During testing, sample energy is estimated directly and high-energy samples above threshold are predicted as anomalies.

3.6 Objective Function

For N samples, the objective function is:

$$\text{Loss} = \frac{1}{N}\sum_{i=1}^{N} L(y_i - \hat{y}_i)^2 + \frac{\lambda_1}{N}\sum_{i=1}^{N} E(h_i) + \lambda_2 P\left(\hat{\Sigma}\right), \tag{7}$$

where y is ground truth, with $\lambda_1 = 0.1$ and $\lambda_2 = -0.005$. This function has three components. $L\left(y_i - \hat{y}_i\right)$ quantifies discrepancy between predictions and actual values, reflecting Transformer prediction accuracy. $E\left(h_i\right)$ represents GMM normal probability modeling, minimizing energy for normal samples and maximizing for abnormal ones. $P(\hat{\Sigma})$ addresses the "identical shortcut" issue by incorporating keyword uncertainty into the loss function, with higher uncertainty indicating higher anomaly probability.

4 Experiment

We first describe our experimental setup, compare MLAD with state-of-the-art baselines, and analyze components' roles and multisystem the impact of datasets.

4.1 Datasets and Setting

Experiments use public BGL, HDFS, and Thunderbird datasets (Oliner and Stearley, 2007), detailed in Table 1. For fair comparison, all models use 100-dimensional embeddings, Adam optimizer with 0.001 learning rate, 0.5 dropout rate on NVIDIA A100 (80G), 512 batch size, and 30 maximum epochs.

	BGL	HDFS	Thunderbird
# Log sequences	2,780,580	5,856,609	9,975,120
# Templates	138 (35)	44 (25)	1,291 (243)
# Words	987	118	6,546
# Anomalies	248,560	10,109	2,456,660
# Train data	2,283,460	5,544,398	5,061,800
# Test data	497,120	312,211	4,913,320

Table 1: The Statistics of datasets

4.2 Baselines and Metrics

We compare with DeepLog (Du et al., 2017b), Dagmm (Zong et al., 2018), LogAnomaly (Meng et al., 2019b), LogRobust (Zhang et al., 2019), LogTAD (Han and Yuan, 2021), PLELog (Yang et al., 2021), LogBERT (Guo et al., 2021), CAT (Zhang et al., 2022) and ChatGPT (OpenAI, 2022). As anomaly detection is binary classification (Chen et al., 2022), we use precision, recall and F1 score for evaluation (Chen et al., 2023b).

4.3 Log Pre-Processing

For HDFS, log sequences are extracted by block IDs, while BGL and Thunderbird use a 20-sized sliding window. Logs are parsed with Drain (He et al., 2017), and anomalies are identified by windows with anomalous messages. The test set includes all abnormal sequences and an equal number of random normal ones, while the training set contains the rest. Table 1 summarizes key statistics.

4.4 Performance Comparison

Table 2 shows MLAD outperforming all baselines by combining Transformer and GMM strengths. DeepLog struggles with complex datasets, often misclassifying anomalies. LogAnomaly achieves stable F1 scores using semantic vector-based template matching. LogTAD performs well on smaller datasets but underperforms on Thunderbird due to word-level information loss. Similarly, Dagmm shows inconsistent results, particularly on Thunderbird. LogRobust requires extensive manual labeling, limiting unsupervised performance. PLELog performs poorly on unsupervised datasets with long training times. Transformer-based LogBERT and CAT excel at capturing global dependencies and contextual information. However, no baseline consistently performs well across all datasets, facing precision-recall balance challenges and identical shortcut issues.

5 Ablation

5.1 Effect of Components

Our ablation experiments assessed each component's contribution to model performance (Table 2).

