2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC) | 979-8-3315-7434-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/COMPSAC65507.2025.00173

Beyond Log Parsers: A Scalable AI-Driven Framework for Efficient Log Anomaly Detection in Software Engineering

Yicheng Sun, Jacky Keung, Hi Kuen Yu, Shuo Liu, Yihan Liao, and Jingyu Zhang
Department of Computer Science, City University of Hong Kong, Hong Kong, China
{yicsun2-c, hikuenyu2-c, sliu273-c, yihanliao4-c, jzhang2297-c}@my.cityu.edu.hk, jacky.keung@cityu.edu.hk

***Abstract*—Log anomaly detection is critical for ensuring software system reliability and security, yet challenges persist in log parser dependency, small-scale dataset applicability, and hyperparameter tuning efficiency. Existing methods over-rely on predefined log templates, leading to information loss and high computational overhead. Additionally, anomaly detection models often struggle with limited log data, and hyperparameter tuning remains computationally expensive in dynamic environments. In this paper, we empirically evaluate seven state-of-the-art anomaly detection models across varied software systems, assessing the necessity of log parsers and model performance on small-scale datasets. Furthermore, we propose SMAC-<T>, an enhanced real-time hyperparameter optimization framework, integrating stochastic gradient descent (SGD) and adaptive learning to improve model adaptability and efficiency. Our experiments on six benchmark datasets demonstrate that SMAC-<T> achieves an overall average F1-score improvement of 4.27%, a 27.55% reduction in hyperparameter tuning time compared to other models, and a 1.35% increase in F1-score when adapting to newly emerging logs, compared to its counterpart without SGD integration. These findings underscore the practical advantages of AI-driven log analysis, providing valuable insights into scalable, software-engineered anomaly detection.**

***Index Terms*—AI-Driven Log Anomaly Detection, Real-Time Hyperparameter Optimization, Log Analysis, Scalable Log Parsing and Detection, Empirical Software Anomaly Analysis**

I. Introduction

Log anomaly detection plays a vital role in modern software engineering by enabling proactive failure prevention [1]. As software systems scale, the volume and heterogeneity of logs increase, rendering manual inspection impractical and making automated detection essential [2]. However, traditional approaches heavily depend on log parsers, which introduce significant computational overhead, enforce rigid template dependencies, and risk discarding anomaly-related information [3], [4]. Studies show that most log parsers cannot process large-scale datasets efficiently [5], and even state-of-the-art models like ChatGPT struggle with parsing time [6]. Moreover, while deep learning models achieve high F1-scores [7], the necessity of intensive parsing remains questionable—especially when many logs contain limited dynamic variables and anomaly detection could proceed without extensive preprocessing.

Beyond parser limitations, current anomaly detection models face challenges in low-resource scenarios and hyperparameter tuning. Small-scale systems—such as embedded devices, lab prototypes, and early-stage software—generate limited logs, yet demand high reliability. These contexts lack sufficient training data, making generalization difficult. Meanwhile, traditional tuning methods like Grid Search are computationally expensive and inefficient, particularly in evolving environments with dynamic log patterns. They often fail to identify optimal configurations due to restricted search scopes, hindering adaptability and responsiveness. These limitations highlight the need for lightweight, parser-free approaches and more efficient tuning strategies that maintain performance without incurring excessive computational costs.

To explore and address these challenges, we conducted a detailed empirical study. Firstly, we compared the performance of various anomaly detection models across different parsers and in the absence of parsers, in order to assess the effectiveness of using log parsers in the detection process. Subsequently, we applied advanced deep learning models to anomaly detection tasks across log datasets generated by systems of varying sizes, evaluating whether the models exhibit consistent performance across large-scale and small-scale log datasets.

