

# A Comprehensive Empirical Evaluation of Lightweight LLMs for Software Log Parsing

**Abstract**—Log parsing, a foundational step in software log anomaly detection, plays a critical role in extracting structured templates from unstructured log data. Although recent advances have demonstrated the effectiveness of Large Language Models (LLMs) in log parsing tasks, current research remains constrained by the use of outdated benchmark datasets and a lack of systematic investigation into the capabilities of smaller, open-source LLMs. This study presents a comprehensive empirical evaluation of ten open-source lightweight LLM-based parsers, with parameter sizes ranging from 0.5B to 9B, across six newly constructed datasets that represent both modern large-scale and low-resource log environments. Our results reveal significant variability in parsing accuracy (PA) and F1-score of template accuracy (FTA), with models achieving averages of up to 70.1% PA and 66.8% FTA on benchmark datasets such as LHSB, but dropping to 58.7% PA and 53.1% FTA on more diverse datasets like FALL. Longer logs and those containing more dynamic variables also led to noticeable declines in performance. Notably, both few-shot learning methods improved performance by an average of 7.46% PA. The example-based few-shot learning approach slightly outperformed the rules-based counterpart, with the most significant improvement observed on the FALL dataset, where it achieved a 14.2% increase in Parsing Accuracy (PA). This empirical study emphasizes the importance of dataset diversity and targeted few-shot learning to enhance the generalization capabilities of LLMs in real-world log parsing tasks, offering actionable insights for researchers and practitioners in software engineering.

**Index Terms**—Log Parsing, Large Language Models (LLMs), Few-shot Learning, Template Extraction, Dataset Diversity

## I. INTRODUCTION

Software logs are a vital source of information in ensuring the stability and reliability of modern systems, enabling developers to monitor system behavior, detect anomalies, and diagnose failures. [1]. As such, they serve as a critical resource for gaining insights into system behavior, identifying errors, and uncovering performance bottlenecks [2]. Efficient anomaly detection is important for identifying potential failures, security breaches, or operational inefficiencies before they escalate into critical issues [3]–[5]. As software systems grow increasingly complex, effective log analysis and anomaly detection techniques are paramount for ensuring system stability.

Log parsing is a crucial step in anomaly detection, as the accuracy of the parsing process directly impacts the effectiveness of subsequent anomaly detection tasks [6], [7]. If logs are parsed incorrectly, valuable information may be missed, leading to poor performance of anomaly detection models [8]. Traditional rule-based parsers [9]–[12] have been widely used for log analysis, but recent research has shown that Large Language Model-based (LLM-based) parsers [13]–[16] generally outperform rule-based approaches in terms of

accuracy and flexibility. LLMs can leverage vast amounts of data and learn complex patterns, which enhances their ability to handle a wide range of log formats and structures [16]. Despite their promise, the application of LLMs to log parsing faces several challenges, including reliance on outdated datasets, high computational costs, and limited exploration of smaller, open-source models.

**Limitations of Existing Assessment Datasets:** Evaluations of LLM-based parsers have largely relied on datasets provided by the LogHub platform [17], which are known to have several constraints [18]. One significant limitation is that these datasets primarily consist of logs from systems over a decade old, featuring relatively uniform log formats. Consequently, the performance observed on these datasets may not generalize well to modern software systems, which typically involve more advanced technologies and diverse log structures. Additionally, LLMs tested on these datasets may exhibit inflated performance, as models might have previously encountered similar log formats during pretraining, leading to "memorization" of specific patterns [13]. Moreover, existing benchmark datasets predominantly focus on logs from large-scale, interconnected systems, such as enterprise-level applications. Although valuable for specific contexts, recent studies highlight the necessity of log analysis in smaller, low-resource environments [19]. Effective anomaly detection is equally critical in laboratory setups, early-stage software deployments, small industrial applications, and embedded systems, where fewer logs are generated, but operational disruptions can be costly [20]. Furthermore, logs from these smaller-scale systems typically exhibit higher diversity, increasing parsing complexity and requiring greater precision.

**Challenges with current LLM-based parsers:** Current LLM-based parsers strongly depend on large-scale models such as ChatGPT [13], [14] for effective log parsing. While these models have demonstrated state-of-the-art performance, they come with several drawbacks. The most significant challenge lies in their cost and computational demands [21]. These large-scale models can be expensive to deploy, especially in real-time scenarios where quick processing of logs is required. For organizations with limited budgets or small-scale systems, the cost of utilizing large models may be prohibitive. Additionally, the efficiency of these large models in real-time applications is a concern, as their inference times may not meet the necessary speed for time-sensitive anomaly detection tasks. Many smaller, open-source LLMs, which are less resource-intensive, have not been explored thoroughly for the task of log parsing.

**Empirical Study:** To address these limitations, this study presents a comprehensive empirical evaluation of open-source LLM-based log parsers, focusing on lightweight models. In order to assess the effectiveness of these models across a variety of environments, we have constructed six diverse datasets that reflect a range of log data sources. The first dataset is based on the LogHub benchmark, providing a well-established reference for evaluating parser performance. The second and third datasets consist of logs from modern interconnected systems, enabling an assessment of how well the models can handle more complex, contemporary software environments. The final three datasets represent low-resource scenarios, including logs from laboratory setups, early-stage software releases, and embedded systems. These datasets are designed to better reflect the challenges faced by smaller systems, where log volume is limited, and high parsing precision is required. In terms of the models, we have selected ten open-source lightweight LLMs, including Deepseek-R1-1.5B, Gemma-2-2B, Gemma-2-9B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B, and Qwen-2.5-7B. These models represent a broad spectrum of open-source options, offering insights into the performance of mid-sized LLMs on log parsing tasks. Through this comprehensive evaluation, we aim to determine how well these models can perform across different datasets and environments, with a particular focus on how few-shot learning can improve their accuracy in log parsing tasks.

**Main Findings:** Based on our results, we present the following key findings: (1) *Dataset Sensitivity:* The performance of open-source LLM-based parsers varies significantly across datasets, with the highest accuracy observed on benchmark datasets such as LHSB, while performance declines on more diverse and complex datasets like FALL. (2) *Impact of Log Characteristics:* Parsing performance is substantially influenced by log characteristics. In particular, longer log entries and those with a higher frequency of dynamic variables consistently reduce model accuracy. (3) *Effectiveness of Few-Shot Learning:* Both few-shot learning methods consistently improve the parsing accuracy of all tested LLM-based parsers, indicating that model adaptation significantly enhances performance compared to zero-shot scenarios. (4) *Comparison of Few-Shot Methods:* Among the two few-shot learning strategies, the example-based method slightly outperforms the rules-based method, especially on datasets with greater complexity and diversity, such as FALL. Overall, our primary contributions can be summarized in four key points:

- We conducted a comprehensive empirical study to evaluate the performance of open-source lightweight LLM-based parsers with smaller parameters for software log parsing across various datasets and environments.
- We constructed six new datasets, including both benchmark data and log data from modern and low-resource systems, to provide a diverse evaluation for log parsers.
- We are the first to comprehensively evaluate the performance of open-source lightweight LLMs for log parsing,

filling a gap in existing research.

- The extensive experiments demonstrate that lightweight LLM-based parsers can achieve high-accuracy log parsing and exhibit further improvements with two few-shot learning methods.

## II. BACKGROUND

### A. Log Collection

The heterogeneous nature of log structures, stemming from varying logging practices and standards adopted by different developers, introduces considerable variability across collected logs [22]. This heterogeneity poses significant challenges for developing anomaly detection models capable of consistent and accurate performance across diverse logging environments. While platforms such as LogHub [17] have provided numerous open-source datasets, these datasets suffer from several limitations, notably their outdated nature and predominant focus on logs generated by large-scale interconnected systems. As a result, they may fail to capture the characteristics of contemporary software systems or address scenarios involving small-scale or resource-constrained environments [19]. In particular, datasets derived from modern interconnected systems are essential, as these systems often adopt advanced architectures and produce logs with complex and dynamic patterns—posing new challenges for parsing and anomaly detection. Moreover, datasets that reflect low-resource scenarios—such as laboratory environments, early-stage software deployments, and embedded systems—are equally important. These environments typically generate fewer logs, yet exhibit greater variability. Due to frequent software updates and complex production workflows, precise log parsing becomes even more critical in these contexts.