	BGL			HDFS			Thunderbird		
	Pre	**Rec**	**F1**	**Pre**	**Rec**	**F1**	**Pre**	**Rec**	**F1**
DeepLog	**0.9659**	0.6396	0.7696	0.5518	0.6785	0.6024	0.7538	0.6027	0.6699
Dagmm	0.9397	0.8831	0.9065	0.9018	0.6214	0.7358	0.5256	0.5395	0.5322
LogAnomaly	0.8918	0.8584	0.7428	0.8213	0.6179	0.7052	0.7672	0.8963	0.8273
LogRobust	0.9531	0.4766	0.6354	0.6989	0.5677	0.6700	0.8675	0.8652	0.8664
LogTAD	0.9102	0.8761	0.8949	0.7793	0.9091	0.8393	0.7523	0.8370	0.7886
PLELog	0.6843	0.8759	0.7314	0.9126	0.8373	0.8799	0.8606	0.8537	0.8671
LogBERT	0.8328	0.8772	0.8579	0.8142	0.7813	0.8089	0.8375	0.8452	0.8402
CAT	0.8727	**0.9481**	0.9106	0.8638	0.8892	0.8771	**0.8994**	0.8838	0.8923
ChatGPT	0.7545	0.6923	0.7221	0.7039	0.7733	0.7369	0.7923	0.7562	0.7738
MLAD	0.9492	0.8932	**0.9184**	**0.9296**	0.8656	**0.8946**	0.8824	**0.9066**	**0.8962**
w/o α-**entmax**	0.9309	0.8904	0.8887	0.7016	**0.9773**	0.8231	0.7892	0.8105	0.8282
w/o **GMM**	0.9128	0.8209	0.8644	0.7443	0.8131	0.7722	0.7534	0.8676	0.8053

Table 2: The performance of different models on the three datasets, and the best model in each column is in bold.

	BGL→Thunderbird		Thunderbird→BGL	
	Pre	**Rec**	**Pre**	**Rec**
DeepLog	0.7225	0.7368	0.7253	0.6817
Dagmm	0.4998	1.0000	0.5005	1.0000
LogAnomaly	0.7517	0.8602	0.7297	0.8029
LogRobust	0.7120	0.8040	0.6473	0.9042
LogTAD	0.8249	0.7322	0.7580	0.7838
PLELog	0.6843	0.7336	0.7367	0.7831
LogBERT	0.7847	0.7916	0.8163	0.8247
CAT	0.7629	0.7292	0.8532	0.8390
MLAD	**0.8277**	**0.8314**	**0.9404**	**0.9635**

Table 3: The transfer performance of the models on two similar datasets (BGL and Thunderbird).

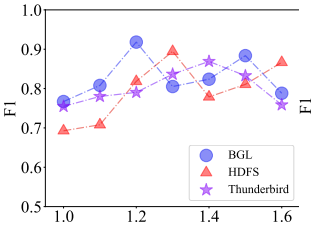


Figure 5: The effect of the $\alpha-$entmax in MLAD.

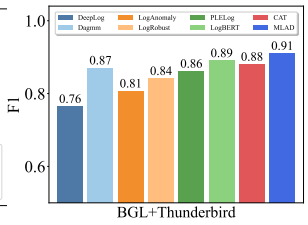


Figure 6: Experiment on multi-system datasets.

Removing the GMM component most significantly degraded performance on BGL and Thunderbird datasets, while having minimal impact on HDFS. This difference correlates with template complexity - BGL (138 templates with 35 in the test only) and Thunderbird (1,291 templates with 243 in the test only) have substantially more templates than HDFS (44 templates with 25 in the test only), demonstrating GMM's importance for learning sparse keyword representations.

We evaluated the effectiveness of $\alpha-$entmax by testing values $\{1.0 \leq \alpha \leq 1.6, \Delta\alpha = 0.1\}$ as shown in Fig. 5. The model performed optimally with α between 1.2-1.5, where $\alpha-$entmax effectively sparsified the dense vector space, enhancing the differentiation between normal and abnormal samples. At α=1 (equivalent to softmax), performance was mediocre, while values above 1.5 introduced excessive sparsity, generating zero-valued keyword weights that caused the model to ignore important features. The sparse transformation remains essential for improving prediction accuracy across tested datasets.

