Why Transformers? A Comprehensive Overview of Transformers in Artificial Intelligence for IT Operations

BINPENG SHI, Nankai University, China
SHENGLIN ZHANG*, Nankai University, Tianjin, China and Haihe Laboratory of Information Technology Application Innovation (HL-IT), Tianjin, China
JINGYA WANG, BOWEN HAO, MINYI SHAO, and YU LUO, Nankai University, Tianjin, China
WENWEI GU, Nankai University, Tianjin, China
YONGQIAN SUN, Nankai University, Tianjin, China and Tianjin Key Laboratory of Software Experience and Human Computer Interaction (TKL-SEHCI), Tianjin, China
SIBO XIA and YONGXIN ZHAO, Nankai University, China
DAN PEI, Tsinghua University, Beijing, China

AIOps (Artificial Intelligence for IT Operations) has emerged as a key strategy for operating large-scale systems. Meanwhile, the Transformer architecture, renowned for its strong performance and potential, is driving advances in AIOps. However, prior reviews leave gaps in explaining why Transformers trigger a major shift in academia and industry, with limited coverage of multimodal data (logs, metrics, events), insufficient analysis of structural and methodological connections between Transformers and AIOps, fragmented views of capabilities, and a lack of consolidated evaluation strategies. To bridge these gaps, our work decomposes "Why Transformers" into three questions. First, we review what roles Transformers assume in AIOps across 70 representative papers, using a structured taxonomy that proceeds sequentially from data to targets, principles, and approaches, which present an evolutionary trajectory of role changes. Then, to explain why Transformers excel, we introduce a capability framework comprising two basic and four advanced capabilities. As for how to evaluate Transformers in AIOps, we consolidate evaluation resources and strategies, including offline datasets, standardized questions, and real-time simulations. Furthermore, probable best practices are distilled, offering actionable guidance for real-world adoption. These explorations aim to clarify the internal rationale and unlock the practical value of Transformers in AIOps.

CCS Concepts: • **Software and its engineering → Maintaining software**; • **Computing methodologies → Artificial intelligence**; • **General and reference → Surveys and overviews**.

Additional Key Words and Phrases: AIOps, transformer, large language model, logs, time series, events, anomaly detection, root cause analysis, failure prediction

1 Introduction

"We do not have knowledge of a thing until we have grasped its why, that is to say, its cause." — Aristotle

*Shenglin Zhang is the corresponding author.

Authors' Contact Information: Binpeng Shi, shibinpeng23@mail.nankai.edu.cn, Nankai University, Tianjin, China; Shenglin Zhang, zhangsl@nankai.edu.cn, Nankai University, Tianjin, China and Haihe Laboratory of Information Technology Application Innovation (HL-IT), Tianjin, China; Jingya Wang, 2111189@mail.nankai.edu.cn; Bowen Hao, haobw@mail.nankai.edu.cn; Minyi Shao, 2120230754@mail.nankai.edu.cn; Yu Luo, luoyu@mail.nankai.edu.cn, Nankai University, Tianjin, China; Wenwei Gu, wwgu@nankai.edu.cn, Nankai University, Tianjin, China; Yongqian Sun, sunyongqian@nankai.edu.cn, Nankai University, Tianjin, China and Tianjin Key Laboratory of Software Experience and Human Computer Interaction (TKL-SEHCI), Tianjin, China; Sibo Xia, xiath@mail.nankai.edu.cn; Yongxin Zhao, zyx_nkcs@mail.nankai.edu.cn, Nankai University, Tianjin, China; Dan Pei, peidan@tsinghua.edu.cn, Tsinghua University, Beijing, China.

Software reliability is crucial for both corporate profitability and reputation. Modern IT operations, faced with increasingly complex systems and services, have turned to Artificial Intelligence for IT Operations (AIOps) as a key strategy to enhance quality and efficiency. Specifically, AIOps employs artificial intelligence techniques, including machine learning and deep learning, to extract features, construct models, detect anomalies, identify failures, and provide decision support based on vast amounts of observable data (*e.g.*, metrics, logs, events) [1–3]. Compared to traditional operations techniques based on rules and scripts, AIOps demonstrates greater adaptability and generalization, enabling it to handle dynamic environments and intricate failure scenarios. Building on these advantages, AIOps has been incorporated into various software systems, from enterprise information platforms to large-scale online services. Particularly in cloud computing [4–9], microservice [2, 10–14], and other distributed environments, it supports the operations pipeline, covering system monitoring, anomaly detection, failure diagnosis, and performance optimization, thereby enhancing the overall reliability of software systems.

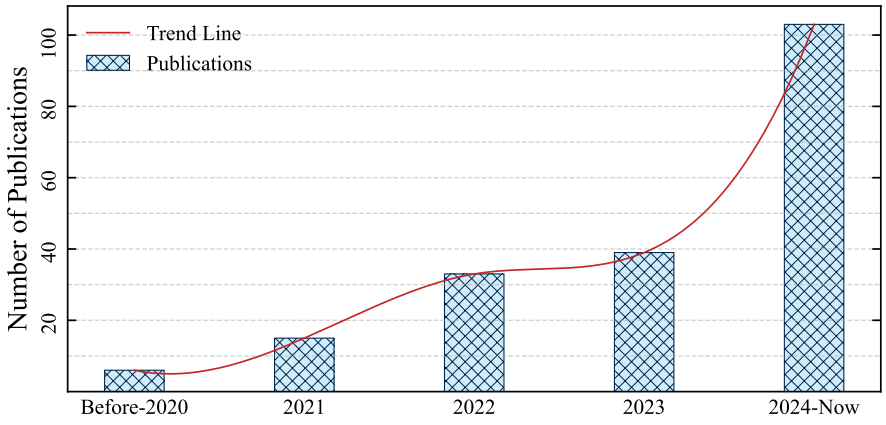

Fig. 1. Related publication trends: Transformers in AIOps.

How, then, can we design an effective model for AIOps? Although traditional machine learning-based techniques have achieved some success, they still perform inadequately when handling complex systems and heterogeneous data sources. Deep learning-based techniques have been widely adopted because of their superior ability to model observable data. From the perspective of deep learning model design, previous research has primarily relied on CNN and RNN architectures [15–18]. CNNs are typically suited for processing spatial local features, but the convolutional kernel size limits their receptive field [19, 20]. RNNs can only capture sequential and contextual relationships in a single direction. Although Bidirectional RNNs can model contextual features bidirectionally, they still cannot resolve the gradient vanishing or exploding problems caused by cyclic dependencies [21, 22]. Moreover, these networks are difficult to parallelize efficiently [23, 24]. Researchers introduced the Transformer architecture to address these issues, establishing it as a new foundational model following CNNs and RNNs [25–27]. Transformer's self-attention mechanism enables it to capture global contextual relationships within sequences, mitigating the sensitivity to sequence length seen in traditional models. Unlike RNNs, the Transformer's parallel processing capability enables it to handle large-scale data more efficiently, significantly reducing computational costs. Empirical evidence supports these theoretical advantages. Recent studies compare Transformer-based models with traditional architectures (*e.g.*, CNNs and RNNs) across multiple

Table 1. State-of-the-art surveys conducted in the fields covered by this work, based on the data modality (log, metric, event), the Transformer scope (Vanilla TF represents plain Transformer architectures, Pre-Trained TF represents large pre-trained models), the application perspective (structural principles and methodological approaches), the capability framework, the related evaluation (publicly available resources such as datasets and metrics, and evaluation strategies). ✓ˣ indicates that the work touches on the relevant aspect, but fails to clearly articulate or elevate it to the level of a taxonomy.

Reference	Year	Data Modality			Transformer Scope		Application		Capability	Evaluation	
		Log	Metric	Event	Vanilla TF	Pre-Trained TF	Principle	Approach		Resource	Strategy
Li et al. [13]	2022										
Wen et al. [34]	2022		✓		✓						✓ˣ
Ahmed et al. [35]	2023			✓		✓		✓ˣ			✓ˣ
Le et al. [36]	2023	✓				✓		✓ˣ			✓ˣ
Remil et al. [37]	2024			✓						✓	
Su et al. [38]	2024	✓ˣ	✓	✓ˣ		✓		✓		✓	
Zhang et al. [2]	2024	✓	✓							✓	
Zhang et al. [3]	2025	✓ˣ	✓ˣ	✓ˣ		✓		✓		✓	✓
Our work	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

AIOps tasks, including time-series forecasting, anomaly detection, and log analysis. These studies consistently show competitive or superior performance in terms of accuracy, efficiency, robustness, and adaptability to heterogeneous data [28–33]. Moreover, we conduct a systematic review and identify 196 relevant studies (Sec. 2.1), based on which we construct the yearly publication trend shown in Fig. 1. The trend indicates that Transformer-based AIOps techniques have attracted increasing research attention, with a notable surge following the emergence of ChatGPT. This rapid growth highlights the potential of Transformers in the field of AIOps.

Why has the Transformer architecture caused a seismic shift in both academia and industry? To date, numerous studies have reviewed the techniques in the field of AIOps, with some specifically concentrating on Transformers or Large Language Models (LLMs), as summarized in Tab. 1. However, these studies expose several critical gaps: **(1) Insufficient exploration of AIOps, Transformers, and their interconnections:** Typically, AIOps systems rely on heterogeneous data sources such as logs, metrics, and events. Comprehensive analysis of these data modalities is essential for effective failure detection, triage, localization, and mitigation. However, most surveys either limit their scope to a single or incomplete modality [2, 13, 34–37], or fail to clearly map reviewed techniques to the specific data modalities and task objectives [3, 38]. This results in an incomplete understanding for researchers of how data characteristics should guide the selection of appropriate analytical methods. In addition, existing studies tend to focus on either standard Transformer models [34] or large pre-trained models [3, 35, 36, 38], with little systematic exploration into the broader space of Transformer forms. More importantly, they often fail to clarify the principled alignment between Transformers and AIOps tasks [2, 3, 13, 34–38] (*e.g.*, how specific architectures or variants are suited to telemetry data analysis) and to assess the strengths and weaknesses of different approaches [2, 13, 34–37] (*e.g.*, architecture-specific models, fine-tuning, prompt engineering, and large pre-trained models). This deficiency not only limits practical guidance for researchers and practitioners but also obscures the core contributions of Transformers in AIOps. **(2) Notable absence of a systematic capability framework for Transformers in AIOps:** Although Transformers have been widely adopted in various AIOps tasks, previous surveys [2, 3, 13, 34–38] still lack a conceptual framework that systematically characterizes their capabilities. In particular, explanations of why Transformers are inherently advantageous for AIOps are often absent or restricted to superficial descriptions, such as generalized claims of “better performance” or “effectiveness in sequence modeling”. This fragmented view impedes the development of a unified and in-depth understanding, thereby hindering