### B. Log Parsing

Log parsing refers to the automatic transformation of raw log messages into structured representations by identifying constant components (event templates) and variable components (parameters) [23]. Research has indicated that over 30% of individual log messages contain more than three variable parameters, with approximately 10% even exceeding five variables [24]. This variability complicates precise parsing, as accurately distinguishing each parameter within these dynamically structured log entries is far from trivial [3]. Prior studies [7], [8] have shown that the quality of log parsing significantly influences the effectiveness of anomaly detection and other downstream log analysis tasks [2], [16]. As such, log parsing is widely regarded as a foundational component of the log analysis pipeline. A variety of log parsing approaches have been proposed in the literature, broadly categorized into two groups: rule-based methods (e.g., Drain [9]) and LLM-based methods. As shown in Figure 1, while rule-based methods are effective in many logging environments, they lack the flexibility required to handle diverse or more complex log formats.

Recently, the emergence of Large Language Models (LLMs), exemplified by ChatGPT [25], has offered promising

Original Log Sequence	LLM-based parsers	Rule-based parsers
generating core.385	generating core.<*>	generating core.<*>
ciid: Error creating node map from file /home/pakini/sweep3d-2.2b/results/random1-8x32x32x2.map: Permission denied	ciid: Error creating node map from file <*>: Permission denied	ciid: Error creating node map from file /home/pakini/sweep3d-2.2b/results/random1-8x32x32x2.map: Permission denied
HTTP exception thrown: No instances found for any event	HTTP exception thrown: No instances found for any event	HTTP exception thrown: No instances found for any event
storage.BlockManager: Found block rdd_2_4 locally	storage.BlockManager: Found block rdd_2_4 locally	storage.BlockManager: Found block rdd_2_4 locally
mod_jk child workerEnv in error state 6	mod_jk child workerEnv in error state 6	mod_jk child workerEnv in error state 6
session opened for user root by (uid=0)	session opened for user <*> by (uid=<*>)	session opened for user root by (uid=0)
ciid: failed to read message prefix on control stream (CioStream socket to 172.16.96.116:33569	ciid: failed to read message prefix on control stream (CioStream socket to <*>	ciid: failed to read message prefix on control stream (CioStream socket to <*><*>
Adding protocol org.apache.hadoop.mapreduce.v2.api.MRClientProtocolPB to the server	Adding protocol org.apache.hadoop.<*> to the server	Adding protocol org.apache.hadoop.mapreduce.v2.api.MRClientProtocolPB to the server
authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser=rhost=61.53.154.93 user=root	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser=rhost=<*> user=<*>	authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser=rhost=<*> user=root

Fig. 1: The performance of the two most commonly used types of parsers. Dynamic variables are highlighted in **red**.

alternatives to rule-based approaches [26], [27]. LogGPT [28] introduces a novel approach to log-based anomaly detection, leveraging ChatGPT-3.5’s language interpretation capabilities. The results show promising performance and strong interpretability, providing valuable insights into the potential of prompt-based models for log data analysis. However, LogGPT’s performance on the BGL dataset is suboptimal, underscoring the need for further improvements to this approach. LogPPT [14] introduces a prompt-based few-shot learning approach for log parsing. This method captures log templates and parameters using minimally labeled data, outperforming traditional techniques in both effectiveness and efficiency across several public log datasets. DivLog [24] utilizes in-context learning with LLMs to enhance log parsing by leveraging diverse, small samples and training-free prompt construction, achieving state-of-the-art performance in accuracy and template precision across multiple large-scale datasets. Additionally, LILAC [13] integrates LLMs with an adaptive parsing cache and ICL, significantly improving template accuracy, parsing efficiency, and reducing reliance on LLM queries. Compared to the rule-based parsers shown in Figure 1, LLM-based log parsers leverage their ability to recognize complex and varied textual patterns, offering significant advantages over traditional methods, especially when parsing logs with high variability. Nevertheless, the practical deployment of such large models is constrained by substantial computational costs and resource requirements, limiting their applicability in real-time or low-resource scenarios. Furthermore, existing research has primarily focused on large parameter LLMs, while the potential of open-source lightweight LLMs remains largely unexplored. This research gap presents an opportunity to investigate whether smaller-scale LLMs, with their increased deployment feasibility, can achieve comparable or superior parsing performance through targeted few-shot learning.

### III. STUDY DESIGN

#### A. Scenarios and Dataset Construction

We constructed six diverse datasets to comprehensively evaluate the performance of log parsing models across different logging environments. The datasets include: (1) the **LogHub Subset Benchmark (LHSB)** dataset, derived from

established benchmarks on the LogHub platform; (2) the **Distributed System Logs (DSL)** dataset, collected from a distributed system environment; (3) the **Large-Scale Web Application Logs (LSWAL)** dataset, representing logs from a modern interconnected system; (4) the **Educational App Interactive Log Dataset (EAILD)**, collected from an educational software deployment; (5) the **Intelligent Laboratory Management System Dataset (ILMSD)**, gathered from a research laboratory management system; and (6) the **Factory Assembly Line Logs (FALL)** dataset, originating from an industrial manufacturing environment. Table I summarizes key details of each constructed dataset.

1) *LHSB Dataset*: For the LHSB dataset, our objective was to construct a raw log corpus that accurately reflects the structural complexity and event distribution characteristic of real-world system operations. While the LogHub platform provides 2,000 manually curated log entries per dataset, these subsets often exclude rare anomalies, particularly those occurring within short intervals or under limited conditions.

To overcome this limitation, we selected five LogHub [17] datasets—HDFS [29], Hadoop [30], OpenStack [31], BGL [32], and Thunderbird [32]—all of which include full log traces accompanied by ground-truth anomaly labels. Our data construction process followed a two-stage strategy: (1) anomaly interval-based extraction and (2) dataset integration with a controlled anomaly ratio.

In the first stage, we extracted all log entries that occurred within the labeled anomalous intervals. For each dataset, the associated metadata files were parsed to determine the start and end timestamps of each abnormal window. All log lines within these intervals were retained in their original format, ensuring comprehensive coverage of both frequent and rare anomalies that may have been omitted in the default benchmark subsets.

In the second stage, we integrated the extracted anomaly logs with the 32,000 log entries sourced from 16 LogHub benchmark subsets. To approximate a realistic class imbalance, we removed a large number of similar normal logs and selectively duplicated a portion of the anomaly logs before merging them into the combined dataset. This resulted in an approximate 9:1 ratio of normal to anomalous entries, reflecting the skewed distribution commonly observed in production environments.

We excluded the remaining LogHub datasets from our processing pipeline due to the absence of reliable anomaly annotations. As a result, our final dataset comprises 9,047 raw log entries, which were selected without applying any template-based parsing or label-based filtering beyond the initial identification of log intervals.

2) *DSL and LSWAL Datasets*: Datasets DSL and LSWAL were collected from contemporary large-scale systems to capture recent log variability. DSL includes logs from a modern distributed microservices architecture, while LSWAL encompasses logs from a major interconnected web application system, both collected over a continuous period in August 2024. For the DSL and LSWAL datasets, which each contained over one million raw log entries, we faced two key challenges:

TABLE I: Summary of constructed datasets.

Dataset	Abbreviation	Category	Amount	Anomalies	Avg. Len (Chars)
LogHub Subset Benchmark	<b>LHSB</b>	Benchmark from LogHub	9,047	961 (10.6%)	173.8
Distributed System Logs	<b>DSL</b>	Modern large-scale system	7,993	1,149 (14.4%)	168.5
Large-Scale Web Application Logs	<b>LSWAL</b>	Modern large-scale system	6,039	813 (13.5%)	185.2
Educational App Interactive Log Dataset	<b>EAILD</b>	Low-resource application	6,285	938 (14.9%)	180.5
Intelligent Laboratory Management System Dataset	<b>ILMSD</b>	Low-resource system	7,480	894 (12.0%)	137.4
Factory Assembly Line Logs	<b>FALL</b>	Low-resource system	5,319	917 (17.2%)	225.8

the overwhelming volume of data and the high redundancy caused by numerous repeated or highly similar normal log messages. To construct smaller, high-quality datasets while preserving diversity, we employed a scalable sampling method based on Determinantal Point Processes (DPP) [33].

At the outset, we used the Sentence-BERT model [34] (specifically all-MiniLM-L6-v2) to encode each log entry into a fixed-size semantic vector. To address the computational overhead of operating on such large-scale embeddings, we applied Principal Component Analysis (PCA) [35] to reduce the vector dimensionality from 384 to 50, retaining the most informative features. We then computed a similarity kernel matrix using cosine similarity [36], which captured the semantic relationships between log entries and laid the foundation for more efficient and meaningful diversity selection through DPP sampling.

To construct a diverse and representative subset, we employed a low-rank approximation of the DPP sampling algorithm, which prioritizes the selection of semantically distinct log messages. This method effectively reduced redundancy while ensuring broad coverage across varying log formats and anomaly types. Through this process, we curated refined datasets in which approximately 90% of log entries represent typical users and system behavior, while the remaining 10% capture more unusual or diverse patterns. The resulting datasets contain 7,993 logs for DSL and 6,039 logs for LSWAL, striking a balance between efficiency, diversity, and representativeness.