5.2 Effect on Multi-System Datasets

To evaluate cross-system performance, we combined BGL and Thunderbird datasets (both preprocessed using fixed-window mode) into a unified dataset. As shown in Fig. 6, MLAD maintained robust performance while baseline models struggled with the increased generalization requirements of the combined dataset. This highlights the ability of MLAD to detect anomalies that might go unnoticed when systems are analyzed separately. Our ablation experiments revealed that removing Sentence-BERT caused minimal performance degradation on single-system logs but significant losses on multi-source logs. This confirms the importance of sentence-level semantic features for cross-system generalization. The self-attention mechanism effectively captured semantic relationships between words, allowing the model to identify semantically similar anomalous patterns despite different wording. For instance, "error" and "exception" were recognized as semantically related indicators of anomalies, even when followed by different variables.

5.3 Effect of Transferred Knowledge

To validate the model's cross-system performance, we conduct a transfer learning experiment for log anomaly detection using two similar datasets: BGL and Thunderbird. We evaluate the models in terms of Precision and Recall, with results presented in Table 3.

BGL→Thunderbird: Models are trained on BGL and tested on Thunderbird. Dagmm,

DeepLog, and PLELog perform poorly on Thunderbird, with Dagmm failing to detect any anomalies, highlighting its lack of cross-system adaptability. In contrast, LogRobust, LogAnomaly, LogTAD, LogBERT, and CAT exhibit better transfer learning due to effective semantic processing, though their performance is limited by shared words between the two datasets, requiring improved reasoning for unseen terms.

Thunderbird→BGL: Training on Thunderbird and testing on BGL yields better results, primarily due to: (1) Thunderbird's larger dataset, allowing for more comprehensive learning, and (2) the higher proportion of shared words between the two datasets, with BGL containing 261 shared terms, representing a larger portion of its test set compared to Thunderbird.

5.4 Effect of Large Language Model

We evaluated large language models' ability to detect log anomalies using a Chain-of-Thought (Wei et al., 2022) approach rather than direct classification. This two-step process first guides the model to generate templates from log sequences, then identify anomalies based on these templates. Table 4 compares results with and without Chain-of-Thought processing. The findings show that LLMs like ChatGPT struggle with complex log anomaly detection despite the improved reasoning approach. This underperformance stems from their limited domain-specific training and inability to capture the subtle patterns and contextual nuances in system logs. The inherent complexity and variability of operational logs often exceed these models' generalization capabilities.

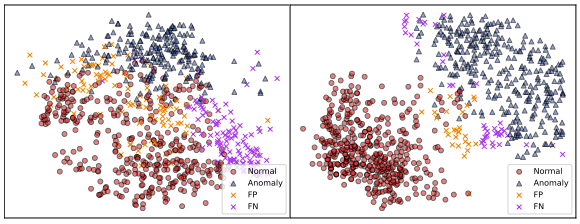
Method	HDFS	BGL	Thunderbird
ChatGPT w/ CoT	0.7369	0.7221	0.7738
ChatGPT w/o CoT	0.6721	0.6542	0.7132

Table 4: F_1 between ChatGPT with/without CoT.

Content

Generation Prompt: Please determine if there are any anomaly in logs, and directly give the answer: Yes or No.

Chain-of-Thought Prompt	
log contents:	2023-08-02 10:30:00 DEBUG: Checking server availability. 2023-08-02 10:30:15 ERROR: NetworkException - Unable to establish connection to server.
Step 1: Log Parsing	
One-Step Prompt:	Extract the templates of *log sequences* while replacing the *variables* with $< * >$
Templates:	1. $< * >$ ERROR: NetworkException - $< * >$ to establish connection to server. 2. $< * >$ DEBUG: Checking server availability.
Step 2: Anomaly Detection	
Two-Step Prompt:	According to the *log sequences*, *Templates:*, the relationship between *Templates:* and *variables*, determine if there are any exceptions in templates and variables, and directly give the answer: Yes or No.
Answer:	Yes or No.