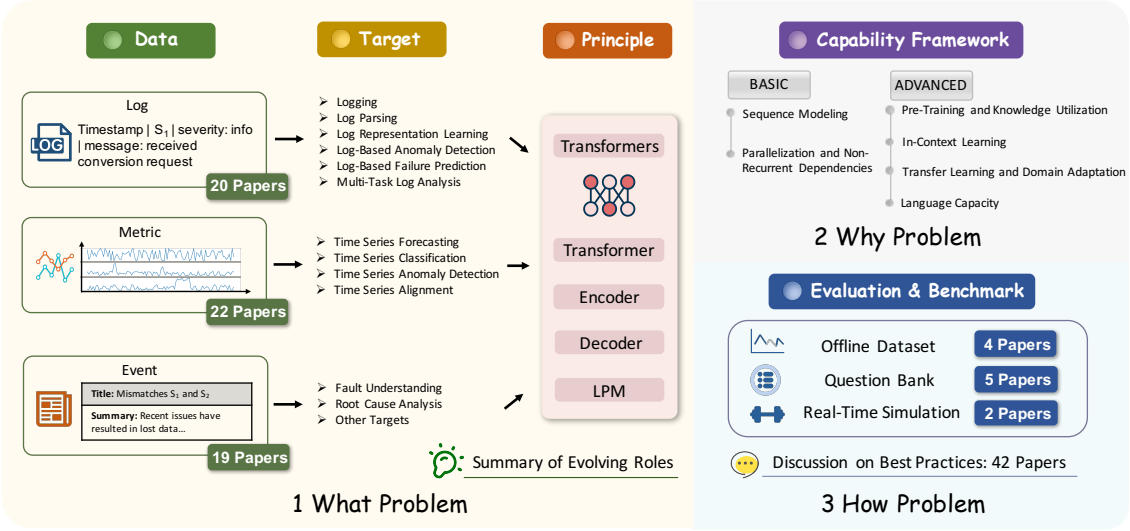

Fig. 2. The overall framework of research questions for Transformers in AIOps.

the full realization of Transformers' potential value. **(3) Inadequate integration of publicly available resources and evaluation strategies:** The progress of AIOps research relies heavily on access to high-quality public resources, including datasets, benchmarks, and evaluation metrics. But existing reviews [13, 34–36] often provide incomplete coverage, narrow perspectives, and non-standardized integration. Moreover, they [2, 13, 34–38] rarely offer in-depth analysis of evaluation strategies, such as how to leverage public resources for assessing related techniques. These limitations undermine research reproducibility, comparative assessments, and continuous improvement, while also creating significant obstacles to the broader adoption of AIOps techniques in both academic and industrial settings.

To bridge these gaps, we break down the problem of "Why Transformers" into three specific research questions. As shown in Fig. 2, our work first explores ***What roles Transformers assume in AIOps (Gap 1)***. We provide an extensive survey of Transformers in AIOps, examining and categorizing these techniques based on data sources, task objectives, employed Transformer architectures or variants, and core methodologies. The summary of role changes also presents an evolutionary trajectory in this context. Furthermore, we investigate the capabilities and benefits that Transformers bring to these techniques, highlighting two basic capabilities: Sequence Modeling (SEQ) and Parallelization and Non-Recurrent Dependencies (PARALLEL), as well as four advanced capabilities: Pre-Training and Knowledge Utilization (PTKL), In-Context Learning (ICL), Transfer Learning and Domain Adaptation (TF), and Language Capacity (LANG). We aim to uncover the rationale behind ***Why Transformers have become the preferred choice (Gap 2)***. Additionally, we summarize existing benchmarks related to the field of IT operations. These benchmarks revolve around constructing datasets, designing test suites, and conducting evaluations for large pre-trained models, addressing the critical question of ***How to assess Transformers in AIOps (Gap 3)***.

Contributions. To recapitulate, we highlight the following contributions:

- ***What Problem.*** We conduct a systematic review of recent advances in applying Transformers to AIOps, broadening the scope to consider both vanilla architectures and pre-trained models jointly. Building on this, we propose a structured taxonomy that categorizes reviewed techniques by data modality (logs in Sec. 5.1, metrics in

Sec. 5.2, and events in Sec. 5.3), and further organizes them within each modality according to task objectives and structural principles. The details, including approaches, are summarized in Tab. 5, 6, and 7. Subsequent analysis highlights the interconnections between Transformer and AIOps across various perspectives, including data nature, task characteristics, architectural paradigms, and methodological approaches (Sec. 5.1.7, 5.2.5, and 5.3.4). From these insights, we summarize the role changes of Transformers in AIOps to underscore the overarching evolution, which may spark inspiration for researchers (Sec. 8).

- ***Why Problem.*** We present a two-level capability framework (Sec. 4.2), comprising two basic capabilities (Sequence Modeling / SEQ, and Parallelization and Non-Recurrent Dependencies / PARALLEL) and four advanced capabilities (Pre-Training and Knowledge Utilization / PTKL, In-Context Learning / ICL, Transfer Learning and Domain Adaptation / TF, and Language Capacity / LANG). Leveraging this framework, we elaborate on the distinctive strengths of Transformers in AIOps to support a comprehensive and in-depth understanding (recorded in Tab. 5, 6, 7 and discussed in Sec. 5.1.7, 5.2.5, 5.3.4). This enables value realization in analyzing diverse data sources and addressing specific tasks. To the best of our knowledge, we are among the first to clarify a capability framework for Transformers within the AIOps domain.
- ***How Problem.*** We provide a detailed integration of existing evaluations and benchmarks, together with the adopted metrics (Sec. 6). In addition, we extend the discussion by further identifying open challenges and outlining future directions, and then proposing corresponding best practices across different scenarios to guide on-call engineers toward the most appropriate solution (Sec. 8). Collectively, our work supports researchers in conducting model assessment, comparison, and selection, offering practical guidance for the continuous optimization of Transformers in AIOps.

Roadmap. The remainder of this paper is organized as follows. Sec. 2 introduces the survey methodology, presents the research framework, and discusses threats to validity. Sec. 3 outlines the evolution of AIOps and the background of Transformers. Sec. 4 proposes a systematic taxonomy, delving into the role of Transformers in AIOps from the perspectives of data sources, task objectives, structural principles and methodological approaches, and defines a two-level capability framework for Transformers in AIOps. It also presents the design rationale and comparative analysis of the proposed taxonomy. Sec. 5 conducts a detailed review of existing Transformer-based AIOps studies, classified by the three data modalities of logs, metrics, and events, and further organized by task objectives and structural principles. Sec. 6 summarizes commonly used benchmarks, datasets, metrics, and evaluation strategies. Sec. 7 provides a comparative analysis of related survey studies. Sec. 8 discusses evolving roles, open challenges, future directions, and best current practices. Sec. 9 concludes the paper.

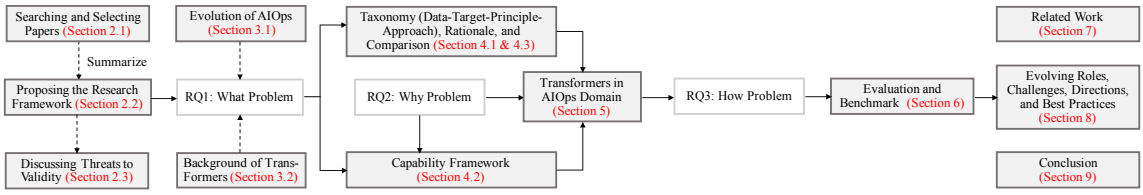

Fig. 3. Overview of topics covered in this article.

Table 2. Quality assessment criteria (QAC)

ID	Criterion	Description
QAC1	Recency of Publication	Is the publication within recent years to ensure the relevance of the findings to the current development of Transformer-based models and AIOps research?
QAC2	Venue Credibility	Is the publication venue ranked among top-tier conferences or journals in artificial intelligence, machine learning, or AIOps, ensuring the credibility and impact of the work?
QAC3	Research Impact	Has the paper received a significant number of citations or been referenced in influential works, indicating its impact and recognition within the research community?
QAC4	Methodological Innovation	Does the paper introduce innovative Transformer-based methodologies, architectures, or applications in time series analysis, log analysis, or incident management, advancing the state-of-the-art in AIOps?
QAC5	Clarity of Presentation	Is the paper well structured and clearly written, presenting its research objectives, methodology, experimental results, and conclusions effectively?
QAC6	Experimental Rigor	Does the paper provide a comprehensive evaluation of the proposed Transformer-based approach, including detailed experimental design, datasets, baselines, and reproducibility information?

2 Methodology

2.1 Search and Selection Process

To systematically collect papers for conducting this survey, we follow the review methodology in [2, 3, 39, 40] to obtain a repository for Transformers in AIOps, consisting of the following four steps.

Search Strategy. Our initial step is to investigate the top peer-reviewed and influential conferences and journals in the domains of artificial intelligence, software engineering, data mining, *etc.* , which include 26 conferences (*i.e.*, AAAI, ASE, CIKM, EuroSys, FSE, HotNets, ICDE, ICLR, ICML, ICS, ICSE, ICWS, IJCAI, IJCNN, INFOCOM, ISSRE, ISSTA, IWQoS, NSDI, NeurIPS, OSDI, SANER, SIGCOMM, SIGKDD, VLDB, and WWW) and 9 journals (*i.e.*, TC, TDSC, TIST, TKDE, TON, TOSEM, TPDS, TSC, and TSE). Subsequently, we manually screened and retrieved 43 papers relevant to our research topic, beginning in 2017. That year, the Transformer was introduced [25] and has since gradually gained popularity in the AIOps domain.

Additionally, we conduct an automated search to systematically retrieve papers from well-known digital databases, including IEEE Xplore [41], ACM Digital Library [42], SpringerLink [43], Web of Science [44], ScienceDirect [45], and arXiv [46]. These repositories encompass a vast collection of papers in the field of computer science, offering robust search functions and abundant open-access information. To retrieve relevant papers from the mentioned repositories, we primarily utilize a predefined set of search keywords, derived from papers identified through our conference and journal review. By applying these keywords, we obtain a total of 1,259 papers spanning from 2017 to the present.

Selection Criteria. After collecting the papers, we perform a relevance selection based on the pre-defined inclusion and exclusion criteria.