3) *Low-resource Scenarios Datasets*: The remaining three datasets (EAILD, ILMSD, FALL) represent low-resource scenarios where logs are limited in volume but diverse in structure.

The **EAILD** dataset was collected from a newly launched educational app in June 2024. The app supports features such as interactive lessons, quizzes, and progress tracking, and is deployed across various mobile devices and networks. Over a focused three-day debugging period, 73,626 log messages were captured, encompassing user interactions, back-end service status, and error events. A significant portion of these logs consisted of near-identical entries that only differed in parameter values (e.g., timestamps or session IDs). To reduce redundancy and enhance the evaluative utility of the dataset, we applied frequency-based filtering to remove excessively repeated log templates while retaining their structural variability. This resulted in a cleaned dataset of 6,285 logs, preserving meaningful diversity without overwhelming the evaluation with trivial variations.

The **ILMSD** dataset was gathered from a laboratory management platform used in chemical and materials research

facilities. The system, developed in Java and instrumented with Apache Log4j, logs system behavior in a structured JSON format to facilitate automation and clarity. Logs include user actions, instrument control commands, job scheduling, and data synchronization processes. Over a 50-hour operation period, 54,637 entries were recorded. Applying the same frequency-aware filtering strategy as with EAILD, we reduced redundant entries and constructed a final dataset of 7,480 logs, maintaining a representative sample of diverse log types within a controlled environment.

The **FALL** dataset was obtained from an operational factory assembly line monitoring system. This environment integrates multiple hardware and software subsystems responsible for tasks such as cutting, assembling, and quality checking of industrial components. Log messages from this system are highly heterogeneous, reflecting real-time hardware status, control commands, and environmental readings. Due to the critical nature of these processes, logs are highly contextual and diverse, with very few instances of repeated structure. We collected all logs generated across three consecutive production days in October 2024, totaling 5,319 entries. Unlike the other datasets, no filtering was necessary due to the natural variability of the logs, making FALL the most structurally diverse and realistic dataset in our evaluation framework.

### B. LLM-based Parsers Design

Smaller parameter open-source lightweight LLMs have shown great potential for log parsing applications due to their lower computational requirements, reduced deployment costs, and greater flexibility for few-shot learning [37]. Moreover, they can meet the necessary speed requirements for time-sensitive anomaly detection tasks. To explore their effectiveness, we evaluate ten representative open-source LLMs on software log parsing tasks.

To evaluate the performance of open-source LLMs [38]–[41] for log parsing, we selected ten models of varying parameter sizes, which include **Deepseek-R1-1.5B**, **Gemma-2-2B**, **Gemma-2-9B**, **Llama-3.2-1B**, **Llama-3.2-3B**, **Llama-3.1-8B**, **Qwen-2.5-0.5B**, **Qwen-2.5-1.5B**, **Qwen-2.5-3B**, and **Qwen-2.5-7B**. As shown in Table II, these models represent a range of small parameter sizes, enabling a comprehensive evaluation across various computational and memory constraints. Leveraging cutting-edge architectures, these models demonstrate strong potential for log parsing applications, particularly as the complexity and diversity of software logs increase. In such contexts, where precise template generation from variable-rich log messages is crucial, these models are well-suited to handle the evolving challenges. In detail, Deepseek-R1-1.5B [38], which is optimized for efficient inference in

general text-processing tasks; Gemma-2-2B and Gemma-2-9B [39], developed by Google DeepMind, emphasizing scalability and robustness for smaller computational environments; and Llama-3.2-1B, Llama-3.2-3B, and Llama-3.1-8B [40] from Meta, known for balancing performance and resource efficiency. Additionally, we assess four models from the Qwen family [41]—Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B, and Qwen-2.5-7B—created by Alibaba Cloud, which are designed to provide strong performance in multilingual and domain-specific tasks while maintaining modest computational demands. Evaluating these diverse models allows for a comprehensive analysis of the potential and limitations of mid-sized open-source LLMs for log parsing applications.

TABLE II: Overview of Open-Source Lightweight LLM-based Parsers.

Model	Size	Training Base (Tokens)	In/Out (Tokens)
Deepseek-R1-1.5B	1.5B	14.8 trillion	64k / 8,192
Gemma-2-2B	2B	2 trillion	8,192 / 8,192
Gemma-2-9B	9B	8 trillion	8,192 / 8,192
Llama-3.2-1B	1B	9 trillion	128k / 2,048
Llama-3.2-3B	3B	9 trillion	128k / 2,048
Llama-3.1-8B	8B	15 trillion	128k / 2,048
Qwen-2.5-0.5B	0.5B	18 trillion	32k / 8,192
Qwen-2.5-1.5B	1.5B	18 trillion	32k / 8,192
Qwen-2.5-3B	3B	18 trillion	32k / 8,192
Qwen-2.5-7B	7B	18 trillion	128k / 8,192

For our log parsing task, we designed **zero-shot prompts** to evaluate the models’ ability to handle log parsing without any prior training on the specific task. The zero-shot prompt was designed as follows:

*You are required to parse the content of the logs and replace the dynamic variables with templates. Output the results directly without further explanation. Please parse the log template from the following log message: [LOG]*

Where *[LOG]* denotes the raw log data to be parsed, this prompt directs the model to identify dynamic components within the log and convert them into structured templates. By utilizing zero-shot prompting, we are able to evaluate the models’ inherent capabilities to perform the task without the need for additional examples. This approach serves as a baseline for comparing the raw performance of each model.

### C. Few-shot Learning

In the following experiments, we assessed the impact of using In-Context Learning (ICL) [42] as a few-shot learning strategy on the performance of ten open-source LLMs in log parsing. Although previous studies have shown that ICL improves the log parsing performance of large models like ChatGPT [43], there has been limited exploration of its impact on the open-source LLMs we selected. Therefore, we aim to further investigate whether ICL enhances these models’ accuracy in parsing logs by leveraging contextual examples to guide their decision-making process. ICL prompting relies on conditioning the model with a small set of examples, which has been demonstrated to improve performance in

tasks requiring multi-step reasoning. In the context of log parsing, ICL prompts assist the model in processing log entries by providing examples, thereby enabling it to identify and extract components such as timestamps, IP addresses, user actions, and error codes. This structured reasoning process is crucial for accurately identifying dynamic variables in logs and replacing them with appropriate templates.

For our specific task, the ICL prompts help the models to carefully examine the different parts of a log message and understand the structural relationships between them. By providing the model with an initial set of Few-Shot examples, we guide the model in learning the correct pattern of how logs should be parsed. This approach allows the model to adapt its parsing behavior to new logs with similar structures, improving its ability to generate accurate log templates even when faced with unseen data. We designed two types of Few-Shot prompts for this purpose:

1) *Example-based Few-Shot Prompt*: In this method, we provided five pairs of original logs and their corresponding parsed log templates to the model. To ensure a diverse range of examples, we curated 1,500 log entries from all datasets using the DPP method. DPP is a selection technique that aims to maximize the diversity of the chosen dataset by selecting logs that are least similar to each other [44]. This ensures that the model is exposed to a wide variety of log formats and patterns, preventing it from overfitting to a specific log structure. The process involves calculating pairwise similarities between all logs in the dataset (using a similarity measure like cosine similarity or Jaccard similarity) and then selecting logs that minimize the overall similarity. This approach iteratively picks the most diverse logs, ensuring the model encounters varied patterns.

After constructing a diverse dataset with DPP, we applied the k-Nearest Neighbors (kNN) algorithm [45] to identify the five most similar logs to a new log entry. In this method, for each new log, the algorithm compares the new log with the curated historical logs and selects the k most similar entries based on a similarity score. By using kNN, we ensure that the model adapts to new log patterns and refines its parsing ability based on past examples. The prompt presented to the models during this phase is as follows:

*You are required to parse the content of the logs and replace the dynamic variables with templates. Output the results directly without further explanation. Here are some examples:  
Original Log: "[Original log examples 1, 2, 3, .....]"  
Parsed Log: "[Parsed log examples 1, 2, 3, .....]"  
Please parse the log template from the following log message: [LOG]*

Where *[Original log examples]* denotes an actual log message that contains dynamic variables such as timestamps, IP addresses, and user IDs, and *[Parsed log example]* denotes the output log with these variables replaced by structured templates like "<\*>".