Finally, we enhance an advanced method for hyperparameter optimization, specifically SMAC, to fine-tune the hyperparameters of model backbones. This approach enables the efficient identification of optimal parameters within deep learning models while significantly reducing the time required for hyperparameter tuning. We name this framework SMAC-<T>, where <T> represents the flexibility of the framework, allowing it to work seamlessly with different backbones, akin to the concept of genericity in programming. The advantages of this framework include enhanced efficiency in determining discrete parameters, improved model performance with minimal computational cost, and the ability to be easily adapted to various deep learning models for diverse log anomaly detection tasks. Moreover, by incorporating Real-Time Online Learning and adaptive learning rate techniques, SMAC-<T> ensures continuous adaptation to evolving system conditions, further optimizing the anomaly detection process in dynamic environments. In summary, our primary contributions can be summarized in four-fold:

- We systematically compare anomaly detection models across different log parsers and in the absence of parsers, while also exploring the necessity of using log parsers in the detection process.
- We are the first to evaluate advanced models for log

2836-3795/25/$31.00 ©2025 IEEE
DOI 10.1109/COMPSAC65507.2025.00173

anomaly detection in small-scale systems, assessing the generalizability of these models across various system sizes and log data volumes.
- We propose the enhanced SMAC-<T> framework for hyperparameter tuning, which provides a more efficient and adaptive solution compared to traditional methods and is specifically designed to improve the performance and adaptability of anomaly detection models across diverse software systems and log data.
- We conduct a series of detailed empirical studies that fill research gaps in log anomaly detection, specifically addressing the effectiveness of log parsers, the generalizability of deep learning models in small-scale systems, and the optimization of hyperparameters for improved efficiency.

II. Background on Log-Based Anomaly Detection

A. Log Parsing

Logs are typically produced by instrumented log statements that record system events, often containing structured fields such as timestamps, severity levels, and message content [8]. The primary goal of log parsing is to convert raw log messages into structured formats by identifying event templates (static parts) and dynamic variables (parameters) [3]. One widely used parser, Drain [9], utilizes a fixed-depth parsing tree to encode specialized parsing rules, making it a popular choice in log parsing. Despite the widespread use of log parsing to structure log data, these methods often rely heavily on predefined templates, which can lead to the loss of crucial information and an overreliance on parser compatibility [4]. Moreover, log parsers can be slow, particularly when processing large datasets, and the inclusion of large language models (LLMs) in parsing can exacerbate these time-related challenges. While recent deep learning models show promising F1-scores, the main focus of log analysis remains rapid anomaly detection [10]. This raises the question: Is extensive log parsing still necessary, given the advancements in detection techniques?

B. Representative Anomaly Detection Models

Anomaly detection in software logs is commonly regarded as a binary classification problem. Researchers have proposed various methods for detecting abnormal logs within large-scale log datasets. These algorithms can be broadly categorized into unsupervised [11], [12] and supervised [13]–[16] learning approaches. In this paper, we systematically evaluate seven representative models—**DeepLog** [11], **HitAnomaly** [17], **NeuralLog** [18], **LogRobust** [16], **PLELog** [13], **LogBERT** [19], and **SemiRALD** [20]—by comparing their anomaly detection performance under two settings: with and without log parsing. Additionally, we assess their effectiveness on small-scale datasets, an aspect that has received limited attention in prior work. The significance of evaluating these models on small-scale datasets lies in the fact that many real-world systems, particularly in early-stage development or resource-constrained environments, do not generate large volumes of logs. By addressing this gap, our study contributes valuable insights into the applicability and adaptability of existing anomaly detection models in real-world, low-resource scenarios.

III. Real-Time Hyperparameter Optimization for Anomaly Detection

The SMAC [21] method provides an efficient approach for hyperparameter optimization; however, it assumes a static data distribution during training. In anomaly detection, data patterns and system behaviors can evolve over time due to new logs, software updates, or changing user behaviors. This requires the model to adapt continuously, which the fixed training approach of SMAC cannot support. To address this limitation, we extend the capabilities of SMAC-<T> by incorporating Real-Time Online Learning to account for the dynamic nature of system logs and evolving patterns in anomaly detection. To enable this, we introduce Stochastic Gradient Descent (SGD) and adaptive learning rate techniques to update the model's parameters in real-time as new data arrives. These techniques ensure the model maintains high accuracy and detects emerging anomalies, facilitating continuous adaptation to evolving system conditions.