(a) LogAnomaly (b) MLAD

Figure 7: Samples in 2-dimensional space learned by LogAnomaly and MLAD. The red dots • are samples from the normal logs, and the blue triangles △ are samples from the abnormal logs, the orange crosses × (FP) indicate normal samples that the model incorrectly predicts, and conversely, the violet crosses × (FN) indicate abnormal samples that the model incorrectly predicts.

6 Visualization

We evaluated classification performance using t-SNE visualization on 800 balanced BGL samples (normal/abnormal=1:1). As shown in Fig. 7, MLAD achieves clearer class separation than LogAnomaly, which exhibits significant overlap between categories. This improvement is attributed to the α-entmax function's enhanced spatial discrimination capability.

Table 2 reveals two key findings: (1) Removing GMM reduces recall while increasing precision, exposing the Transformer's vulnerability to identical shortcut learning; (2) The 30% lexical gap between training and test sets underscores the persistent challenge of detecting rare keywords in anomaly detection.

7 Conclusion

We propose MLAD, a unified log anomaly detection model combining Transformer and GMM addressing the "identical shortcut" problem. Transformer captures semantic relations, while GMM models complex distributions and handles rare keyword uncertainty through covariance. Experiments on three datasets demonstrate the effectiveness.

Limitations

Hyperparameter Tuning. The hyperparameters used in this study were not fully optimized. Further adjustments and fine-tuning are necessary to better explore the capabilities of model and ensure optimal performance across various experimental settings.

Ethical Considerations

Our method utilizes publicly available log datasets without sensitive user information. However, practical deployment should ensure data privacy and handle potential false alarms carefully to avoid negative impacts on operational reliability.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62276017, 62406033, U1636211, 61672081), and the State Key Laboratory of Complex & Critical Software Environment (Grant No. SKLCCSE-2024ZX-18).

References

Jonathan T. Barron. 2017. Continuously differentiable exponential linear units. *CoRR*, abs/1704.07483.

J. Bridle. 1989. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. *Neurocomputing : Algorithm, architectures, and applications*.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58.

Xuanhao Chen, Liwei Deng, Yan Zhao, and Kai Zheng. 2023a. Adversarial autoencoder for unsupervised time series anomaly detection and interpretation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, page 267–275, New York, NY, USA. Association for Computing Machinery.

Xuanhao Chen, Liwei Deng, Yan Zhao, and Kai Zheng. 2023b. Adversarial autoencoder for unsupervised time series anomaly detection and interpretation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, WSDM '23, page 267–275, New York, NY, USA. Association for Computing Machinery.

Yufu Chen, Meng Yan, Dan Yang, Xiaohong Zhang, and Ziliang Wang. 2022. Deep attentive anomaly detection for microservice systems with multimodal time-series data. In *2022 IEEE International Conference on Web Services (ICWS)*, pages 373–378.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017a. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1285–1298.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017b. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 1285–1298.

Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. Logbert: Log anomaly detection via BERT. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*, pages 1–8. IEEE.

Hongcheng Guo, Yuhui Guo, Jian Yang, Jiaheng Liu, Zhoujun Li, Tieqiao Zheng, Liangfan Zheng, Weichao Hou, and Bo Zhang. 2023a. Loglg: Weakly supervised log anomaly detection via log-event graph construction. In *Database Systems for Advanced Applications - 28th International Conference, DASFAA 2023, Tianjin, China, April 17-20, 2023, Proceedings, Part IV*, volume 13946 of *Lecture Notes in Computer Science*, pages 490–501. Springer.

Hongcheng Guo, Jian Yang, Jiaheng Liu, Jiaqi Bai, Boyang Wang, Zhoujun Li, Tieqiao Zheng, Bo Zhang, Junran Peng, and Qi Tian. 2024. Logformer: A pre-train and tuning pipeline for log anomaly detection. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 135–143. AAAI Press.

Hongcheng Guo, Jian Yang, Jiaheng Liu, Liqun Yang, Linzheng Chai, Jiaqi Bai, Junran Peng, Xiaorong Hu, Chao Chen, Dongfeng Zhang, Xu Shi, Tieqiao Zheng, Liangfan Zheng, Bo Zhang, Ke Xu, and Zhoujun Li. 2023b. OWL: A large language model for IT operations. *CoRR*, abs/2309.09298.