2) *Replacement Rules-based Few-Shot Prompt*: In this method, we provided a set of explicit replacement rules that the model must follow to ensure consistent and accurate parsing



of logs. These rules help standardize how specific dynamic variables should be replaced with templates. The predefined replacement rules guide the model to recognize and replace components like timestamps, IP addresses, and user IDs, which are commonly encountered in log data. For instance:

*Replace all timestamps (e.g., [Examples]) with "TIME";  
Replace all IP addresses (e.g., [Examples]) with "IP";  
Replace user IDs (e.g., [Examples]) with "USER".*

Where [Examples] denotes specific examples of dynamic variables, we select the types of dynamic variables present in different datasets, such as timestamps, IP addresses, and user IDs. These rules serve as clear instructions, reducing ambiguity in the log parsing process and ensuring the model parses logs in a structured and consistent manner, regardless of the specific log format. By offering these predefined rules, we guide the model’s parsing decisions, allowing it to focus on specific elements that require replacement while ignoring irrelevant components. The prompt presented to the models, incorporating the replacement rules, is as follows:

*You are required to parse the content of the logs and replace the dynamic variables with templates. Output the results directly without further explanation. Please refer to these replacement rules: Replace all timestamps (e.g., [Examples 1, 2, 3, .....]) with "TIME"; Replace all IP addresses (e.g., [Examples 1, 2, 3, .....]) with "IP"; Replace user IDs (e.g., [Examples 1, 2, 3, .....]) with "USER".*  
.....  
*Please parse the log template from the following log message: [LOG]*

#### D. Experimental Design and Research Questions

In this section, we will present the designed research questions and the experimental setup for each research question.

##### **RQ1: How do the ten open-source LLM-based parsers perform across six datasets?**

In this study, we aim to evaluate the performance of ten open-source lightweight LLM-based parsers across six distinct datasets, representing a diverse range of log data. This allows us to benchmark the models’ performance on both high-volume, large-scale logs and smaller, more varied log entries. We apply each of the ten models—Deepseek-R1-1.5B, Gemma-2-2B, Gemma-2-9B, Llama-3.2-1B, Llama-3.2-3B, Llama-3.1-8B, Qwen-2.5-0.5B, Qwen-2.5-1.5B, Qwen-2.5-3B, and Qwen-2.5-7B—to parse logs from the following six datasets: LHSB, DSL, LSWAL, EAILD, ILMSD, and FALL. In addition, we adopt GPT-3.5-Turbo, provided by OpenAI, as a benchmark large-parameter LLM for log parsing. We evaluate the parsing accuracy by measuring the models’ ability to replace dynamic variables with the correct templates across various log formats.

##### **RQ2: What factors of log entries influence log parsing performance?**

To explore which factors of log entries impact parsing performance, we examine a range of log characteristics, including the length of log entries and the frequency of dynamic variables. For the length of log entries, we categorize the logs

from the six datasets into three groups based on their character count: (1) logs with a length of 100 characters or fewer, (2) logs with lengths ranging from 100 to 200 characters, and (3) logs longer than 200 characters. This classification allows us to create a new evaluation dataset based on log length, which enables us to determine how the length of logs influences parsing accuracy.

Additionally, some logs, despite being long, may not present significant challenges for the models if they contain few dynamic variables. To account for this, we introduce another factor: the frequency of dynamic variables. We divide the logs into three categories based on the number of dynamic variables they contain: (1) 1 to 2 variables, (2) 3 to 4 variables, and (3) 5 or more variables. This categorization allows us to create a second evaluation dataset that reflects how the number of dynamic variables influences log parsing performance.

We then assess how each of these factors—log length and frequency of dynamic variables—affects the parsing accuracy of the models across different datasets. To avoid excessive repetition of similar logs (i.e., logs with only dynamic variables differing), which could affect the evaluation metrics, we removed a portion of logs with identical formats, ensuring that no more than five instances of each log type remain. By analyzing these factors, we aim to identify which log characteristics have the greatest impact on log parsing performance, offering insights into how the performance of LLM-based parsers varies with different log structures.

##### **RQ3: To what extent does few-shot learning improve log parsing performance?**

In this study, we explore the impact of few-shot learning on the performance of LLM-based parsers for log parsing tasks. To evaluate this, we fine-tune the selected models using the methods described in Section III-C, where the models are trained to improve their log parsing performance. We compare the performance of two types of fine-tuned models—few-shot prompts (example-based and replacement rules-based)—with their zero-shot performance across the same six datasets. Few-shot learning is expected to enhance parsing accuracy by enabling the models to adapt more specifically to the types of logs and the unique characteristics of the datasets, allowing us to quantify the benefits of model adaptation in log parsing.

#### E. Evaluation Metrics

To evaluate the performance of the ten open-source LLM-based parsers in log parsing, we follow recent studies [13], [16] and use PA and FTA as the evaluation metrics.

1) *Parsing Accuracy (PA)*: PA measures the ability of the parser to accurately extract templates from the log entries, which is essential for downstream tasks such as anomaly detection and log analysis. PA is defined as the proportion of correctly parsed log messages to the total number of log messages. A log message is considered correctly parsed if, and only if, all tokens of the templates and variables are correctly identified.  $PA = \frac{\text{Number of correctly parsed logs}}{\text{Total number of logs}}$ , where a "correctly parsed log" means that all dynamic variables are accurately identified and replaced.

2) *F1-score of Template Accuracy (FTA)*: FTA is a template-level metric that evaluates the correctness of identified templates in parsed logs. It is based on the precision and recall of the templates extracted from log messages. The FTA is calculated as the harmonic mean of Precision and Recall, which are computed specifically for the templates. A template is considered correctly identified if and only if the parsed log shares the same template as the ground-truth template, and all tokens of the parsed template match exactly with those of the ground-truth template.  $\text{Precision} = \frac{\text{Number of correctly identified templates}}{\text{Number of parsed templates}}$ ,  $\text{Recall} = \frac{\text{Number of correctly identified templates}}{\text{Number of ground-truth templates}}$ . Thus,  $\text{FTA} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$ , where the FTA provides a balanced measure of template identification accuracy, considering both the ability to identify templates correctly (Precision) and the ability to capture all ground-truth templates (Recall).

#### IV. STUDY RESULTS

In this section, we present the experimental results and provide an analysis to address each research question.

##### A. Effectiveness of Ten LLM-based Parsers (RQ1)


In this section, we evaluate the performance of ten open-source lightweight LLM-based parsers, as well as the benchmark large-parameter model GPT-3.5-Turbo, across six distinct datasets representing a diverse range of log data. As shown in Table III, we assess their performance using the PA and FTA metrics, which measure the models' ability to replace dynamic variables with the correct templates across various log formats.

Overall, the performance of the models shows significant variability across different datasets. The LHSB yields the highest results across most models, which may suggest that the dataset was well-exposed to the models during training. This observation could point to a potential data leakage issue, where the models have previously encountered similar logs or the logs are based on older formats that the models are more familiar with. In contrast, the FALL dataset, which represents the most diverse and complex sample set, resulted in the lowest performance across all models. This highlights the challenges posed by datasets with higher variability and diverse log structures, where the models struggle to generalize beyond their training data. Moreover, the benchmark large-parameter model GPT-3.5-Turbo does not exhibit a clear advantage over the lightweight LLM-based parsers. Its PA and FTA scores are only 2.5% and 2.7% higher, respectively, than those of the best-performing lightweight model, Deepseek-R1-1.5B. The most noticeable differences appear on the LHSB and ILMSD datasets, both of which have higher overall scores compared to the other four datasets, indicating that GPT-3.5-Turbo performs relatively better in more structured or less diverse log environments, but its advantage diminishes in complex or low-resource scenarios.

Among the models tested, Deepseek-R1-1.5B and Qwen-2.5-7B consistently perform the best across most datasets. Specifically, Qwen-2.5-7B exhibits a unique behavior on the

EAILD dataset, where the FTA score exceeds the PA score. This anomaly suggests that while Qwen-2.5-7B excels in identifying templates, it may still face challenges in precisely replacing dynamic variables in the logs. This indicates that the model might be identifying templates more accurately but not yet replacing all dynamic components with the expected precision. Deepseek-R1-1.5B, on the other hand, shows robust and balanced performance with high scores in both PA and FTA, which suggests that it handles the log parsing task more effectively across diverse log formats.

The differences in performance across datasets highlight that LLM-based parsers do not exhibit strong generalization capabilities, especially under zero-shot conditions. While the models perform well on benchmark datasets like LHSB, they struggle to maintain consistent performance when parsing logs from more diverse and complex datasets, such as FALL. This suggests that, in zero-shot settings, the models heavily rely on prior knowledge from similar logs or outdated formats that they encountered during training. As a result, the models may fail to generalize well to new and more complex log structures, indicating a limitation in their ability to adapt to novel log data without further few-shot learning or domain-specific training. This also underscores the importance of improving the generalization ability of LLM-based parsers, particularly for real-world log parsing tasks where log formats can vary greatly.