- ✓ *The paper must be written in English.*
- ✓ *The paper must have an accessible full text.*
- ✓ *The paper must be a peer-reviewed full research paper published either in a conference or a journal* [1].
- ✓ *The paper must adopt Transformer-based models to solve tasks in the AIOps domain, such as log analysis.*
- ✗ *The paper does not have more than 5 pages.*
- ✗ *The paper is a literature review or survey.*
- ✗ *Duplicate papers or similar studies by the same authors.*

[1]For preprint papers released on arXiv in the past three years that are not yet formally published, as well as public resources such as open-source benchmarks, we retain them when checking venues and perform a manual quality check to decide whether to include them in our study.

✗ *The paper does not utilize the Transformer architecture or its variants.*

✗ *The paper is not involved in the domain of AIOps.*

Initially, the number of included papers is reduced to 493 after filtering out short papers (exclusion criterion 1), literature reviews (exclusion criterion 2), and duplicates (exclusion criterion 3). Subsequently, we manually review the titles, authors, and abstracts of the remaining papers (exclusion criteria 4-5). As a result, we identify and select 196 papers that are directly relevant to Transformers in the AIOps domain. Fig. 1 presents the trend of annual papers.

Quality Assessment. To identify representative studies, we define six quality assessment criteria (QAC), as summarized in Tab. 2, to evaluate the effectiveness and significance of the collected papers. Each criterion is scored on a four-point scale ranging from 0 to 3 (0 = poor, 1 = fair, 2 = good, and 3 = excellent). Specifically, QAC1 (Recency of Publication) is determined based on the publication year, with more recent studies receiving higher scores. QAC2 (Venue Credibility) and QAC3 (Research Impact) are assessed based on the reputation of the publication venue and the influence of the study within the research community, respectively. QAC4–QAC6 evaluate methodological innovation, clarity of presentation, and rigor of experimental evaluation through qualitative assessment. The collected papers are initially assigned to four master's students (each with approximately two years of research experience), who evaluate their assigned subsets according to their domain expertise in log, metric, event, and multimodal analysis. This approach ensures that each study is reviewed by a researcher familiar with the corresponding topic. The assessment results are subsequently reviewed by a PhD student with four years of research experience to ensure scoring consistency. Any discrepancies are resolved through discussion among the co-authors until consensus is reached. Disagreements are relatively limited and primarily involve borderline cases, suggesting a generally consistent assessment process. After the quality assessment, 57 papers are retained with total scores exceeding 14 (*i.e.*, 80% of the maximum possible score).

Forward and Backward Snowballing. To ensure comprehensive coverage and minimize the risk of overlooking representative studies, we conduct both backward and forward snowballing. This process involves examining the references cited in our initially selected papers as well as identifying subsequent publications that reference these studies. As a supplementary step, we incorporate additional relevant papers and reapply our selection criteria, including filtering, deduplication, and quality assessment. Ultimately, this iterative process results in a final set of 70 papers for our further analysis.

To enhance transparency and reproducibility, we provide a publicly available replication package containing the selected papers and associated information, including search keywords, selection criteria, quality assessment criteria, and scoring results. The replication package is available at: https://anonymous.4open.science/r/survey_repository.

2.2 Research Framework

We propose a comprehensive framework for Transformers in AIOps, structured around three key research questions, as presented in Fig. 2.

RQ1: *What roles do Transformers assume in AIOps, and in what ways are they applied to enhance IT operations?*
RQ2: *Why have Transformers become widely adopted in AIOps, and what key capabilities and benefits underpin their utility in optimizing IT operations?*
RQ3: *How can the effectiveness of Transformers in AIOps be rigorously evaluated and systematically assessed?*

On the left side of Fig. 2, we categorize the collection of papers into three groups based on data sources. Specifically, logs capture runtime behavior as semi-structured text. Metrics monitor the system status through the values and changes. Events summarize high-level incident-related information such as impact scope, root causes, and mitigation steps in natural language. Through the adoption of specialized Transformer variants for distinct data modalities, the surveyed papers demonstrate enhancement across multiple AIOps tasks. In Sec. 5, we organize the content around data modalities, task objectives, and structural principles, while expanding on methodological approaches and core techniques. Additionally, the summary of evolving roles in Sec. 8 also provides an overarching perspective. This comprehensive analysis highlights the roles of Transformers in AIOps, in response to *RQ1*.

As shown on the upper right side of Fig. 2, we address *RQ2* by establishing a two-level capability system for Transformers in AIOps, informed by both the surveyed papers and practical insights [25, 47–49]. The fundamental capabilities and benefits consist of *Sequence Modeling* and *Parallelization and Non-Recurrent Dependencies*, which are brought by the core characteristic of Transformer architecture, particularly the multi-head self-attention mechanism. The advanced capabilities and benefits are derived from large pre-trained Transformers, including *Pre-Training and Knowledge Utilization*, *In-Context Learning*, *Transfer Learning and Domain Adaptation*, and *Language Capacity*. Detailed definitions and elaborations on these capabilities can be found in Sec. 4.2. Building on the two-level system, we further analyze the Transformer capabilities manifested in the AIOps techniques, as recorded in Tab. 5, 6, 7 and discussed in Sec. 5.1.7, 5.2.5, 5.3.4, respectively.

Regarding *RQ3*, we illustrate the common evaluation approaches on the lower right of Fig. 2, including offline data analysis, standardized question banks (question answering, multiple-choice), and real-time simulation. The surveyed papers utilize these approaches to conduct evaluations and benchmark tests, thereby assessing the effectiveness of Transformers in AIOps, with a particular emphasis on large pre-trained models. Sec. 6 provides a comprehensive summary of benchmark construction processes, dataset scales, covered domains, common metrics, evaluation strategies, open-source toolkits, benchmarking perspectives, preliminary conclusions, *etc.* Furthermore, Sec. 8 extends the discussion to open challenges, future directions, and probable best practices.

By responding to the three key questions in our research framework, we aim to provide a comprehensive understanding of Transformers in AIOps, exploring "what" they are, "why" they matter, and "how" their effectiveness can be quantitatively benchmarked. Ultimately, this will advance AIOps techniques and push the boundaries of Transformer applications in this field.

2.3 Threats to Validity

Despite our efforts to conduct a comprehensive survey, several threats to validity remain. (1) Internal validity may be affected by potential search and selection biases. Some relevant studies might still be missed due to keyword limitations or indexing differences. In addition, the classification of techniques within our taxonomy involves a certain degree of subjective judgment. To mitigate these risks, we combine manual screening of top venues, automated searches across multiple digital libraries, detailed paper assessment and selection, and backward snowballing. Furthermore, the survey results are reviewed by multiple authors, and disagreements are resolved through discussion. (2) External validity concerns the generalizability of our findings. First, publicly available literature may not fully capture proprietary industrial practices. However, many reviewed studies [5, 30, 35, 50–66] involve collaborations between academia and industry, partially mitigating this limitation. Second, the application of Transformers to trace analysis remains relatively underexplored in existing research, which may limit the generalization of our observations. We therefore discuss the absence of trace-related studies and potential directions in Sec. 8.4.

3 Preliminary

3.1 Evolution of AIOps

Operation and maintenance (O&M) is the most enduring phase in the lifecycle of a software system. Traditional O&M approaches rely on manual rules or automated scripts. However, these approaches are overwhelmed by the increasing complexity of systems and heterogeneous data. Modern O&M is gradually shifting to intelligent O&M due to its effectiveness and efficiency. Specifically, AIOps applies artificial intelligence techniques, including traditional machine learning methods (*e.g.*, decision trees, SVM), deep learning methods (*e.g.*, VAE, LSTM, CNN), reinforcement learning methods (*e.g.*, MCTS), knowledge graph methods, graph computation methods, *etc.* , to analyze application-layer and machine-layer data within internet services or web services (*e.g.*, search engines, online shopping, online video, instant messaging) [2, 3, 67]. By leveraging extensive multimodal data, operators can perform anomaly detection, root cause analysis, and fault prediction, thereby ensuring service reliability and enhancing user experience [4].

To effectively support the application of AIOps techniques, data sources must provide a comprehensive, in-depth, and easily interpretable description of the system status at runtime. Logs, metrics, and events are recognized as the key dimensions of observability, which are commonly utilized in the AIOps domain [3, 68]. Typically, logs capture system runtime behavior as semi-structured text, containing information such as operations, warnings, and errors [69]. Metrics are multidimensional time series used to monitor hardware, systems, and services, such as throughput, CPU utilization, and memory usage. Observing changes in these variables over time enables the visual measurement of system performance, the identification of abnormal behavior, and the discovery of potential issues [70]. Events, termed incidents from the cloud side or tickets from the customer side, constitute another important data source, recorded as natural language text throughout the lifecycle of a failure [3, 10]. These records include fields like the associated service, event source, timestamp, and problem descriptions. They can be generated automatically using predefined templates or manually by engineers. Continuously collect and analyze the above multimodal data, and we can describe the system status in a comprehensive and multi-dimensional manner, fully leveraging AIOps techniques to ensure system stability and reliability. In addition, the three pillars of observability are typically logs, metrics, and traces [68]. However, this paper does not include a dedicated section on Transformers for trace analysis, mainly because there is very limited research applying Transformers to traces. Therefore, we only discuss trace analysis together with multimodal integration in Sec. 8.4 as a supplementary topic.

Notably, as large pre-trained models demonstrate impressive performance in CV and NLP, more and more researchers are investigating their potential in the AIOps domain to tackle challenges that traditional machine learning or deep learning techniques have struggled to resolve, including multi-task and multi-scenario generalization, interactive O&M experiences, knowledge deposition and utilization, continuous learning and updating, among others.

3.2 Background of Transformers

Transformer is a Seq2Seq model centered on the self-attention mechanism and consists of two parts: the encoder and the decoder, both of which are composed of multiple stacked layers with the same structure [25]. Each encoder layer contains two sublayers: the Multi-Head Self-Attention Layer and the Feed-Forward Neural Network Layer. Each decoder layer contains three sublayers: the Masked Multi-Head Self-Attention Layer, the Encoder-Decoder Attention Layer, and the Feed-Forward Neural Network Layer. Through the Self-Attention mechanism, the Transformer can capture long-range dependencies while simultaneously allowing parallel training to enhance both training and inference efficiency. Specifically, the Self-Attention mechanism projects elements of the input sequence into three vectors: Q

(Query), K (Key), and V (Value), computes the similarity between Q and K to produce the attention matrix, and then sums the vectors V weighted by the attention coefficients to obtain the vector representation. Multiple attention heads allow the Transformer to simultaneously focus on diverse features across various subspaces, enabling the model to capture data patterns more comprehensively. The Self-Attention mechanism is formulated as:

$$\begin{aligned} Attention(Q,K,V) &= Softmax(\frac{QK^T}{\sqrt{d_k}})V \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \\ MultiHead &= Concat(head_1, ..., head_h)W^O \end{aligned} \tag{1}$$

where the projections are parameter matrices $W_i^Q, W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. Additionally, the feed-forward network consists of two linear transformations separated by a ReLU activation. Residual connection and layer normalization are added after each sub-layer to alleviate the gradient vanishing problem and accelerate model training.