Step 1: Construct a training set using the initial variables as input for the deep learning model, then feed this training set into the model.

Step 2: SMAC-<T> constructs the surrogate model using Random Forest and Bayesian Optimization techniques, enabling efficient hyperparameter optimization. Initially, SMAC performs extensive random sampling to find the optimal hyperparameter configurations. After this, we integrate Stochastic Gradient Descent (SGD) for real-time updates, allowing the model to continuously adjust its weights as new data samples arrive. The weight update rule is as follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla \mathcal{L}(\theta_t, x_t)$$

where η_t is the learning rate at time step t, and $\nabla \mathcal{L}(\theta_t, x_t)$ is the gradient of the loss function with respect to the current sample x_t. This ensures that the model adapts to the evolving data distribution without requiring a full retraining process.

In addition to SGD, we employ adaptive learning rate techniques like Adam, which adjusts the learning rate dynamically for each parameter. The update rule for Adam is:

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

where m_t and v_t are the first and second moment estimates, η_t is the learning rate, and ϵ is a small constant to avoid division by zero. Adam's adaptive learning rate improves the efficiency and stability of the training process, especially when the system behavior changes over time.

Step 3: Once the initial hyperparameters are optimized using SMAC, the model enters a real-time learning phase, where its weights are continuously updated as new log data is received. Leveraging Stochastic Gradient Descent (SGD) with adaptive learning rates, the model incrementally adjusts

its parameters to adapt to evolving system patterns. Over time, these updates lead the model to converge toward an optimal set of weights, enabling accurate anomaly detection in dynamic environments. As new log sequences arrive, the model processes them in real time; if an anomaly is detected, an alert is immediately generated and sent to engineers for timely diagnosis and resolution.

IV. Study Design

A. Datasets

We select four public datasets to evaluate the performance of anomaly detection models: HDFS [22], BGL [23], Thunderbird [23], and Spirit [23]. These datasets are labeled with log types and exhibit significant differences in log formats, enabling us to assess the models' generalization performance. Additionally, we conduct experiments on two small-scale log datasets, which were constructed from our own software. These datasets are: the Educational App Interactive Learning Dataset (EAILD) and the Intelligent Laboratory Management System Dataset (ILMSD). The specific distribution of logs is outlined in Table I.

TABLE I: The number of log messages in each dataset.

Dataset	Category	Numbers	Anomalies
HDFS	Distributed file system	11,175,629 (575,061 blocks)	16,838 blocks (2.93%)
BGL	Supercomputer	4,747,963	348,469 (7.34%)
Thunderbird	Supercomputer	10,000,000	4,934 (0.49%)
Spirit	Supercomputer	5,000,000	764,500 (15.29%)
EAILD	Limited-scale	73,626	4,645 (6.31%)
ILMSD	Limited-scale	54,637	3,129 (5.73%)

B. Research Questions

This paper mainly focuses on the following three research questions and designs our experiments accordingly.

RQ1: Can the model maintain consistent and exceptional anomaly detection performance across different software systems and configurations?

To evaluate whether the model can maintain consistent and exceptional anomaly detection performance across different software systems and configurations, we conduct experiments using seven models on both large-scale and small-scale datasets. The large-scale datasets include HDFS, BGL, Thunderbird, and Spirit, while the small-scale datasets are EAILD and ILMSD. The models evaluated include unsupervised learning-based models (DeepLog), supervised learning-based models (HitAnomaly, NeuralLog, LogRobust), and semi-supervised learning-based models (PLELog, LogBERT, SemiRALD). This diverse set of models is tested on datasets that represent different system behaviors and log structures, with the aim of assessing their ability to consistently and effectively detect anomalies across a variety of software environments and configurations. Detailed descriptions of these benchmark models are provided in Section II-B. Notably, the experimental results for DeepLog are sourced from the LogBERT paper, while the remaining results are taken from their respective original research.