 **Answer to RQ1:** The evaluation of the ten open-source LLM-based parsers across six datasets reveals significant performance variability, with models performing best on the LogHub Subset Benchmark (LHSB) and struggling with the more diverse and complex logs in the FALL dataset.

##### B. Ablation Study of Different Factors (RQ2)

In this section, we examine the impact of log characteristics on parsing performance by focusing on two factors: the length of log entries and the frequency of dynamic variables. Detailed classification criteria can be found in Section III-D. We categorize the logs from the six datasets into three groups based on log entry length: short logs, medium-length logs, and long logs. The corresponding results are presented in Table IV. Additionally, we classify the logs by the frequency of dynamic variables into three groups: logs with 1–2 variables, 3–4 variables, and 5 or more variables. The results for this grouping are shown in Table V.


Overall, the length of log entries significantly impacts parsing performance. Short logs and medium-length logs show relatively similar performance across the models, with short logs slightly outperforming the medium-length logs. In fact, for short logs, most models perform better than their results in RQ1, suggesting that shorter logs may be easier for LLM-based parsers to handle. However, the performance of all models shows a consistent decline when processing long logs, with a more noticeable drop in performance on the FALL dataset. This decline is likely due to the increased complexity and variability of longer logs, which present more challenges in correctly identifying and replacing dynamic variables. Logs

TABLE III: The performance of ten open-source LLM-based parsers across six datasets (%).

Model	LHSB		DSL		LSWAL		EAILD		ILMSD		FALL		Average	
	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA
Deepseek-R1-1.5B	<b>80.2</b>	76.7	<b>76.3</b>	68.4	<b>74.9</b>	<b>70.3</b>	<b>72.5</b>	66.7	<b>77.3</b>	<b>74.8</b>	<b>68.4</b>	<b>62.5</b>	<b>74.5</b>	69.9
Gemma-2-2B	65.2	60.8	61.3	56.2	61.2	56.1	54.3	53.2	65.2	59.8	53.7	49.1	60.2	55.9
Gemma-2-9B	74.2	72.1	70.7	65.3	70.1	66.9	65.2	62.8	72.8	70.4	63.7	58.6	69.5	66.0
Llama-3.2-1B	66.6	63.2	65.4	59.8	63.7	56.9	58.0	56.8	70.2	62.4	56.9	50.4	63.5	58.3
Llama-3.2-3B	62.7	57.2	58.9	52.5	59.4	54.6	52.8	50.4	63.6	57.9	50.2	46.7	58.0	53.2
Llama-3.1-8B	70.6	67.3	67.5	62.1	68.4	61.7	60.1	60.2	71.2	67.9	60.5	53.8	66.4	62.2
Qwen-2.5-0.5B	61.3	56.8	56.3	51.2	57.9	52.4	50.2	50.3	62.2	58.3	48.3	41.6	56.0	51.8
Qwen-2.5-1.5B	68.2	64.1	64.9	59.3	65.2	59.1	57.2	57.9	68.5	63.6	58.1	51.2	63.7	59.2
Qwen-2.5-3B	73.9	72.8	71.2	66.8	69.4	67.5	63.1	64.5	70.2	68.6	61.1	57.0	68.2	66.2
Qwen-2.5-7B	78.5	<b>77.4</b>	74.6	<b>69.9</b>	72.5	70.2	68.8	<b>69.2</b>	74.6	73.1	66.0	60.2	72.5	<b>70.0</b>
Average	70.1	66.8	66.7	61.2	66.3	61.6	60.2	59.2	69.6	65.7	58.7	53.1	65.3	61.3
GPT-3.5 (Baseline)	83.7	80.5	77.4	71.2	76.1	70.9	74.4	70.4	80.6	78.2	69.9	64.0	77.0	72.6

in the FALL dataset are more diverse in terms of structure and content, further exacerbating parsing difficulties for longer logs.

Similarly, the frequency of dynamic variables plays a critical role in model performance. Logs containing five or more dynamic variables pose significant challenges for all models, with overall performance lower than that observed for long logs. This is especially evident in the FALL dataset, where the FTA is only 46.9%, which is 6.2% lower than the performance observed in Section IV-A. Since a log is considered correctly parsed only when all dynamic components are accurately identified and replaced, the presence of multiple variables increases parsing difficulty significantly. This challenge is particularly pronounced when handling logs with five or more dynamic variables, where models generally perform poorly across all datasets, including advanced models such as Deepseek-R1-1.5B. The difficulty arises from the complexity of ensuring accurate replacements for multiple variables within a single log entry, demanding higher precision from the models. Thus, future research should focus on developing parsing methods specifically tailored to effectively handle log entries characterized by high complexity and numerous dynamic variables.


 **Answer to RQ2:** The ablation study shows that log length and the frequency of dynamic variables significantly impact the performance of LLM-based parsers, with longer logs and logs containing more dynamic variables leading to a noticeable decline in parsing accuracy.

### C. Impact of Few-shot Learning on Log Parsing Performance (RQ3)

In this section, we examine the impact of few-shot learning on the performance of LLM-based parsers in log parsing tasks. To evaluate this, we fine-tune the selected models using two approaches: (1) Example-Based Few-Shot Prompting, and (2) Replacement Rules-Based Few-Shot Prompting. The example-based approach provides the model with a set of original logs and their corresponding parsed templates as demonstrations, whereas the rules-based approach supplies explicit replacement rules to guide the parsing process. As shown in Table VI, we compare the PA performance of the two fine-tuned

methods with that of their zero-shot counterparts across the same six datasets.

Overall, few-shot learning leads to significant improvements in parsing performance across all models, regardless of the specific method used. The fine-tuned models consistently outperform their zero-shot counterparts across all datasets, with an average PA increase of 7.46%, highlighting that few-shot learning enables the models to adapt more effectively to the nuances of log parsing tasks. This adaptation is especially noticeable in complex log formats, such as those found in three low-resource datasets, where the models better understand the structure and context of log messages. While both few-shot learning methods—example-based (average PA improvement: 7.71%) and rules-based (average PA improvement: 7.21%)—result in performance gains, the example-based method generally yields slightly superior results. The difference between the two methods is not substantial, suggesting that both approaches effectively help the models adjust to the task. Upon further inspection, we found that the example-based method significantly outperforms the rules-based method when handling logs with five or more dynamic variables, indicating its advantage in processing more complex logs. Additionally, on the FALL dataset, which contains more diverse and complex log samples, both fine-tuned methods show substantial improvements, with average PA increasing by 13.84%. This emphasizes the necessity of few-shot learning in LLM-based parsers, enabling the model to make more precise and context-aware template replacements.

 **Answer to RQ3:** Few-shot learning significantly improves log parsing performance across all models, particularly in datasets with greater sample diversity. The example-based approach outperforms the rules-based method, though the difference is slight.

## V. IMPLICATIONS

The findings from this study provide significant insights into the effectiveness and limitations of lightweight LLM-based parsers for log parsing, particularly in the context of real-world log data. One key implication is that the performance of log parsing models is highly dependent on the characteristics of the log entries, such as log length and the frequency of



TABLE IV: The performance of ten open-source LLM-based parsers in terms of log entry length (%).