Transformer-based models, *i.e.*, Transformers, can be categorized into four structural principles (referred to as "Principle" in this manuscript): (1) Transformer-Encoder-Based structures generally focus on encoding sequence data; (2) Transformer-Decoder-Based structures are ideal for sequence generation tasks; (3) A Transformer-Based structure integrates an encoder and a decoder, dedicated to sequence transformation tasks. (4) Large Pre-Trained Models emphasize learning from extensive amounts of unlabeled data, showing significant generalization capabilities.

Moreover, to fully unlock the potential of Transformers in meeting diverse demands, several methodological "Approaches" have been developed: (1) Transformer-Architecture Based. This approach typically utilizes the vanilla Transformer or modifies specific modules such as the self-attention mechanism, to better accommodate particular scenarios. (2) Prompt Engineering. This approach focuses on designing appropriate prompts, which guide the foundation model in completing tasks without gradient updates. Some techniques utilize tailored prompts for fine-tuning, allowing greater flexibility and adaptability. (3) Fine-Tuning. Compared to prompt engineering, this approach allows for adjusting the model's parameters to better align with specific scenarios or tasks. However, it necessitates high-quality fine-tuning datasets and significant computational resources. (4) Foundation Model. This approach involves training a foundational model on extensive datasets (usually through large-scale self-supervised learning combined with post-tuning such as supervised fine-tuning, reinforcement learning, *etc.*), which can then be adapted to a wide range of downstream tasks using either prompt engineering or fine-tuning. In this survey, we also categorize techniques leveraging language models like BERT for semantic embedding under this group, given their domain-specific capabilities.

4 Taxonomy

For the collected papers in this study, we present a comprehensive framework to systematically characterize Transformers, including their data sources, task objectives, architectural variants, and core methodologies (Sec. 4.1). We then define the capabilities of Transformers, comprising two basic capabilities and four advanced capabilities (Sec. 4.2). We further discuss the design rationale of the taxonomy and compare it with existing taxonomies (Sec. 4.3).

4.1 Characteristic

As detailed in Sec. 3.1, we classify the surveyed papers into three main categories based on the data modalities they analyze: ***Logs, Metrics, and Events (Data)***. Within each modality, relevant techniques are subsequently grouped

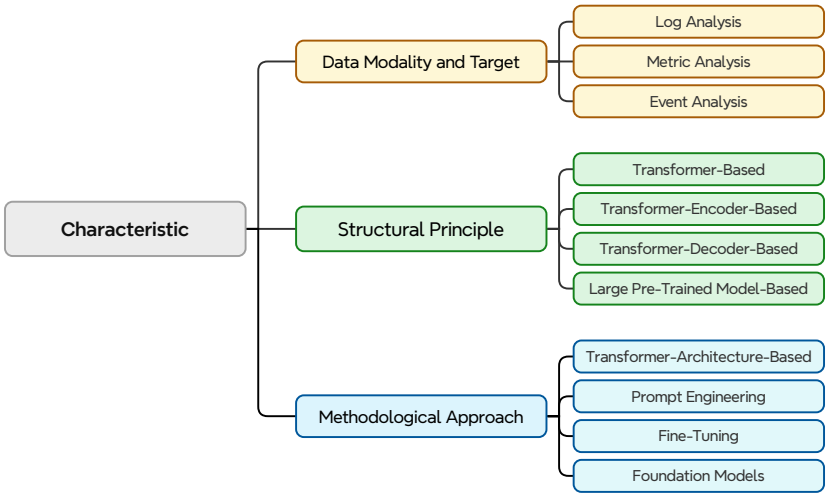

Fig. 4. Characteristics of Transformers in AIOps.

according to their task objectives, such as log parsing, time series forecasting, and fault analysis. This paper explores the application of Transformers across these three data modalities, covering 12 common AIOps tasks ***(Target)***. As illustrated in Fig. 4, we further identify the techniques by their principles and approaches to highlight the intrinsic connections between Transformers and AIOps in practical applications. Specifically, from a model architecture perspective ***(Principle)***, the surveyed papers cover four groups: ***Transformer-Based, Transformer-Encoder-Based, Transformer-Decoder-Based, and Large Pre-Trained Model-Based***. From an application method perspective ***(Approach)***, they are classified into four types: ***Transformer-Architecture-Based, Prompt Engineering, Fine-Tuning, and Foundation Models***. The definitions of principles and approaches are detailed in Sec. 3.2.

Tab. 5, 6, and 7 summarize the above characteristics, as well as the core methods that make up relevant techniques.

4.2 Capabilities or Benefits

This section presents the capability framework derived through a qualitative analysis. Specifically, we analyze the 61 model-design-oriented papers among the 70 surveyed papers (61 discussed in Sec. 5, 11 evaluation-oriented papers in Sec. 6, with 2 papers overlapping) and observe two main sources of Transformer advantages: intrinsic architectural mechanisms (*e.g.*, multi-head self-attention) and the pre-training paradigm, which naturally correspond to the basic and advanced levels of the capability framework. Based on the two sources, we iteratively identify a set of recurring keywords, as shown in the Keywords column of Tab. 3. These patterns are further abstracted into six capabilities: two basic ones (***Sequence Modeling, SEQ***; ***Parallelization and Non-Recurrent Dependencies, PARALLEL***) and four advanced ones (***Pre-Training and Knowledge Utilization, PTKL***; ***In-Context Learning, ICL***; ***Transfer Learning and Domain Adaptation, TF***; ***Language Capacity, LANG***).

To provide empirical grounding for the capability framework, we examine how the six capabilities are reflected in the reviewed studies, as summarized in the Capabilities or Benefits column of Tab. 5, 6, and 7. The results show that SEQ and PARALLEL appear in 37.7% and 29.5% of the papers, while PTKL, ICL, TF, and LANG appear in 78.7%, 60.7%, 31.1%, and 57.4%, respectively, indicating their broad presence in the literature and supporting the validity of the

Table 3. The two-level system of Transformers encompassing six capabilities or benefits

Level	Capabilities or Benefits	Abbreviation	Keywords
Basic	Sequence Modeling	SEQ	Multi-Head Self-Attention, Sequential Features, Long-Range Dependencies
	Parallelization and Non-Recurrent Dependencies	PARALLEL	Multi-Head Self-Attention, Parallelization, Non-Recurrent Dependencies
Advanced	Pre-Training and Knowledge Utilization	PTKL	Large-Scale Corpora, Pre-Trained Models, Parametric Knowledge, Generalizability and Model Adaptability
	In-Context Learning	ICL	Prompt, Zero-Shot, One-Shot, Few-Shot, No Gradient Updates, Multi-Tasks, CoT, Emergence
	Transfer Learning and Domain Adaptation	TF	Transfer Learning, Fine-Tuning, Specific Domains and Tasks, FFT, PEFT, RL, Label-Less
	Language Capacity	LANG	Language Generation, Conditional Text Generation, Semantic Understanding, Code Synthesis

framework. Additionally, the framework is orthogonal to the Data–Target–Principle-Approach taxonomy introduced in Sec. 4.1. While the taxonomy organizes the literature by application characteristics, the capability framework highlights the underlying sources of Transformer advantages, thereby addressing the "Why" problem.

Basic-Level Capabilities. On the one hand, basic-level capabilities primarily stem from the multi-head self-attention mechanism, which is the core of the Transformer.

Based on this mechanism, the Transformer can capture dependencies between different positions within a sequence, thereby enabling effective sequence modeling (***SEQ***). A key challenge in sequence modeling involves addressing long-range dependencies. One critical factor is the length of the paths that forward and backward signals have to traverse in the network. The shorter these paths between different positions, the easier it is to learn long-range dependencies [71]. The self-attention mechanism used by Transformers does not rely on window size or path length, allowing direct interaction between positions in the sequence. This facilitates the learning of long-range dependencies and demonstrates superior performance.

Apart from delivering superior performance, Transformers achieve more efficient sequence modeling by mitigating recurrent dependencies and enabling higher parallelization (***PARALLEL***). Specifically, RNNs must sequentially compute hidden states, with each time step's hidden state depending on the previous one, which hinders parallel computation. In addition, this sequential dependency necessitates storing hidden states, consuming substantial memory, and reducing batch processing capacity. These factors significantly impact the training efficiency and inference speed of RNNs. Furthermore, due to recurrent dependencies, RNNs may experience vanishing or exploding gradients during backpropagation, which degrades performance, particularly with long sequences [21–24]. In contrast, the multi-head self-attention mechanism allows computations at each position to be independent. Consequently, Transformers can compute hidden representations in parallel for all input and output positions. Without recursive loops, models become more efficient. Experiments on two machine translation tasks show that Transformers excel in quality, while being more parallelizable and requiring much less training time [25].

Advanced-Level Capabilities. On the other hand, advanced-level capabilities pertain to the description of large pre-trained models.

Pre-training forms the foundation of various capabilities of large pre-trained models (LPMs). Learning from large-scale corpora, LPMs can capture the underlying patterns and latent correlations, internalizing them as generalized knowledge (***PTKL***) [72, 73]. The knowledge encompasses the laws of linguistic phenomena, semantic information,

logical reasoning, text generation, and so on, which endows LPMs with significant potential to solve multiple tasks across diverse domains. The scale and quality of the corpora are critical to the success of pre-training. Additionally, model architecture, training acceleration methods, and optimization techniques also deserve attention [73].

Harnessing the potential of foundational models often involves prompt engineering. Typically, a formatted natural language prompt includes a task description, examples, and inputs. In zero-shot, one-shot, or few-shot settings, LPMs can handle various new tasks without gradient updates. Additionally, Chain-of-Thought is an improved prompt strategy that explicitly exposes the intermediate reasoning process leading to the output [74]. The above prompt strategies rely on a clear understanding of the context, offering precise guidance to help LPMs interpret the query's intent and deliver the optimal output (***ICL***). The ICL capability was first defined in [47], being considered one of the emergent abilities of LPMs [49]. Specifically, Wei et al. [48] explore the emergent abilities of several LPMs across multiple tasks. When the parameter scale reaches a certain threshold, the pre-trained model suddenly acquires some emergent ICL abilities. Brown et al. [47] also quantitatively demonstrate that the ICL capability enhances as the parameter scale expands from 0.1B to 175B. Generally, ICL refers to the capability to complete new tasks based on the understanding of the given context, which may improve as the parameter scale increases. This represents the initial level of leveraging the capabilities or benefits of LPMs.