RQ2: How does the omission or presence of a log parser impact the performance of the models?

In this experiment, we evaluate five representative log anomaly detection models (DeepLog, PLELog, LogBERT, LogRobust, and SemiRALD) to compare their performance with and without the use of log parsers. Additionally, we measure the time saved by bypassing log parsers. The BGL and Spirit datasets are selected for this experiment due to their diverse log structures and representation of both large-scale and complex systems. These datasets offer a broad range of anomalies, enabling us to test the models' adaptability to different data types.

We assess five of the most common log parsers on both datasets: four rule-based parsers (Drain [9], Spell [24], AEL [25], and IPLoM [8]) and a state-of-the-art LLM-based parser, LILAC [26], which leverages the GPT-3.5-turbo-06132 model. This analysis allows us to evaluate how effectively anomaly detection models perform without predefined log templates, offering insights into whether log parsers are still necessary, given the capabilities of advanced deep learning models. We also measure the computational efficiency of models with and without parsers, comparing the time required for both approaches.

RQ3: How does our optimized SMAC-<T> framework perform in anomaly detection tasks?

In this experiment, we combine the SMAC method with traditional deep learning models, specifically LSTM, GRU, and RNN, as well as the best-performing model from RQ1, SemiRALD, to conduct experiments with eight models: SMAC-LSTM, SMAC-GRU, SMAC-CNN, SMAC-SemiRALD, LSTM, GRU, CNN, and SemiRALD. These models are evaluated on the six datasets from RQ1. The goal is to assess the improvement in anomaly detection performance provided by the SMAC method and its generalization effectiveness across different deep learning models.

We adopt a semi-supervised learning approach, providing only 10% of the logs as labeled data for training. Additionally, when constructing the training set, we exclude 20% of the logs that are in an unfamiliar format (logs that do not appear in the training set but only in the test set). This allows us to evaluate the SMAC framework's ability to generalize and adapt to previously unseen log formats, providing valuable insights into its performance under dynamic and evolving system conditions.

C. Evaluation Metrics

Log anomaly detection is treated as a classification problem. To assess the effectiveness of models in detecting anomalies, we rely on $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F1\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$ metrics to evaluate model performance. Specifically, TP (True Positive) is the count of anomalous log sequences correctly detected by the model. FP (False Positive) is the count of normal log sequences incorrectly identified as anomalies. TN (True Negative) is the count of normal logs that are correctly classified. FN (False

Negative) is the count of anomalous log sequences that the model failed to detect.

V. Results and Analysis

A. Effectiveness of the anomaly detection models (RQ1)

In this section, we conduct experiments using seven models on both large-scale and small-scale datasets. The large-scale datasets include HDFS, BGL, Thunderbird, and Spirit, while the small-scale datasets are EAILD and ILMSD, as shown in Table II. The models evaluated encompass unsupervised, supervised, and semi-supervised learning approaches.

Overall, the models perform significantly better on large-scale datasets compared to small-scale ones. More advanced models, such as SemiRALD and LogBERT, exhibit superior performance due to their ability to capture complex patterns in data through sophisticated architectures. These models incorporate techniques such as BERT (in the case of LogBERT) and hybrid language models (such as SemiRALD), enabling them to better understand the underlying structure of logs, even in semi-supervised learning scenarios and in the presence of noise or incomplete information. For example, SemiRALD utilizes the strengths of a hybrid language model, combining different model types to enhance its anomaly detection capabilities. This approach ensures stable and consistent performance across both large- and small-scale datasets, demonstrating that more advanced and complex models can maintain high accuracy while adapting to diverse log formats, regardless of dataset size. This robustness is a key differentiator compared to traditional models, which often struggle to maintain such consistency with smaller, more complex datasets.