Model	Length	LHSB		DSL		LSWAL		EAILD		ILMSD		FALL		Average	
		PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA
Deepseek-R1-1.5B	Short	<b>83.6</b>	<b>80.2</b>	<b>79.9</b>	<b>75.1</b>	<b>77.3</b>	<b>74.8</b>	75.0	70.2	<b>80.6</b>	<b>77.5</b>	<b>78.8</b>	<b>76.3</b>	<b>76.2</b>	<b>76.6</b>
	Medium	<b>80.7</b>	77.1	<b>75.8</b>	68.6	<b>73.7</b>	70.8	<b>71.3</b>	66.0	<b>76.4</b>	<b>72.9</b>	<b>71.4</b>	<b>64.2</b>	<b>71.5</b>	<b>70.4</b>
	Long	<b>78.5</b>	75.1	<b>73.6</b>	66.8	<b>71.3</b>	<b>69.1</b>	<b>70.0</b>	64.4	<b>73.6</b>	<b>72.9</b>	<b>62.8</b>	<b>59.1</b>	<b>67.4</b>	<b>66.7</b>
Gemma-2-2B	Short	67.4	62.9	63.7	60.9	64.6	59.4	57.3	55.8	67.5	62.9	64.1	61.7	61.2	62.2
	Medium	66.9	61.4	62.8	58.5	60.7	57.2	55.7	54.1	62.8	58.0	56.8	52.1	56.7	56.7
	Long	63.8	57.6	58.3	54.1	57.4	52.9	51.7	50.6	62.7	57.3	48.2	44.3	53.9	52.8
Gemma-2-9B	Short	77.6	74.9	73.2	69.8	73.0	68.5	68.3	64.9	74.4	72.8	72.5	70.3	70.2	70.9
	Medium	75.1	73.2	71.3	65.1	68.6	65.3	66.7	63.4	74.2	69.8	65.1	60.0	67.4	66.7
	Long	71.8	70.6	68.3	62.8	67.2	63.1	62.9	60.2	70.2	68.8	59.1	55.7	62.9	62.8
Llama-3.2-1B	Short	68.3	65.8	67.9	64.1	65.8	59.4	60.0	57.3	72.8	64.9	65.8	63.5	63.4	64.6
	Medium	67.2	64.1	66.7	60.4	64.3	57.6	56.4	55.2	69.2	61.9	58.7	51.3	58.6	59.2
	Long	63.8	61.6	62.7	56.4	61.9	54.3	55.7	53.9	68.2	60.6	50.3	47.2	57.2	56.2
Llama-3.2-3B	Short	63.9	60.2	61.7	58.2	63.1	58.0	54.6	53.9	66.4	60.9	59.7	58.3	58.8	59.7
	Medium	63.6	58.4	60.1	54.5	61.7	55.3	52.4	49.6	64.2	58.5	53.4	49.8	55.6	55.2
	Long	60.1	55.7	55.2	50.3	56.8	53.2	50.1	47.6	60.6	54.0	44.1	40.3	50.0	49.4
Llama-3.1-8B	Short	72.6	68.8	69.3	66.9	70.3	62.9	62.5	61.8	74.6	69.2	68.4	66.8	66.6	67.9
	Medium	70.0	66.9	68.2	62.7	67.3	62.0	58.9	58.1	70.3	66.8	62.8	56.7	62.2	62.8
	Long	67.4	64.9	65.2	60.3	65.7	58.6	57.4	57.1	68.4	66.0	52.8	49.3	60.1	58.9
Qwen-2.5-0.5B	Short	63.5	58.2	57.4	55.9	59.6	54.8	53.4	52.0	65.7	60.9	57.3	53.9	56.9	57.8
	Medium	61.5	56.2	57.4	52.9	59.2	53.1	49.6	49.3	60.1	59.4	51.2	44.8	53.8	53.1
	Long	58.0	53.4	52.2	49.3	54.1	49.5	47.3	48.0	59.3	54.0	42.5	38.4	48.4	48.4
Qwen-2.5-1.5B	Short	70.2	66.6	67.2	64.7	67.3	62.9	60.4	58.5	69.9	65.8	66.7	64.3	64.0	64.9
	Medium	67.8	63.6	64.2	60.5	66.8	60.3	56.5	58.3	69.2	61.7	60.6	54.8	60.2	60.8
	Long	65.2	62.5	62.7	57.3	63.1	56.9	54.8	54.1	66.4	61.7	53.1	48.6	57.8	57.0
Qwen-2.5-3B	Short	75.6	74.7	74.5	71.3	72.0	69.1	64.8	65.8	72.6	70.2	69.7	68.3	68.6	69.2
	Medium	74.3	73.0	71.1	66.2	68.8	66.9	64.2	64.8	72.1	67.5	64.2	59.6	66.6	65.8
	Long	70.8	70.1	69.4	64.5	67.2	65.1	60.5	62.3	68.4	65.9	55.4	54.1	61.7	61.3
Qwen-2.5-7B	Short	81.4	80.0	77.8	74.5	76.3	73.7	<b>75.8</b>	<b>71.7</b>	78.1	76.9	75.7	73.9	75.3	75.3
	Medium	79.3	<b>77.7</b>	74.2	<b>69.0</b>	72.9	<b>71.5</b>	67.9	<b>68.5</b>	73.2	72.5	70.3	62.8	69.5	69.3
	Long	76.1	<b>75.3</b>	72.2	<b>67.1</b>	70.3	67.6	66.2	<b>65.8</b>	71.9	72.4	60.3	57.6	<b>67.4</b>	65.8
Average	Short	72.4	69.3	69.2	66.1	68.9	64.4	63.2	61.2	72.3	68.2	67.9	65.7	-	-
	Medium	70.6	67.2	67.2	61.8	66.4	62.0	60.0	58.7	69.2	64.9	61.5	55.6	-	-
	Long	67.6	64.7	64.0	58.9	63.5	59.0	57.7	56.4	67.0	64.4	52.9	49.5	-	-

dynamic variables. Our results demonstrate that the performance of lightweight LLM-based parsers is comparable to that of the large-parameter model ChatGPT. While these models perform well on simpler, more uniform datasets like LHSB, they struggle with more complex and diverse logs, such as those in the FALL dataset. This highlights the need to enhance the generalization capability of LLM-based parsers, as their current performance is limited by the challenges posed by modern software systems, which involve more diverse and dynamic log formats.

Another important implication is the significant role of few-shot learning in enhancing log parsing accuracy. Our study demonstrates that few-shot learning, particularly the example-based method, leads to substantial improvements in model performance compared to zero-shot configurations. This finding highlights the importance of tailoring models to specific log types, as few-shot learning enables models to better adapt to the structure and content of logs in real-world scenarios. By illustrating the effectiveness of few-shot learning, this study underscores that generic models may not perform optimally in specialized tasks like log parsing, where the diversity and complexity of log data necessitate more targeted approaches.

Finally, our study also highlights the challenges posed by logs with high numbers of dynamic variables, which

significantly hinder model performance. This reinforces the importance of understanding how the complexity of log data impacts parsing accuracy, and it points to the critical need for models to handle diverse and highly variable log entries. These findings contribute to the growing body of literature on log analysis, underscoring the necessity of developing more robust and accurate log parsing systems capable of addressing the complexities of modern log data.

## VI. THREATS TO VALIDITY

### A. Data Quality and Representativeness

The LHSB dataset may introduce biases due to its relatively outdated log format, which the models might have already encountered during pretraining. This could lead to artificially high performance on this dataset, particularly when compared to others like FALL, which contains more diverse and complex log entries. To mitigate this threat, we utilize multiple datasets from varied domains to better evaluate the models' ability to generalize across different log formats and environments.

### B. Few-shot Learning Method Generalization

The two few-shot learning approaches—example-based and replacement rules-based—may not encompass all possible strategies for adapting models to log parsing tasks. While both

TABLE V: The performance of ten open-source LLM-based parsers in terms of the frequency of dynamic variables (%).