However, an increasing number of studies find that further adaptation to specific tasks or domains requires additional model adjustments [75–77]. Fine-tuning, a prevalent technique in transfer learning, is frequently utilized in this adjustment process (***TF***). It can be classified into Full Fine Tuning (FFT) and Parameter-Efficient Fine Tuning (PEFT) according to the scale of parameters involved in the adjustment. While FFT offers significant performance gains, it is hindered by high training costs, catastrophic forgetting, and the need for high-quality fine-tuning data. To balance cost and effectiveness, PEFT has emerged as a popular fine-tuning approach by introducing a small number of trainable components, such as Adapter Tuning [78], Prompt Tuning [79], Prefix Tuning [80], LoRA [81], and QLoRA [82]. Many of these techniques have been open-sourced in [83]. Additionally, reinforcement learning (RL) plays a crucial role in fine-tuning by optimizing both process-based and outcome-based signals from specialized reward models [84, 85]. Depending on the reward source, RL can be categorized into Reinforcement Learning with Human Feedback (RLHF) and Reinforcement Learning with AI Feedback (RLAIF) [76]. In summary, the ultimate goal of fine-tuning is to adapt large pre-trained models to specific tasks or domains under controllable costs. This represents the second level of leveraging the capabilities or benefits of LPMs.

Moreover, language generation is a fundamental capability, involving next-token prediction from historical tokens [86] and generating task-specific text based on given conditions [87] (***LANG***). This capability stems from LPMs' deep semantic understanding. Broadly, LPMs can also employ similar approaches to handle tasks across different language forms, including code synthesis [88].

4.3 Design Rationale and Comparative Analysis of the Taxonomy

Justification of the Data-Modality Axis. To understand what roles Transformers assume in AIOps, several aspects need to be considered, including data modality, task objectives, architectural principles, and methodological approaches. Among them, data modality is chosen as the primary organizing axis of the taxonomy, as it defines the problem space in AIOps (Data). The main observability sources (*e.g.*, logs, metrics, and events) exhibit distinct structures and semantics, from which different task objectives arise (Target). These characteristics further influence the design of model architectures (Principle), while methodological strategies adapt accordingly (Approach).

Table 4. Distribution of the 61 model-design-oriented papers across the proposed taxonomy. Each number indicates paper counts. We classify the techniques based on data, targets, principles (TF represents *Transformer-Based*, TFE represents *Transformer-Encoder-Based*, TFD represents *Transformer-Decoder-Based*, LPM represents *(Large) Pre-Trained Model-Based*), and approaches (TAB represents *Transformer-Architecture-Based*, PE represents *Prompt Engineering*, FT represents *Fine-Tuning*, FM represents *Foundation Models*).

Data	**Target**	**Principle**				**Approach**			
		# TF	# TFE	# TFD	# LPM	# TAB	# PE	# FT	# FM
Log (20 papers)	Logging (2 papers)	1	-	-	1	-	1	1	1
	Log Parsing (6 papers)	-	-	-	6	-	5	1	-
	Log Representation Learning (2 papers)	-	2	-	-	-	-	2	2
	Log-Based Anomaly Detection (8 papers)	-	6	-	2	4	2	2	4
	Log-Based Failure Prediction (1 paper)	-	-	1	-	1	-	-	-
	Multi-Task Log Analysis (1 papaer)	-	-	-	1	-	-	1	1
Metric (22 papers)	Time Series Forecasting (17 papers)	5	3	4	5	6	4	3	6
	Time Series Classification (2 papers)	-	2	-	-	2	-	1	-
	Time Series Anomaly Detection (2 papers)	1	1	-	-	2	-	-	-
	Time Series Alignment (1 paper)	-	-	-	1	-	-	1	1
Event (19 papers)	Fault Understanding (8 papers)	1	4	1	2	1	3	2	3
	Root Cause Analysis (10 papers)	-	1	1	8	-	7	3	-
	Other (1 papers)	-	-	-	1	-	1	-	-

Compared with alternative taxonomies, each perspective alone has certain limitations. A task-oriented taxonomy organizes studies by application goals but often overlooks the alignment between Transformer capabilities and data characteristics. Architecture-oriented and approach-oriented taxonomies focus on model structures and technical methods, which help analyze design variations but tend to abstract away from the operational context of AIOps. In contrast, this survey adopts data modality as the organizing axis and incorporates target, principle, and approach dimensions, forming a *Data–Target–Principle–Approach* taxonomy. This design connects the problem space (Data and Target) with the modeling space (Principle and Approach), enabling a more systematic understanding of the "What" problem.

Comparison with Existing Taxonomies. As summarized in Tab. 1, existing surveys cover diverse perspectives on AIOps solutions; however, their taxonomies remain limited in both scope and depth. (1) Some studies discuss certain dimensions without establishing an explicit taxonomy. Ahmed et al. [35] focus on root causing and mitigation recommendation, while Le et al. [36] emphasize prompt strategies. These implicit dimensions are not elevated into a structured framework. (2) Several surveys adopt a single-dimensional taxonomy, organizing techniques primarily by either observability data [2, 13] or operational targets [37]. While useful, such taxonomies capture only one aspect of the problem. (3) Some recent works consider multiple aspects simultaneously. Wen et al. [34] discuss targets and network modifications, Su et al. [38] analyze forecasting and anomaly detection from targets and approaches, and Zhang et al. [3] consider data, targets, and approaches for LLM applications in the AIOps domain. Nevertheless, these dimensions are typically presented in parallel rather than integrated into a unified analytical framework that systematically explains how different design choices relate to one another. Moreover, prior surveys rarely provide an explicit capability-oriented framework for understanding the advantages of Transformers in AIOps. In contrast, our work proposes a multi-layer taxonomy (Data–Target–Principle–Approach) together with a two-level capability framework. Based on this, we derive new insights regarding the evolving roles of Transformers in AIOps, the alignment between the problem space and the modeling space, and the capability-oriented explanation of Transformer effectiveness.

5 Transformers in AIOps Domain

We present a detailed analysis of Transformer-based techniques in the AIOps domain, grounded in the 70 papers collected through the systematic search and selection process described in Sec. 2.1, without introducing additional

studies or excluding any papers from the surveyed corpus. Among these, 61 papers focus on model design (highlighted in this section), 11 papers address evaluation and benchmark studies (covered in Sec. 6), and 2 papers cover both aspects.

To ensure traceability between the surveyed corpus and the presented analysis, the narrative follows the research framework in Sec. 2.2 and the taxonomy in Sec. 4 (*Data–Target–Principle-Approach*). Specifically, techniques are first categorized by the data modality, then grouped by the AIOps tasks, and finally analyzed according to the structural principles and methodological approaches. Tab. 4 summarizes the distribution of the 61 model-design-oriented papers across the dimensions of the proposed taxonomy, serving as a bridge between the surveyed corpus and the subsequent analysis.

5.1 Transformers for Log Analysis

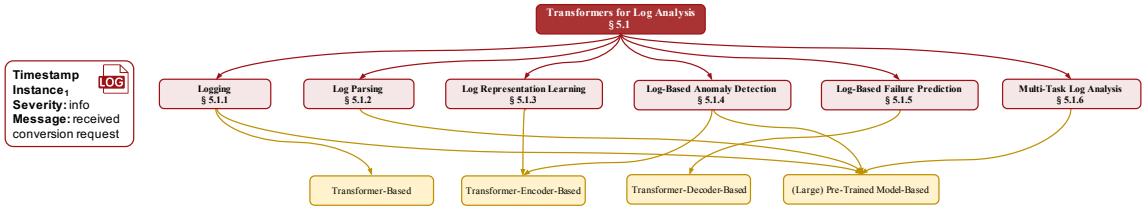

Fig. 5. Categories of Transformers for log analysis.

Logs are semi-structured textual data, with the structured part containing timestamps, log levels, source addresses, *etc.* , while the unstructured message content follows a fixed format [69]. They are generated by developer-embedded code at predefined points, recording events, status information, and alarms throughout the operation of a system, application, or service. Generally, log printing is regarded as a good programming practice, enabling developers to monitor key variables and trace the execution flow during runtime. This approach facilitates system development, debugging, testing, and maintenance.

As shown in Fig. 5, the log analysis process typically involves four steps: (1) Logging involves recording logs at key points in the program (Sec. 5.1.1). (2) Log Parsing involves preprocessing semi-structured log messages into structured templates or events, serving as the foundation for various downstream tasks (Sec. 5.1.2). (3) Log Representation Learning involves mapping raw data into a latent space, thereby capturing the intrinsic characteristics (Sec. 5.1.3). (4) Downstream Tasks perform essential AIOps functions through log data, including anomaly detection (Sec. 5.1.4), failure prediction (Sec. 5.1.5), and multi-task analysis (Sec. 5.1.6).

(1) For a long time, log printing relied heavily on developers manually configuring it during the program design phase. With software systems scaling up, this approach exposes several issues: inefficiency, dependence on expert knowledge, lack of standardized logging documentation, and maintenance challenges due to dispersed logging code sources, *etc.* As a result, there is a growing shift toward automated log printing. However, this approach faces three core challenges: determining where to log, what information to log, and the appropriate log level [69]. (2) Log messages are applied to downstream tasks after log data is generated and collected. Since most existing techniques require structured log inputs, log parsing becomes an essential step that transforms semi-structured log messages into structured templates or events. Traditional techniques encompass regular expression matching, delimiter-based parsing, and predefined pattern matching. However, they require manual configuration, are inefficient, and lack generalizability. Subsequent researchers proposed log parsing techniques based on clustering [89–91], frequent item mining [92, 93], and heuristics [94, 95], but

these still face limitations, such as the need for manual configuration of regular expressions, high preprocessing time, *etc.* Moreover, NeuralLog [96] empirically examines the severe impact of log parsing errors caused by out-of-vocabulary (OOV) words and semantic misunderstandings. It emphasizes that enhancing parsing accuracy is critical for effective log analysis. (3) Log preprocessing is not the sole critical factor; robust feature extraction also benefits a lot in improving downstream tasks. Common techniques like word2vec and TF-IDF are used for log vectorization, but they depend on hand-crafted features or domain-specific vectors, making them resource-intensive and less effective across multiple domains. Furthermore, general-purpose word embeddings are not tailored for log data, often resulting in suboptimal performance in log analysis. (4) A common approach for downstream tasks like log anomaly detection or failure prediction resembles the "next sentence prediction" task in NLP. It leverages neural networks to model the normal characteristics of log sequences and predict the most probable next log template. An anomaly is detected if the observed log deviates from the prediction or if the predicted template contains known fault patterns [30, 97]. In log sequence modeling, traditional RNN architectures like LSTM face challenges due to inherent recurrent dependencies, resulting in vanishing or exploding gradients and high computation time [21–24]. While Bi-LSTM captures contextual features in both directions, it still struggles with gradient-related issues. In contrast, CNN architectures lack recurrent dependencies, but their ability to capture contextual features is restricted by their receptive field [19, 20, 97]. Additionally, prior research rarely introduces unified models for multiple tasks, largely due to limitations in architectural generalization [21, 98, 99]. Therefore, modeling and analysis based on previous network architectures are more or less problematic.