Answer to RQ1: Experimental results highlight the critical importance of enhancing performance on small-scale datasets, where models must contend with both limited training data and high log diversity. Notably, advanced models such as SemiRALD demonstrate strong robustness, achieving high accuracy even under these constrained conditions.

B. Impact of Log Parsers on Model Performance (RQ2)

In this section, we evaluate five representative log anomaly detection models (DeepLog, PLELog, LogBERT, LogRobust, and SemiRALD) to compare their performance with and without the use of log parsers. We assess five of the most common log parsers: four rule-based parsers (Drain, Spel, AEL, and IPLoM), along with a state-of-the-art large language model (LLM)-based parser, LILAC, which leverages the GPT-3.5-turbo-06132 model. Table III presents the performance of various log parsers across different models when applied to the BGL and Spirit datasets.

It is important to note that models incorporating parsers do not consistently outperform their counterparts without a log parser. For instance, on the BGL dataset, CNN paired with the Drain parser, PLELog with both the Drain and Spell parsers, LogBERT with the Spell parser, and SemiRALD with the Spell parser all showed lower F1 scores than their versions without parsers. In contrast, LILAC, which utilizes ChatGPT as a parser, achieved the best performance across four models (excluding the supervised model DeepLog), demonstrating F1-score improvements of 28.3% for PLELog and 4.18% for LogBERT over other parsers. A similar trend was observed with the Spirit dataset, where LILAC provided the best results across three models. However, when parsers were excluded, all six models outperformed those relying on certain rule-based parsers, suggesting that these parsers do not always benefit anomaly detection tasks.

When no log parser is used, performance remains highly competitive, especially with advanced models. For example, on the BGL dataset, CNN achieves an F1-score of 0.914 without a parser, nearly matching the 0.972 achieved with the ChatGPT parser. Similarly, LogBERT scores an F1 of 0.893 without a parser, compared to 0.909 with ChatGPT. Furthermore, omitting the log parsing step reduces the overall time required for model training and evaluation. Our experiments show average processing time reductions of 23.7% and 30.9% on the two datasets, making this approach particularly practical for quick deployment or real-time anomaly detection.

Interestingly, the SemiRALD model demonstrates almost identical performance with or without a log parser, achieving F1-scores of 0.990 and 0.988 on the two datasets, respectively. This consistency highlights that, for advanced models like SemiRALD, the use of log parsers has minimal impact on accuracy. In fact, bypassing the parser results in substantial time savings, reducing runtime by 30.4% and 37.1% on the two datasets. These findings suggest that for sophisticated models such as SemiRALD, log parsing is not crucial.

Answer to RQ2: SemiRALD demonstrated comparable performance with and without the use of a log parser, suggesting that a parser is not essential for effective anomaly detection. In comparison to other benchmark models, advanced models can achieve high accuracy even without relying on a log parser, as evidenced by the small performance differences observed in CNN and LogBERT. Furthermore, bypassing the log parsing step results in significant time savings. Therefore, the necessity of using a log parser, along with its associated time costs, should be carefully reconsidered when designing future anomaly detection models.

C. Evaluation of Efficiency and Generation Time (RQ3)

In this section, we integrate the SMAC method with traditional deep learning models, specifically LSTM, GRU, and RNN, along with the best-performing model from RQ1, SemiRALD, to conduct experiments with eight models: SMAC-LSTM, SMAC-GRU, SMAC-CNN, SMAC-SemiRALD, LSTM, GRU, CNN, and SemiRALD. For SMAC-LSTM, SMAC-GRU, and SMAC-CNN, we tune the following hyperparameters: *batch size*, *learning rate*, *epochs*, *embedding dimension*, *hidden size*, *number of layers*, and *dropout*. Since SemiRALD is a more complex model, we determine hyperparameters for RoBERTa's *learning rate* and *dropout rate*, BiLSTM's *learning rate*, *hidden size*, and *dropout rate*, as well as the number of *attention heads* for the attention mechanism, and the *batch size* and *epochs* for the entire model. We adopt a semi-supervised learning approach, where only 10% of the logs are labeled for training.