Model	Variable	LHSB		DSL		LSWAL		EAILD		ILMSD		FALL		Average	
		PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA	PA	FTA
Deepseek-R1-1.5B	1-2	<b>84.9</b>	82.5	<b>81.2</b>	<b>76.0</b>	<b>79.6</b>	<b>75.4</b>	77.1	73.5	<b>82.4</b>	<b>79.3</b>	<b>80.2</b>	<b>77.9</b>	<b>80.9</b>	<b>80.1</b>
	3-4	80.3	77.6	<b>75.2</b>	67.3	<b>73.8</b>	71.4	<b>70.0</b>	65.2	<b>75.8</b>	<b>72.0</b>	<b>70.8</b>	65.9	<b>74.3</b>	<b>71.8</b>
	$\geq 5$	<b>76.3</b>	73.8	<b>71.5</b>	64.9	<b>69.7</b>	<b>66.9</b>	<b>67.2</b>	62.0	<b>71.5</b>	<b>70.2</b>	<b>59.4</b>	<b>56.8</b>	<b>69.3</b>	<b>65.4</b>
Gemma-2-2B	1-2	69.8	63.7	65.2	62.5	65.9	60.8	59.2	58.1	69.4	63.6	67.3	62.9	66.1	65.9
	3-4	65.2	60.6	61.4	57.2	59.4	56.6	54.7	53.2	63.5	60.4	55.2	51.4	59.9	58.1
	$\geq 5$	60.2	54.5	56.4	52.	55.2	50.7	49.6	48.3	60.5	55.9	45.7	40.9	54.6	51.5
Gemma-2-9B	1-2	78.3	75.6	75.4	71.3	76.2	70.5	70.9	66.7	75.9	74.0	75.8	73.1	75.4	74.8
	3-4	74.7	73.1	70.9	66.2	69.4	66.2	65.4	62.9	73.9	69.2	64.8	59.2	69.9	67.4
	$\geq 5$	68.3	64.7	64.3	60.2	65.5	61.9	60.7	58.1	68.7	65.3	56.4	51.9	64.0	61.3
Llama-3.2-1B	1-2	70.3	66.3	69.7	65.9	66.4	60.7	64.2	60.1	74.9	66.3	68.9	66.5	69.1	69.1
	3-4	67.0	64.8	67.4	60.0	64.1	57.2	55.6	53.7	67.1	60.3	57.2	50.3	63.1	59.6
	$\geq 5$	61.4	59.9	60.2	55.8	59.5	52.7	53.1	50.2	66.4	58.2	48.5	45.3	58.2	55.0
Llama-3.2-3B	1-2	66.8	62.4	65.3	60.7	65.9	60.4	57.9	55.4	69.2	64.4	63.5	61.6	64.8	64.7
	3-4	63.0	57.9	60.4	54.2	62.0	56.1	51.2	50.4	63.1	57.2	55.1	50.5	59.1	57.0
	$\geq 5$	57.8	52.5	53.6	50.6	54.2	51.7	49.3	46.0	58.4	51.7	42.9	38.6	52.7	48.9
Llama-3.1-8B	1-2	74.7	70.1	72.4	68.6	72.5	64.7	64.9	64.2	77.4	71.3	70.6	68.9	72.1	72.1
	3-4	70.2	67.3	67.6	61.4	65.2	61.9	57.3	56.4	71.1	67.3	61.4	55.7	65.5	64.2
	$\geq 5$	64.9	62.5	63.8	57.5	63.1	56.3	55.3	54.0	66.2	64.5	50.5	47.0	60.6	57.8
Qwen-2.5-0.5B	1-2	65.7	60.4	59.2	56.6	60.3	55.9	55.7	53.6	62.3	62.8	59.4	54.2	60.4	59.8
	3-4	60.3	55.7	56.3	51.2	58.4	52.6	48.2	48.5	59.7	57.2	51.4	45.2	55.7	53.8
	$\geq 5$	56.5	52.6	51.5	47.8	53.1	48.2	46.4	47.3	58.1	53.2	41.0	37.1	51.1	48.1
Qwen-2.5-1.5B	1-2	71.9	67.3	69.4	66.2	68.7	63.6	62.8	60.1	70.8	66.9	68.8	66.5	68.7	68.4
	3-4	66.8	62.7	63.9	59.4	64.5	59.8	57.4	58.8	70.3	62.5	60.4	53.6	63.9	62.1
	$\geq 5$	63.1	60.5	60.3	54.8	62.0	54.1	52.1	51.8	63.2	59.3	50.1	46.5	58.5	55.5
Qwen-2.5-3B	1-2	77.9	75.8	76.3	74.1	74.2	70.6	67.2	68.0	74.5	72.9	71.3	70.0	73.6	72.5
	3-4	73.3	72.1	70.3	67.9	69.2	67.4	66.0	66.0	73.5	69.8	65.1	59.4	69.6	67.5
	$\geq 5$	68.8	67.9	66.8	63.2	64.1	63.7	58.2	60.4	65.9	62.1	52.8	51.4	62.8	59.0
Qwen-2.5-7B	1-2	83.1	<b>82.7</b>	79.3	75.8	77.9	75.0	<b>78.0</b>	<b>77.5</b>	80.1	78.2	77.3	75.2	79.3	78.1
	3-4	<b>80.7</b>	<b>78.4</b>	<b>75.2</b>	<b>69.8</b>	73.6	<b>72.2</b>	66.3	<b>67.8</b>	72.7	71.9	<b>70.8</b>	<b>67.2</b>	73.2	71.2
	$\geq 5$	74.3	<b>74.0</b>	70.5	<b>66.4</b>	67.2	64.8	63.5	<b>64.1</b>	68.1	68.0	57.8	55.2	66.9	63.2
Average	1-2	73.1	71.2	69.9	66.0	68.8	65.5	63.2	63.7	70.8	67.8	62.9	59.5	-	-
	3-4	70.2	67.0	66.9	61.5	66.0	62.1	59.2	58.3	69.1	64.8	61.2	55.8	-	-
	$\geq 5$	65.2	62.3	61.9	57.3	61.4	57.1	55.5	54.2	64.7	60.8	50.5	46.9	-	-

TABLE VI: The PA performance of the two fine-tuned methods compared to their zero-shot counterparts (%).

Model	LHSB		DSL		LSWAL		EAILD		ILMSD		FALL	
	E-B	RR-B	E-B	RR-B	E-B	RR-B	E-B	RR-B	E-B	RR-B	E-B	RR-B
Deepseek-R1-1.5B	85.6 ( $\uparrow$ 5.4)	84.8 ( $\uparrow$ 4.6)	81.8 ( $\uparrow$ 4.8)	80.7 ( $\uparrow$ 4.4)	79.5 ( $\uparrow$ 4.6)	79.2 ( $\uparrow$ 4.3)	81.3 ( $\uparrow$ 8.8)	80.6 ( $\uparrow$ 8.1)	84.2 ( $\uparrow$ 6.9)	83.7 ( $\uparrow$ 6.4)	81.0 ( $\uparrow$ 12.6)	79.9 ( $\uparrow$ 11.5)
Gemma-2-2B	72.4 ( $\uparrow$ 7.2)	71.5 ( $\uparrow$ 6.3)	67.4 ( $\uparrow$ 6.1)	66.9 ( $\uparrow$ 5.6)	67.6 ( $\uparrow$ 6.4)	67.1 ( $\uparrow$ 5.9)	62.5 ( $\uparrow$ 8.2)	62.3 ( $\uparrow$ 8.0)	73.7 ( $\uparrow$ 8.5)	73.4 ( $\uparrow$ 8.2)	72.2 ( $\uparrow$ 18.5)	71.5 ( $\uparrow$ 17.8)
Gemma-2-9B	78.8 ( $\uparrow$ 4.6)	77.9 ( $\uparrow$ 3.7)	74.3 ( $\uparrow$ 3.9)	74.2 ( $\uparrow$ 3.8)	74.3 ( $\uparrow$ 4.2)	74.1 ( $\uparrow$ 4.0)	71.9 ( $\uparrow$ 6.7)	71.3 ( $\uparrow$ 6.1)	78.5 ( $\uparrow$ 5.7)	77.9 ( $\uparrow$ 5.1)	76.4 ( $\uparrow$ 12.7)	75.9 ( $\uparrow$ 12.2)
Llama-3.2-1B	74 ( $\uparrow$ 7.4)	73.1 ( $\uparrow$ 6.5)	70.7 ( $\uparrow$ 5.3)	70.4 ( $\uparrow$ 5.0)	69.6 ( $\uparrow$ 5.9)	69.1 ( $\uparrow$ 5.4)	65.5 ( $\uparrow$ 7.5)	65.1 ( $\uparrow$ 7.1)	76.6 ( $\uparrow$ 6.4)	76.4 ( $\uparrow$ 6.2)	70.2 ( $\uparrow$ 13.3)	69.6 ( $\uparrow$ 12.7)
Llama-3.2-3B	68.9 ( $\uparrow$ 6.2)	68.2 ( $\uparrow$ 5.5)	64.2 ( $\uparrow$ 5.3)	64.0 ( $\uparrow$ 5.1)	64.5 ( $\uparrow$ 5.1)	64.1 ( $\uparrow$ 4.7)	60.2 ( $\uparrow$ 7.4)	59.6 ( $\uparrow$ 6.8)	70.5 ( $\uparrow$ 6.9)	70.4 ( $\uparrow$ 6.8)	64.1 ( $\uparrow$ 13.9)	63.4 ( $\uparrow$ 13.2)
Llama-3.1-8B	75.2 ( $\uparrow$ 4.6)	74.7 ( $\uparrow$ 4.1)	71.5 ( $\uparrow$ 4.0)	71.1 ( $\uparrow$ 3.6)	72.9 ( $\uparrow$ 4.5)	72.4 ( $\uparrow$ 4.0)	67.1 ( $\uparrow$ 7.0)	67.3 ( $\uparrow$ 7.2)	76.4 ( $\uparrow$ 5.2)	76.5 ( $\uparrow$ 5.3)	75.0 ( $\uparrow$ 14.5)	74.9 ( $\uparrow$ 14.4)
Qwen-2.5-0.5B	68.1 ( $\uparrow$ 6.8)	67.2 ( $\uparrow$ 5.9)	61.2 ( $\uparrow$ 4.9)	61.2 ( $\uparrow$ 4.9)	64.4 ( $\uparrow$ 6.5)	63.3 ( $\uparrow$ 5.4)	58.4 ( $\uparrow$ 8.2)	58.0 ( $\uparrow$ 7.8)	68.6 ( $\uparrow$ 6.4)	68.4 ( $\uparrow$ 6.2)	60.8 ( $\uparrow$ 12.5)	59.9 ( $\uparrow$ 11.6)
Qwen-2.5-1.5B	75.3 ( $\uparrow$ 7.1)	74.9 ( $\uparrow$ 6.7)	72.1 ( $\uparrow$ 7.2)	71.5 ( $\uparrow$ 6.6)	71.4 ( $\uparrow$ 6.2)	70.9 ( $\uparrow$ 5.7)	66.1 ( $\uparrow$ 8.9)	65.6 ( $\uparrow$ 8.4)	75.5 ( $\uparrow$ 7.0)	75.0 ( $\uparrow$ 6.5)	71.8 ( $\uparrow$ 13.7)	71.2 ( $\uparrow$ 13.1)
Qwen-2.5-3B	80.2 ( $\uparrow$ 6.3)	79.6 ( $\uparrow$ 5.7)	77.3 ( $\uparrow$ 6.1)	76.9 ( $\uparrow$ 5.7)	75.2 ( $\uparrow$ 5.8)	74.6 ( $\uparrow$ 5.2)	72.2 ( $\uparrow$ 9.1)	71.5 ( $\uparrow$ 8.4)	77.7 ( $\uparrow$ 7.5)	77.2 ( $\uparrow$ 7.0)	75.6 ( $\uparrow$ 14.5)	75.1 ( $\uparrow$ 14.0)
Qwen-2.5-7B	86.0 ( $\uparrow$ 7.5)	85.8 ( $\uparrow$ 7.3)	80.6 ( $\uparrow$ 6.0)	80.2 ( $\uparrow$ 5.6)	77.8 ( $\uparrow$ 5.3)	77.1 ( $\uparrow$ 4.6)	78.4 ( $\uparrow$ 9.6)	77.8 ( $\uparrow$ 9.0)	82.3 ( $\uparrow$ 7.7)	81.8 ( $\uparrow$ 7.2)	81.3 ( $\uparrow$ 15.3)	80.7 ( $\uparrow$ 14.7)
Avg. Improvement	$\uparrow$ 6.3	$\uparrow$ 5.5	$\uparrow$ 5.4	$\uparrow$ 5.0	$\uparrow$ 5.5	$\uparrow$ 4.9	$\uparrow$ 8.1	$\uparrow$ 7.7	$\uparrow$ 6.8	$\uparrow$ 6.5	$\uparrow$ 14.2	$\uparrow$ 13.5