Overall, existing log analysis techniques have the following drawbacks: (1) Labor-intensive and time-consuming processes, characterized by a strong dependence on expert knowledge and low efficiency. (2) Specificity to particular data scenarios, lacking generalization capabilities and failing to handle few-shot or zero-shot tasks. (3) Constrained modeling capabilities, limiting effective feature extraction and comprehensive data modeling. Recently, an increasing number of log analysis techniques have adopted Transformer architectures to tackle the aforementioned challenges. We organize these works into six categories based on their task objectives: logging, log parsing, log representation learning, log-based anomaly detection, log-based failure prediction, and multi-task log analysis.

5.1.1 Logging. The key to automating log generation is to address three fundamental aspects: determining where and what to log, and selecting the appropriate log level.

Transformer-Based. LANCE [100] collects a large corpus of publicly available Java code to train a Text-to-Text Transfer Transformer (T5) model [101]. During the pre-training phase, the model is trained on 6.8 million samples across two tasks: Firstly, randomly masking portions of the code blocks to learn general Java syntax, including the logging statements; and secondly, predicting the optimal locations for inserting log statements. In the fine-tuning phase, using 0.1 million samples, the model's task becomes more specific: generating log statements that include the location, log level, and message content.

(Large) Pre-Trained Model-Based. In contrast to LANCE, UniLog [50] takes a different approach, moving away from the conventional pre-training and fine-tuning paradigm. Instead, it leverages the in-context learning (ICL) capabilities of existing large language models. Specifically, it incorporates KNN-selected log examples directly into the prompt to facilitate automated log generation. Compared to fine-tuning a language model of the same scale, this approach requires less than 4% of the parameter fine-tuning time, thereby significantly reducing computational overhead.

5.1.2 Log Parsing. Most log analysis techniques rely on structured input, making log parsing critical for converting semi-structured raw logs into structured templates (*i.e.*, events) and parameters. The first step in this process is to identify the structure of the logs and then extract templates based on this structure. These templates capture the general

patterns of the logs, which are typically defined by the printing statements. Subsequently, the parser recognizes the parameters within the templates, representing the specific values, identifiers, or keywords found in the logs, and extracts them from the raw log entries.

(Large) Pre-Trained Model-Based. The essence of this process is to understand natural language and explore the commonalities between textual contents. Therefore, large language models with robust linguistic capabilities demonstrate significant potential for improving the accuracy and efficiency of log parsing, while also reducing the need for manual intervention. Significant efforts have been made in this direction, with initial successes already documented.

Some techniques pursue the gains offered by prompt strategies [36, 51]. ChatGPTParser [36] utilizes ChatGPT along with three simple yet effective prompts (*i.e.*, simple prompt, enhanced prompt, and few-shot prompt) to perform log parsing tasks on a dataset containing manually annotated logs from 16 different domains, with 2,000 logs per domain. Similar to ChatGPTParser, LogPrompt (Liu et al.) [51] also emphasizes the design of effective prompts, including Self-Prompt, Chain-of-Thought Prompt, and In-Context Prompt. The Self-Prompt approach enables the large language model to generate the prompt autonomously, thereby narrowing the disparity between human and machine languages and facilitating more intuitive interaction. On the other hand, the CoT Prompt aims to improve the model's interpretability by prompting it to provide reasoning for its judgments while explicitly defining the steps needed to complete the task. This strategy helps reduce hallucinations and enhance the reliability of generated outputs. In few-shot scenarios, the In-Context Prompt approach integrates a small number of supervised examples directly into the prompt, allowing for more effective guidance. Moreover, LogPrompt (Liu et al.) offers not only examples of log messages and their corresponding templates but also examples of both normal and anomalous logs, allowing the model to perform both log parsing and anomaly detection simultaneously. This study compares the effectiveness of using ChatGPT alone against that of the LogPrompt (Liu et al.) technique in log parsing. Both approaches confirm the potential of large language models in this task, while the differences in their performance emphasize the critical role of prompt strategies.

Certain techniques [102–105] strive to advance sampling-based frameworks that improve the performance of large language models in log parsing. DivLog [102] improves the accuracy of log parsing through a fixed-format prompt. Specifically, it begins by sampling a small subset of log messages from the sources using the Determinantal Point Process (DPP) to generate candidate prompt examples. It then selects the n log templates most similar to the target log message based on KNN-computed similarity, arranging them in ascending order for inclusion in the prompt. This strategy is driven by the recency bias characteristic of large language models. Additionally, the prompt directs the model to produce special <start> and <end> tags to address the issue of redundant output, facilitating the extraction of expected results. It is worth noting that DivLog uses GPT-3 as its backbone rather than ChatGPT, yet it still demonstrates strong performance in log parsing. LILAC [103] adopts an approach similar to DivLog for sampling candidate sets and selecting supervised examples for prompts. However, it also incorporates a cache block mechanism, designed to store all log templates in a hierarchical, tree-like structure. Before each log message is processed by the large language model and after each log template is output, this structure requires matching, insertion, or redefinition operations on the cache tree. This mechanism not only reduces query overhead on the language model, thereby enhancing parsing efficiency, but also ensures the consistency of parsing results. LLMParser [104] also adopts a clustering sampling algorithm to select representative and diverse log samples. Unlike DivLog and LILAC, these sampled subsets are used for few-shot fine-tuning, guiding the language model to "translate" raw logs into log templates. Consequently, LLMParser achieves an average log parsing accuracy of 96%. Moreover, LLMParser systematically evaluates the performance of different large language models on the log parsing task, revealing that in certain cases, smaller models may outperform larger ones while providing faster inference. Going a step further, LUNAR [105] eliminates the need for labeled samples

by introducing Log Contrastive Units (LCUs). LCUs, groups of logs with similar structures but differing parameters, enable LLMs to infer templates through comparative reasoning. To efficiently extract LCUs from large-scale log corpora, LUNAR applies hierarchical sharding based on log length and frequent tokens, followed by a hybrid ranking method that balances structural commonality and parameter variability. Once high-quality LCUs are identified, LUNAR assembles a demonstration-free prompt containing task instructions, parameter examples, and output-format constraints to guide LLMs in generating structured templates. This fully unsupervised approach not only removes annotation costs but also improves robustness to evolving log patterns.

5.1.3 Log Representation Learning. Pre-trained language models like BERT [106] have been widely applied to diverse NLP tasks, enabling efficient semantic representation without the need to train a new model from scratch. Yet, unlike typical natural language text, semi-structured log data often includes distinctive patterns, specific formats, and domain-specific information related to particular systems or applications. This requirement highlights the importance of language models capable of understanding domain knowledge and specialized terminology. As a result, general-purpose word embeddings, such as BERT, may not always be the most effective solution for handling log data.

Transformer-Encoder-Based. To obtain a unified log representation optimized for a range of downstream analysis tasks, some researchers have turned their attention to developing log-domain-aware pre-trained models [52, 107]. Biglog [52] utilizes a large-scale log dataset, comprising 0.45 billion logs from 16 diverse domains, to pre-train a stack-transformer model similar to BERT. This approach is designed to equip the model with the capability to capture both in-sentence and cross-sentence features intrinsic to log data. The unified log representations produced by the Biglog-encoder demonstrate impressive few-shot learning capability and adaptability across domains in various downstream tasks, including log parsing, anomaly detection, and failure prediction. Hil-BERT [107] introduces a hierarchical Transformer model designed to perform modeling at both the log token and log sequence levels. The model incorporates two mask-based self-supervised tasks tailored specifically for these levels. Pre-training was conducted on Loghub [108], a dataset spanning 17 real-world sources across diverse domains, including distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software. By leveraging the pre-trained model, high-quality log representations are generated and then fine-tuned via a linear layer to evaluate the normality of log sequences. Notably, experimental results demonstrate that initializing the model with BERT enhances the effectiveness of pre-training. This finding suggests that general-domain corpora support a certain degree of generalization capability, which can be advantageous when adapted to the log domain. Nonetheless, experiments also reveal that pre-training solely on log data from scratch does not lead to significant performance degradation. So pre-training exclusively within the log domain remains effective and can produce robust, satisfactory outcomes.

5.1.4 Log-Based Anomaly Detection. Fundamentally, log-based anomaly detection techniques focus on identifying the underlying patterns of normal system behavior, as reflected in event flows (including execution sequences and log printing time intervals) and log semantics.

Transformer-Encoder-Based. With the dual focus on log data, most techniques employ BERT as their core model, motivated by two main considerations.

On the one hand, BERT acts as a sequence modeler, where its bi-directional encoding of the Transformer architecture enables effective sequence-level contextual modeling [30, 97]. For example, LogBERT [97] leverages the BERT architecture to model sequence-level features of log templates while also extracting shared characteristics of normal log template sequences. To this end, it employs two self-supervised training tasks: The first task is masked log message prediction, akin to the pre-training strategy used in BERT. A certain proportion of log templates in the sequence is

selected and masked, and the cross-entropy loss between the model's output and the true templates is calculated. The second task is hypersphere volume minimization. This approach leverages CLS encoding at the beginning of each sequence, representing the features of the entire sequence. These CLS encodings are averaged to determine the centroid. The objective is to minimize the distances between each template sequence and the centroid, thus clustering normal log sequences more closely in the embedding space. During online detection, if the difference between the predicted log template and the actual output exceeds a certain threshold, the system is flagged as deviating from normal behavior, thereby an anomaly is detected. Like LogBERT, Loader [30] also trains a BERT-based model to predict the next log template. However, Loader introduces improvements to the anomaly detection strategy. Unlike LogBERT, which selects the k most probable log templates as normal candidates, Loader chooses templates whose cumulative probabilities exceed a threshold p. This approach mitigates accuracy sensitivity to the hyperparameter k and enhances robustness from a probabilistic perspective.