TABLE II: The performance of different models on four datasets.

Model	HDFS			BGL			Thunderbird			Spirit			EAILD			ILMSD		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Unsupervised learning-based																		
DeepLog	0.884	0.695	0.773	0.897	0.828	0.861	0.484	0.895	0.628	0.438	**1**	0.609	0.327	**0.963**	0.488	0.311	0.347	0.328
Supervised learning-based																		
HitAnomaly	**1**	0.970	0.980	0.950	0.900	0.920	0.950	0.810	0.880	0.920	0.750	0.840	0.793	0.670	0.726	0.748	0.661	0.702
NeuralLog	0.960	**1**	0.980	0.980	0.980	0.980	0.920	**0.990**	0.950	0.919	0.908	0.914	0.857	0.831	0.844	0.822	0.807	0.815
LogRobust	0.980	0.960	0.970	0.910	0.770	0.830	0.610	0.780	0.680	0.985	0.915	0.949	0.829	0.740	0.782	0.847	0.759	0.801
Semi-supervised learning-based																		
PLELog	0.950	0.963	0.957	0.965	**0.999**	0.982	0.864	0.941	0.901	0.931	0.767	0.841	0.824	0.801	0.813	0.797	0.773	0.785
LogBERT	0.870	0.781	0.823	0.894	0.923	0.908	0.962	0.965	0.963	0.862	0.807	0.834	0.773	0.726	0.749	0.764	0.822	0.792
SemiRALD	0.994	0.991	**0.993**	0.991	0.989	**0.990**	**0.986**	0.981	**0.984**	**0.990**	0.987	**0.988**	**0.931**	0.949	**0.939**	**0.952**	**0.965**	**0.958**

TABLE III: The performance of different log parsers on the BGL and Spirit datasets.

Model		BGL						Spirit					
		Drain	Spell	AEL	IPLoM	LILAC	-	Drain	Spell	AEL	IPLoM	LILAC	-
CNN	P	0.871	0.994	0.942	0.937	0.986	0.932	0.986	0.959	0.986	0.986	0.967	0.958
	R	0.947	0.965	0.947	0.959	0.958	0.966	1	1	1	1	0.991	0.982
	F1	0.908	**0.979**	0.945	0.948	0.972	0.949	**0.993**	0.979	**0.993**	**0.993**	0.979	0.970
DeepLog	P	0.270	0.271	0.271	0.273	0.398	0.310	0.438	0.602	0.413	0.607	0.621	0.528
	R	0.988	0.988	0.988	0.988	0.798	0.568	1	1	1	1	0.994	0.891
	F1	0.426	0.426	0.425	0.427	**0.531**	0.401	0.609	0.751	0.584	0.755	**0.764**	0.663
PLELog	P	0.702	0.560	0.661	0.655	0.958	0.879	0.931	0.859	0.863	0.500	0.884	0.737
	R	0.791	0.404	0.854	0.433	0.856	0.797	0.767	0.859	0.887	0.662	0.897	0.792
	F1	0.744	0.476	0.744	0.521	**0.904**	0.836	0.841	0.859	0.875	0.570	**0.890**	0.764
LogBERT	P	0.897	0.887	0.824	0.831	0.901	0.828	0.862	0.793	0.752	0.774	0.883	0.802
	R	0.907	0.881	0.853	0.859	0.917	0.861	0.807	0.767	0.793	0.803	0.871	0.759
	F1	0.902	0.884	0.838	0.845	**0.909**	0.844	0.834	0.780	0.772	0.788	**0.877**	0.780
LogRobust	P	0.910	0.849	0.844	0.726	0.902	0.693	0.985	0.947	0.986	0.973	0.980	0.963
	R	0.770	0.988	0.982	0.977	0.926	0.802	0.915	1	1	1	0.942	0.940
	F1	0.830	0.914	0.908	0.833	**0.915**	0.743	0.949	0.973	**0.993**	0.986	0.965	0.951
SemiRALD	P	0.990	0.988	0.990	0.991	0.990	0.991	0.991	0.985	0.989	0.988	0.987	0.990
	R	0.990	0.989	0.987	0.989	0.991	0.989	0.984	0.988	0.988	0.986	0.987	0.987
	F1	**0.990**	0.989	0.989	**0.990**	**0.990**	**0.990**	0.987	0.986	**0.988**	0.987	0.987	**0.988**

Note: The highest performance for each model is highlighted in **bold**, and "–" indicates no log parser is used.