methods demonstrate performance gains, their effectiveness may vary across domains or with datasets beyond those examined in this study. Moreover, the observed improvements from few-shot learning may be highly dataset-dependent. Further experimentation with additional and more diverse datasets is needed to more thoroughly evaluate the robustness and scalability of these approaches.

## VII. CONCLUSION

This empirical study evaluates the performance of ten open-source lightweight LLM-based parsers with small parameter sizes across six carefully constructed datasets, representing both modern large-scale and low-resource log environments.

Our findings reveal significant variability in PA and FTA, with models achieving averages of up to 70.1% PA and 66.8% FTA on benchmark datasets such as LHSB, but dropping to 58.7% PA and 53.1% FTA on more diverse datasets like FALL. Longer logs and those containing more dynamic variables also resulted in noticeable declines in performance. Few-shot learning—particularly the example-based approach—demonstrated substantial improvements, achieving the highest gain of 14.2% PA on the most challenging dataset, FALL. These results underscore the importance of dataset diversity in benchmarking and highlight the potential of targeted few-shot learning to substantially enhance the generalization capabilities of LLMs in log parsing tasks.

## REFERENCES

- [1] X. Wu, H. Li, and F. Khomh, "On the effectiveness of log representation for log-based anomaly detection," *Empirical Software Engineering*, vol. 28, no. 6, p. 137, 2023.
- [2] Z. Li, C. Luo, T.-H. Chen, W. Shang, S. He, Q. Lin, and D. Zhang, "Did we miss something important? studying and exploring variable-aware log abstraction," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 830–842.
- [3] 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.
- [4] Y. Sun, J. Keung, J. Zhang, H. K. Yu, W. Luo, and S. Liu, "Unveiling hidden anomalies: Leveraging smac-1stm for enhanced software log analysis," in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024, pp. 1178–1183.
- [5] 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 *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 2023, pp. 392–404.
- [6] V.-H. Le and H. Zhang, "Log-based anomaly detection with deep learning: How far are we?" in *Proceedings of the 44th international conference on software engineering*, 2022, pp. 1356–1367.
- [7] Z. A. Khan, D. Shin, D. Bianculli, and L. Briand, "Guidelines for assessing the accuracy of log message template identification techniques," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1095–1106.
- [8] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and benchmarks for automated log parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 121–130.
- [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] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, "Logram: Efficient log parsing using  $n$ -gram dictionaries," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 879–892, 2020.
- [11] 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.
- [12] X. Wang, X. Zhang, L. Li, S. He, H. Zhang, Y. Liu, L. Zheng, Y. Kang, Q. Lin, Y. Dang *et al.*, "Spine: a scalable log parser with feedback guidance," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 1198–1208.
- [13] 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.
- [14] V.-H. Le and H. Zhang, "Log parsing with prompt-based few-shot learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2438–2449.
- [15] Z. Xu, B. Huang, and N. Chen, "Log parsing using semantic filtering based prompt learning," in *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2024, pp. 667–676.
- [16] Z. Ma, A. R. Chen, D. J. Kim, T.-H. P. Chen, and S. Wang, "Llmparser: An exploratory study on using large language models for log parsing," 2024.
- [17] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for ai-driven log analytics," in *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2023, pp. 355–366.
- [18] M. Landauer, F. Skopik, and M. Wurzenberger, "A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, pp. 1354–1375, 2024.
- [19] 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.
- [20] 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.
- [21] C. Almodovar, F. Sabrina, S. Karimi, and S. Azad, "Logfit: Log anomaly detection using fine-tuned language models," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 1715–1723, 2024.
- [22] A. Tufek and M. S. Aktas, "On the provenance extraction techniques from large scale log files," *Concurrency and Computation: Practice and Experience*, vol. 35, no. 15, p. e6559, 2023.
- [23] 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.
- [24] 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.
- [25] K. I. Roumeliotis and N. D. Tselikas, "Chatgpt and open-ai models: A preliminary review," *Future Internet*, vol. 15, no. 6, p. 192, 2023.
- [26] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," in *Generative AI for Effective Software Development*. Springer, 2024, pp. 71–108.
- [27] S. Gao, X.-C. Wen, C. Gao, W. Wang, H. Zhang, and M. R. Lyu, "What makes good in-context demonstrations for code intelligence tasks with llms?" in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 761–773.
- [28] J. Qi, S. Huang, Z. Luan, S. Yang, C. Fung, H. Yang, D. Qian, J. Shang, Z. Xiao, and Z. Wu, "Loggpt: Exploring chatgpt for log-based anomaly detection," in *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2023, pp. 273–280.
- [29] 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.
- [30] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, 2016, pp. 102–111.
- [31] 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.
- [32] 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.
- [33] N. Tremblay, S. Barthelmé, and P.-O. Amblard, "Determinantal point processes for coresets," *Journal of Machine Learning Research*, vol. 20, no. 168, pp. 1–70, 2019.
- [34] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.
- [35] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.
- [36] P. Xia, L. Zhang, and F. Li, "Learning similarity with cosine similarity ensemble," *Information sciences*, vol. 307, pp. 39–52, 2015.
- [37] X. Lin, W. Wang, Y. Li, S. Yang, F. Feng, Y. Wei, and T.-S. Chua, "Data-efficient fine-tuning for llm-based recommendation," in *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, 2024, pp. 365–374.
- [38] D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv preprint arXiv:2501.12948*, 2025.
- [39] G. Team, M. Riviere, S. Pathak, P. G. Sessa, C. Hardin, S. Bhupatiraju, L. Hussenot, T. Mesnard, B. Shahriari, A. Ramé *et al.*, "Gemma 2: Improving open language models at a practical size," *arXiv preprint arXiv:2408.00118*, 2024.
- [40] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

- [41] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, “Qwen2. 5 technical report,” *arXiv preprint arXiv:2412.15115*, 2024.
- [42] Y. Zhang, K. Zhou, and Z. Liu, “What makes good examples for visual in-context learning?” *Advances in Neural Information Processing Systems*, vol. 36, pp. 17 773–17 794, 2023.
- [43] S. K. Singh, S. Kumar, and P. S. Mehra, “Chat gpt & google bard ai: A review,” in *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*. IEEE, 2023, pp. 1–6.
- [44] A. Kulesza and B. Taskar, “k-dpps: Fixed-size determinantal point processes,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1193–1200.
- [45] L. E. Peterson, “K-nearest neighbor,” *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.