On the other hand, as a semantic encoder pre-trained on large-scale unlabeled text, BERT is widely utilized to derive high-quality representations of log data. SwissLog [109] introduces a dictionary-based log parsing approach that operates without the need for parameter tuning. Building on this log parsing process, it leverages BERT to extract semantic embeddings from log templates. These embeddings are then complemented by a fully connected layer that effectively maps the time intervals between log entries into a high-dimensional space. By concatenating the semantic and time embeddings, SwissLog generates a comprehensive log representation. This representation is then used in a binary classification task to distinguish normal from abnormal logs, employing a bi-directional LSTM network with an attention mechanism. The bi-directional structure captures the contextual features of the log sequence, while attention weights help reduce the impact of noisy or irrelevant logs. Furthermore, SwissLog constructs instance graphs based on the logs and identifies leaf nodes to produce a set of candidate root causes for detected anomalies.

Beyond treating these roles in isolation, certain techniques combine them, utilizing BERT simultaneously as a semantic encoder and a sequence modeler. NeuralLog [96] investigates how log parsing errors, especially those introduced by out-of-vocabulary (OOV) words and semantic misunderstandings, impact anomaly detection, and proposes an anomaly detection framework that functions independently of log parsing. After removing digits and punctuation from log messages, NeuralLog applies WordPiece Tokenization and BERT to encode these messages into semantic vectors. The sequence of semantic vectors is then fed into a BERT architecture, and the output is passed through pooling, dropout, and a binary classification fully connected layer to determine the probability of the log being normal or abnormal. LogPrompt (Zhang et al.) [110] focuses on prompt tuning for anomaly detection. Specifically, a Bi-LSTM is used to encode and project the prompt template, which is then concatenated with the semantic and sequential tokens of the logs and fed into a pre-trained language model (*e.g.*, BERT, RoBERTa, ALBERT). After extracting semantic and sequential features, the model predicts the value of the [MSK] token in the prompt template to indicate whether the system is anomalous. Notably, this work incorporates a focal loss function for fine-tuning, which mitigates the negative impact of log class imbalance while increasing focus on hard-to-classify logs by adding modulating factors and a focusing parameter to the cross-entropy loss. LogSynergy [53] further extends BERT to the task of cross-system log anomaly detection. It introduces an LLM–based event interpretation module, which employs ChatGPT to normalize heterogeneous log syntax by translating raw log messages into concise, standardized semantic descriptions. The interpreted log statements are subsequently encoded with a fine-tuned BERT model to generate unified event embeddings. Then, these embedding sequences are processed by System-Unified Feature Extraction (SUFE), a disentanglement mechanism built on a Transformer encoder that filters out system-specific information while preserving transferable, system-independent features for effective cross-system learning.

(Large) Pre-Trained Model-Based. In rigorous industrial settings, reducing false alarms is a precondition for anomaly detection techniques to gain widespread adoption. ScaleAD [54] is a tree-based parsing and lightweight anomaly detection technique that utilizes large language models to identify potential errors described in logs. It produces final diagnostic results by combining insights from a trie-based detection agent with LLM feedback, enhancing accuracy and reducing false alarms. Additionally, ScaleAD presents a case study evaluating the capability of ChatGPT to provide feedback comparable to that of on-call engineers. Also, it is well-understood that hallucinations in large language models pose a challenge for rigorous industrial applications. To mitigate this, RAGLog [111] employs retrieval-augmented generation. Using a vectorized database of normal log entries constructed through unsupervised clustering, the framework first retrieves relevant log entries related to the queried log entry. It then employs a QA-based prompt to guide the LLM in determining whether the queried log entry represents normal or abnormal behavior.

5.1.5 Log-Based Failure Prediction. The approach for failure prediction resembles that of anomaly detection, also leveraging Transformers to capture the contextual features of log sequences. However, unlike most unsupervised anomaly detection techniques, which capture normal patterns, failure prediction typically does not differentiate between normal and abnormal system states.

Transformer-Decoder-Based. Clairvoyant [112] first identifies specific failure events, *i.e.*, log template, and then utilizes a stacked Transformer-decoder architecture to predict the next log event for a historical log sequence. By sliding a time window, it generates a sequence of upcoming events of a specified length, which then enables inference of the likelihood of an impending failure event.

5.1.6 Multi-Task Log Analysis. The aforementioned techniques are largely task-specific, with separate models trained for individual objectives such as log parsing or anomaly detection. Although such specialization can maximize performance at the task level, it results in substantial maintenance overhead and limited flexibility. Leveraging the generality of large pre-trained models, recent studies have begun to explore unified approaches to multi-task log analysis.

(Large) Pre-Trained Model-Based. LogLM [55] exemplifies this direction by employing instruction tuning to support diverse log analysis tasks, including parsing, anomaly detection, interpretation, root cause localization, and solution recommendation. Specifically, it reformulates traditional task-specific log-label pairs into a unified instruction–response paradigm. This design enables a pre-trained model to capture semantic commonalities across tasks and operate within a single architecture, thereby eliminating the need for endless model designs or large-scale labeled datasets for each task. Overall, LogLM serves as a notable practice in evolving IT operations from conventional task-focused approaches toward natural language-enabled, intelligent management.

5.1.7 Summary. In Tab. 5, we summarize the Transformer-based models for log analysis, providing information on Targets, Principles, Approaches, Core Methods, and Capabilities.

The aforementioned techniques highlight that Transformer-based log analysis mainly builds on three core ideas: Transformer architecture, foundation models, and prompt design. (1) Firstly, the most fundamental approach involves the direct application of Transformer architectures [30, 53, 96, 97, 112]. These techniques primarily aim to model log sequences to detect sequence-level anomalies. However, adapting models to specific scenarios or tasks often incurs significant costs in terms of design, simulation, and verification. (2) To address this challenge and improve generalization, certain techniques focus on domain-specific foundation models [52, 55, 100, 107]. Among these, LANCE serves as a foundation model tailored for log printing. The Java syntax knowledge acquired during its pre-training phase is potentially transferable to other tasks. Meanwhile, BigLog and HilBERT emphasize unified log representations,

Table 5. Summary of Transformer-based models for log analysis. We classify the techniques based on targets, principles (TF represents *Transformer-Based*, TFE represents *Transformer-Encoder-Based*, TFD represents *Transformer-Decoder-Based*, LPM represents *(Large) Pre-Trained Model-Based*), approaches (TAB represents *Transformer-Architecture-Based*, PE represents *Prompt Engineering*, FT represents *Fine-Tuning*, FM represents *Foundation Models*), and core methods. Additionally, we list the capabilities or benefits of the Transformer (SEQ represents *Sequence Modeling*, PARALLEL represents *Parallelization and Non-Recurrent Dependencies*, PTKL represents *Pre-Training and Knowledge Utilization*, ICL represents *In-Context Learning*, TF represents *Transfer Learning and Domain Adaptation*, LANG represents *Language Capacity*).

Target	Technique	Principle				Approach				Core Methods	Capabilities or Benefits					
		TF	TFE	TFD	LPM	TAB	PE	FT	FM		SEQ	PARALLEL	PTKL	ICL	TF	LANG
Logging	LANCE [100]	✓						✓	✓	T5 + Pre-Training + Fine-Tuning			✓		✓	✓
	UniLog [50]				✓		✓			GPT3-Based Codex + KNN Comparison + Few-Shot			✓	✓		✓
Log Parsing	ChatGPTParser [36]				✓		✓			ChatGPT + Zero/Few-Shot + Enhanced Prompt			✓	✓		✓
	LogPrompt (Liu et al.) [51]				✓		✓			ChatGPT + Self-Prompt + CoT + In-Context Prompt			✓	✓		✓
	DivLog [102]				✓		✓			GPT3 + DPP + KNN + Few-Shot			✓	✓		✓
	LILAC [103]				✓		✓			ChatGPT + Candidate Sampling + KNN + Cache Block			✓	✓		✓
	LLMParser [104]				✓			✓		Flan-T5 + LLaMA + ChatGLM + Few-Shot Tuning			✓	✓	✓	✓
	LUNAR [105]				✓		✓			ChatGPT + Sharding + Reasoning			✓	✓		✓
Log Representation Learning	Biglog [52]		✓					✓	✓	BERT + Pre-Training + Fine-Tuning + Clustering			✓		✓	✓
	HilBERT [107]		✓					✓	✓	Hierarchical BERT + Pre-Training + Fine-Tuning	✓		✓		✓	✓
Log-Based Anomaly Detection	LogBERT [97]		✓			✓				BERT + Mask Prediction + Hypersphere Volume Minimization	✓	✓				
	Loader [30]		✓			✓				BERT + Top-P Selection	✓	✓				
	SwissLog [109]		✓					✓	✓	BERT + Attention-Based Bi-LSTM + Heuristic Algorithm			✓			✓
	NeuralLog [96]		✓			✓			✓	BERT + WordPiece Tokenization	✓	✓	✓			✓
	LogPrompt (Zhang et al.) [110]		✓					✓	✓	Bi-LSTM Head + Prompt Tuning + Focal Loss	✓	✓	✓		✓	✓
	LogSynergy [53]		✓			✓			✓	BERT + ChatGPT + Adaptation Network	✓	✓	✓	✓		✓
	ScaleAD [54]				✓		✓			Trie-Based Detection Agent + ChatGPT + Zero-Shot			✓	✓		✓
	RAGLog [111]				✓		✓			Embedding Model + RAG + Few-Shot			✓	✓		✓
Log-Based Failure Prediction	Clairvoyant [112]			✓		✓				Stack Transformer-Decoder + Linear Layer + Softmax Layer	✓	✓				✓
Multi-Task Log Analysis	LogLM [55]				✓			✓	✓	LLaMA2 + Instruction Tuning			✓	✓	✓	✓

demonstrating their effectiveness in downstream applications such as log parsing, anomaly detection, and failure prediction. LogLM, on the other hand, unifies multiple log analysis tasks into a question-answering paradigm. Beyond domain-specific models, general-purpose foundation models also play a role in log analysis. For example, SwissLog [109], NeuralLog [96], and LogSynergy [53] leverage BERT for log encoding; ScaleAD [54] utilizes ChatGPT for feedback generation; RAGLog [111] employs retrieval-augmented generation to mitigate hallucinations; and LogSynergy [53] adopts ChatGPT to standardize heterogeneous log syntax. (3) Last but not least, techniques centered on prompt design [36, 50, 51, 102, 103, 105] also become popular. Combining existing large language models with either manually crafted or machine-generated prompt templates, these techniques excel in tasks such as logging and log parsing, which are oriented toward semantic understanding.