Additionally, when constructing the training set, 20% of the logs with unfamiliar formats—logs present only in the test set and not in the training set—are excluded. To enable real-time adaptation to new data patterns, we apply SGD and Adam for updates. These models are evaluated on the six datasets from RQ1, and Figure 1 illustrates their specific performance.

The experimental results demonstrate that models integrated with SMAC—SMAC-LSTM, SMAC-GRU, SMAC-CNN, and SMAC-SemiRALD—achieved significant improvements in average F1-scores across six datasets compared to their counterparts without SMAC. These results underscore the effectiveness of SMAC in optimizing hyperparameters and enhancing model performance. The improvements are particularly noteworthy given the semi-supervised learning setting, where only 10% of the logs were labeled, highlighting SMAC's ability to fine-tune models even with limited labeled data.

In addition to performance gains, the time efficiency of hyperparameter tuning was significantly improved by SMAC. Models without SMAC relied on grid search for hyperparameter optimization, resulting in average tuning time increases of 25.3%, 37.1%, 44.2%, 39.6%, 10.8%, and 8.3% across the six datasets compared to their SMAC counterparts. Furthermore, by dynamically updating parameters, SGD effectively prevents overfitting and adapts to diverse log formats. The inclusion of SGD resulted in an average F1-score improvement of 1.35% across all datasets. This optimization further reinforces SMAC's role in enhancing both efficiency and accuracy for anomaly detection tasks.

Answer to RQ3: Our enhanced SMAC-<T> framework significantly improves anomaly detection performance across diverse datasets while reducing hyperparameter tuning time. This highlights its effectiveness in optimizing both performance and efficiency, even in semi-supervised settings with limited labeled data.

VI. Conclusion

This paper presents a comprehensive empirical study on log anomaly detection, addressing key challenges in log parser dependency, small-scale data performance, and hyperparameter tuning efficiency. Our findings demonstrate that log parsers, while historically useful, may not be essential for advanced anomaly detection models, particularly when using hybrid AI-driven approaches. Furthermore, our proposed SMAC-<T> framework significantly enhances hyperparameter tuning

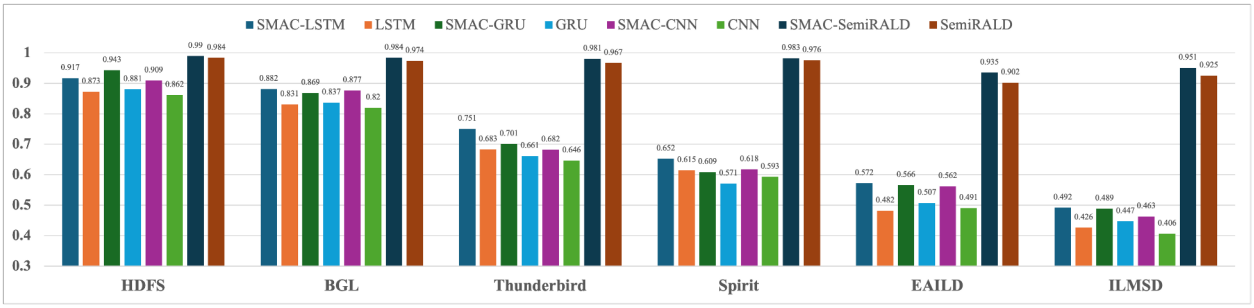

Fig. 1: The F1-score performance of models across different datasets.

efficiency, achieving an overall average F1-score improvement of 4.27%, a 27.55% reduction in tuning time compared to other models, and a 1.35% increase in F1-score when adapting to newly emerging logs, compared to its counterpart without SGD integration. These results highlight the practical advantages of AI-driven log analysis, offering a scalable, software-engineered solution for real-time anomaly detection.