Xiao Han and Shuhan Yuan. 2021. Unsupervised cross-system log anomaly detection via domain adaptation. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, page 3068–3072.

Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services, ICWS*, pages 33–40.

Shaohan Huang, Yi Liu, Carol J. Fung, Rong He, Yining Zhao, Hailong Yang, and Zhongzhi Luan. 2020. Hitanomaly: Hierarchical transformers for anomaly detection in system log. *IEEE Trans. Netw. Serv. Manag.*, 17(4):2064–2076.

Huber and PeterJ. 2009. *Robust statistics / 2nd ed.*

Max Landauer, Florian Skopik, and Markus Wurzenberger. 2024. A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques. *Proceedings of the ACM on Software Engineering*, 1(FSE):1354–1375.

Mingrui Ma, Lansheng Han, and Chunjie Zhou. 2024. Research and application of transformer based anomaly detection model: A literature review. *arXiv preprint arXiv:2402.08975*.

Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. 2011. Rapid detection of maintenance induced changes in service performance. In *Proceedings of the 2011 Conference on Emerging Networking Experiments and Technologies, Co-NEXT '11, Tokyo, Japan, December 6-9, 2011*, page 13. ACM.

Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. 2019a. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4739–4745.

Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. 2019b. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, pages 4739–4745.

Adam J. Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN*, pages 575–584.

OpenAI. 2022. Chatgpt.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, pages 1504–1519.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP*, pages 3980–3990.

Jing Su, Chufeng Jiang, Xin Jin, Yuxin Qiao, Tingsong Xiao, Hongda Ma, Rong Wei, Zhi Jing, Jiajun Xu, and Junhong Lin. 2024. Large language models for forecasting and anomaly detection: A systematic literature review. *arXiv preprint arXiv:2402.10350*.

C. Tsallis. 1988. Possible generalization of boltzmann-gibbs statistics. *Journal of Statistical Physics*, pages 479–487.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, pages 5998–6008.

Luke Vilnis and Andrew McCallum. 2015. Word representations via gaussian embedding. In *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 845–854.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, and Wenbin Zhang. 2021. Plelog: Semi-supervised log-based anomaly detection via probabilistic label estimation. In *43rd IEEE/ACM International Conference on Software Engineering: Companion Proceedings, ICSE*, pages 230–231.

Xincheng Yao, Ruoqi Li, Zefeng Qian, Lu Wang, and Chongyang Zhang. 2024. Hierarchical gaussian mixture normalizing flow modeling for unified anomaly detection. In *European Conference on Computer Vision*, pages 92–108. Springer.

Zhiyuan You, Lei Cui, Yujun Shen, Kai Yang, Xin Lu, Yu Zheng, and Xinyi Le. 2022. A unified model for multi-class anomaly detection. *CoRR*, abs/2206.03687.

Boxi Yu, Jiayi Yao, Qiuai Fu, Zhiqing Zhong, Haotian Xie, Yaoliang Wu, Yuchi Ma, and Pinjia He. 2024. Deep learning or classical machine learning? an empirical study on log-based anomaly detection. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, pages 1–13.

Shengming Zhang, Yanchi Liu, Xuchao Zhang, Wei Cheng, Haifeng Chen, and Hui Xiong. 2022. CAT: beyond efficient transformer for content-aware anomaly detection in event sequences. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 4541–4550. ACM.

Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, Junjie Chen, Xiaoting He, Randolph Yao, Jian-Guang Lou, Murali Chintalapati,

Furao Shen, and Dongmei Zhang. 2019. Robust log-based anomaly detection on unstable log data. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT FSE*, pages 807–817.

Yin Zhang, Zihui Ge, Albert G. Greenberg, and Matthew Roughan. 2005. Network anomography. In *Internet Measurement Conference*, pages 317–330. USENIX Association.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Dae-ki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *6th International Conference on Learning Representations, ICLR, Conference Track Proceedings*.