ACKNOWLEDGMENT

This work is partially supported by the General Research Fund of the Research Grants Council of Hong Kong and the research funds from the City University of Hong Kong (6000796, 9229109, 9229098, 9220103, 9229029).

REFERENCES

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

[2] J. Nyyssölä and M. Mäntylä, "Efficiency of unsupervised anomaly detection methods on software logs," *arXiv preprint arXiv:2312.01934*, 2023.

[3] V.-H. Le and H. Zhang, "Log parsing with prompt-based few-shot learning," *arXiv preprint arXiv:2302.07435*, 2023.

[4] X. Xie, S. Jian, C. Huang, F. Yu, and Y. Deng, "Logrep: Log-based anomaly detection by representing both semantic and numeric information in raw messages," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 194–206.

[5] Z. Jiang, J. Liu, J. Huang, Y. Li, Y. Huo, J. Gu, Z. Chen, J. Zhu, and M. R. Lyu, "A large-scale evaluation for log parsing techniques: How far are we?" in *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024, pp. 223–234.

[6] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Divlog: Log parsing with prompt enhanced in-context learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–12.

[7] B. Yu, J. Yao, Q. Fu, Z. Zhong, H. Xie, Y. Wu, Y. Ma, and P. He, "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*, 2024, pp. 1–13.

[8] A. A. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1255–1264.

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

[10] Z. A. Khan, D. Shin, D. Bianculli, and L. C. Briand, "Impact of log parsing on deep learning-based anomaly detection," *Empirical Software Engineering*, vol. 29, no. 6, p. 139, 2024.

[11] M. Du, F. Li, G. Zheng, and V. Srikumar, "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.

[12] V. Zeufack, D. Kim, D. Seo, and A. Lee, "An unsupervised anomaly detection framework for detecting anomalies in real time through network system's log files analysis," *High-Confidence Computing*, vol. 1, no. 2, p. 100030, 2021.

[13] Y. Lin and Y.-Y. Chiang, "A semi-supervised learning approach for abnormal event prediction on large network operation time-series data," in *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 1024–1033.

[14] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.

[15] S. Ying, B. Wang, L. Wang, Q. Li, Y. Zhao, J. Shang, H. Huang, G. Cheng, Z. Yang, and J. Geng, "An improved knn-based efficient log anomaly detection method with automatically labeled samples," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 3, pp. 1–22, 2021.

[16] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li *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.

[17] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, and Z. Luan, "Hitanomaly: Hierarchical transformers for anomaly detection in system log," *IEEE transactions on network and service management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[18] V.-H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 492–504.

[19] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 international joint conference on neural networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[20] Y. Sun, J. W. Keung, Z. Yang, S. Liu, and H. K. Yu, "Semirald: A semi-supervised hybrid language model for robust anomalous log detection," *Information and Software Technology*, p. 107743, 2025.

[21] Y. Sun, J. Keung, J. Zhang, H. K. Yu, W. Luo, and S. Liu, "Unveiling hidden anomalies: Leveraging smac-lstm for enhanced software log analysis," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024, pp. 1178–1183.

[22] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, 2009, pp. 117–132.

[23] A. Oliner and J. Stearley, "What supercomputers say: A study of five system logs," in *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN'07)*. IEEE, 2007, pp. 575–584.

[24] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 859–864.

[25] Z. M. Jiang, A. E. Hassan, P. Flora, and G. Hamann, "Abstracting execution logs to execution events for enterprise applications (short paper)," in *2008 The Eighth International Conference on Quality Software*. IEEE, 2008, pp. 181–186.

[26] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu, "Lilac: Log parsing using llms with adaptive parsing cache," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 137–160, 2024.