Transformers in the log domain address a wide range of tasks, including log printing, parsing, representation learning, anomaly detection, failure prediction, and multi-task analysis. These tasks collectively encompass the diverse automation and analysis needs that arise throughout the operation and maintenance lifecycle of logs. Depending on the specific characteristics of each task, Transformers adopt distinct roles and demonstrate unique capabilities. For tasks such as logging and log parsing, which are oriented toward semantic understanding, the focus lies on *Pre-Training and Knowledge Utilization* as well as the *Language Capacity* of Transformers. Pre-training on large-scale corpora equips Transformers with the ability to comprehend execution flows and log text, enabling them to generate meaningful log statements at critical program points [50, 100]. In the case of log parsing, Transformers often rely on their *In-Context Learning* capability, where carefully designed prompt strategies guide them to leverage latent linguistic knowledge and parse logs into structured templates [36, 51, 102, 103, 105]. The targets of log representation learning and multi-task analysis, on the other hand, necessitate the development of foundation models, showcasing not only the *Pre-Training and Knowledge Utilization* but also the *Transfer Learning and Domain Adaptation* capabilities of Transformers [52, 55, 107]. Domain-specific models trained on log data significantly outperform general-purpose

models by incorporating domain knowledge and specialized terminology. Through fine-tuning, these representations effectively transfer knowledge to various downstream tasks. As for anomaly detection and failure prediction, the emphasis shifts to extracting contextual information and sequence features from log sequences. These tasks leverage the *Sequence Modeling* and *Parallelization and Non-Recurrent Dependencies* capabilities of the Transformer architecture, enhancing both accuracy and computational efficiency [30, 53, 96, 97, 110, 112].

In general, logs play a crucial role in helping on-call engineers interpret system runtime behavior from two perspectives: sequential information and natural language text. These perspectives align with the basic and advanced capabilities of Transformers, respectively. From one perspective, log sequences inherently represent system workflows. Deviations or anomalies in these sequences often indicate that workflows are not functioning as expected, potentially signaling system failures. Employing Transformers to model the normal patterns of log sequences has become a common practice for automating the analysis of system behavior and ensuring system reliability. From another perspective, logs are semi-structured text that encapsulates rich semantic information, which can be exploited to identify implicit performance issues. Advances in natural language processing have introduced a wealth of general-purpose pre-trained models and semantic understanding solutions. As a result, analyzing the semantic content of logs no longer requires building models from scratch. Instead, the primary challenges lie in domain adaptation and task-specific requirements unique to log analysis.

5.2 Transformers for Metric Analysis

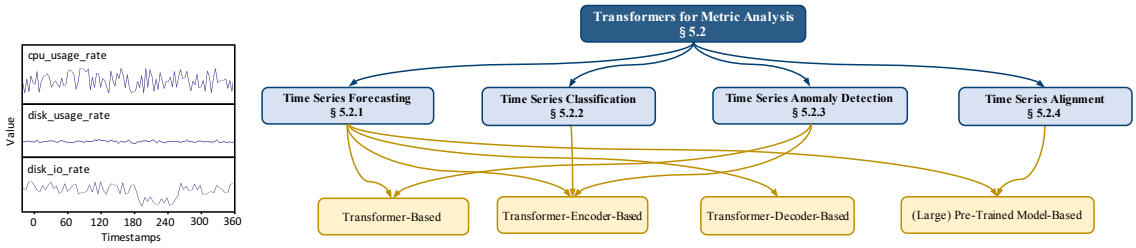

Fig. 6. Categories of Transformers for metric analysis.

Metrics, such as throughput, CPU utilization, and memory usage, are used to quantitatively monitor the status and performance of hardware, systems, and services. At a given time point, metrics provide a snapshot of the system status [10]. Over a continuous period, however, metrics in the form of time series exhibit characteristics such as trend, seasonality, periodicity, and random noise [34]. These characteristics are crucial for monitoring system changes, identifying anomalous behaviors, detecting potential problems, and optimizing system operations.

Commonly, metric analysis tasks can be categorized into four types as shown in Fig. 6: (1) Time Series Forecasting involves analyzing historical data to predict the trend of metrics at specific time points or over periods (Sec. 5.2.1). Accurate time series forecasting aids in failure prediction and prevention, resource optimization and management, trend analysis, and other critical functions. (2) Time Series Classification involves categorizing data patterns, which can be used to identify anomalous patterns or performance degradation [113] (Sec. 5.2.2). (3) Time Series Anomaly Detection involves identifying unusual data points or patterns within a system to detect anomalous behaviors, unexpected events, or potential failures, making it one of the core tasks in AIOps (Sec. 5.2.3). To some extent, time series forecasting and classification form the foundation of anomaly detection. Trends and deviations identified through forecasting, when

combined with patterns extracted via classification, could significantly enhance the accuracy of anomaly detection. (4) Time Series Alignment for Understanding and Reasoning (Sec. 5.2.4). This task aligns time series with natural language, transforming metric analysis into language-based understanding and reasoning [59]. Such alignment forms the foundation for evolving human-AI interaction in AIOps.

Traditional techniques are primarily statistical, including moving averages, exponential smoothing, and ARIMA [114, 115]. While they can capture features such as smoothing, trends, and seasonality, their capacity to model nonlinear relationships is constrained, as they generally rely on linear assumptions [115, 116]. Moreover, they struggle with handling missing data, interpolation, outliers, feature selection, and model complexity. Deep learning models based on RNN, CNN, *etc.* , extract features and deal with nonlinear relationships automatically. However, these models are prone to overfitting and poor generalization when the amount of data is small or limited to a specific domain [21, 98, 99]. Last but not least, long-term time series forecasting remains a significant challenge in metric analysis, focusing on predicting data for more distant future time points [34]. Specifically, models must capture more complex factors due to the high uncertainty, the dynamic nature, and the presence of seasonality and cyclicality. Moreover, long-term forecasting inevitably involves long-range dependencies, *i.e.*, correlations between distant historical data and current data points. This requires the model to capture essential contexts while mitigating information decay. Frustratingly, existing architectures are often overstretched in addressing these challenges.

To overcome the limitations, researchers turn to the potential of Transformers in metrics analysis. In the following sections, we present notable works of these Transformers in the areas of time series forecasting, classification, anomaly detection, and alignment for understanding and reasoning.

5.2.1 Time Series Forecasting. Time series forecasting focuses on predicting future values based on historical data to understand system changes. Accurate long-term time series forecasting is highly valuable in practical applications; however, several challenges remain, such as capturing long-range dependencies that influence performance and addressing the temporal and spatial complexity that impacts efficiency [34].

Initially, Zeng et al. [117] question the effectiveness of Transformers for time series forecasting. They point out that the self-attention mechanism is permutation-invariant and anti-order, causing inevitable temporal information loss even with various positional encoding techniques, and in some cases, it performs worse than a simple fully connected layer. To optimally harness the potential of Transformers in metric analysis, researchers have proposed various improvements, including Transformer architecture variants (such as self-attention mechanism adaptations [29, 31, 118], multi-scale frameworks [119], and token input modifications [120, 121]), and large pre-trained Transformers (such as time-series foundation models [32, 122–126] and large language models for time series [56, 57, 127–129]).

Transformer-Based. Some techniques focus on variants of the self-attention mechanism in different scenarios. To mitigate the excessive computation and memory consumption of Transformers when handling long sequences, Informer [31] introduces sparse self-attention and self-attention distillation, which effectively reduces the time and space complexity to $O(L \cdot \log L)$. However, this sparse representation risks losing critical information. Autoformer [29] addresses this by proposing an auto-correlation mechanism, which leverages the periodicity of time series to perform self-attention operations at the series level. This approach fully exploits the sequence information while reducing spatio-temporal complexity. Additionally, Autoformer explicitly decomposes the input time series into trend-cyclical and seasonal components through progressive decomposition, thereby alleviating the issue of entangled temporal patterns, which may result in unreliable predictions. To improve predictability, stationarization is commonly applied in previous studies to reduce the non-stationarity of the original series [130, 131]. However, this approach often removes the

inherent non-stationarity of the series, leading to poor performance in forecasting burst events. To tackle the dilemma between series predictability and model capacity, Non-stationary Transformer [118] proposes series-stationarization and de-stationary attention, the latter designed to reintroduce non-stationary information into temporal dependencies.

Through introducing a generalized multi-scale framework, Scaleformer [119] can be seamlessly integrated with other Transformer-based models, acting broadly orthogonally to their own contributions. By leveraging the rich information contained in time series of various scales, this framework samples the time series at different rates, ranging from low to high frequencies, to achieve precise predictions from coarse to fine granularity. Additionally, cross-scale normalization is implemented to reduce errors caused by varying scale distributions, and adaptive loss is employed to minimize error accumulation during iterations.

Moreover, TimeGPT [122] is the first foundational model capable of zero-shot inference for time-series forecasting. Utilizing a Transformer architecture, it is trained on a large-scale time-series dataset, enabling it to tackle various time-series tasks through either zero-shot learning or fine-tuning. At the time of this writing, TimeGPT is accessible only via an API [132].

Transformer-Encoder-Based. Some techniques make straightforward yet effective modifications to the input tokens of the Transformer encoder. PatchTST [120] aggregates multiple time steps into a single patch as input tokens, allowing for the extraction of rich semantic information while significantly reducing computational complexity. Furthermore, unlike other Transformers that use channel-mixing, PatchTST employs a channel-independence approach, where each dimension of the multivariate time series is fed separately into the Transformer backbone. Each channel corresponds to a univariate time series, sharing the same embedding and Transformer weights. Given that different variables within a time step represent distinct physical meanings, a token formed by a single time step is unlikely to convey useful information. Therefore, iTransformer [121] takes an inverted view on time series and embeds the entire time series of each channel into a token. This approach allows for more variate-centric representations of the time series, which can be better leveraged through attention mechanisms for multivariate correlating. Going beyond token-level redesign, MOIRAI [124] introduces a universal paradigm through pretraining a masked encoder-only Transformer. It incorporates multi-patch-size projections to handle diverse frequencies, Any-variate Attention to support arbitrary variate numbers, and a flexible mixture distribution for probabilistic and metric-agnostic forecasting. These designs enable strong generalization, achieving competitive zero-shot performance against full-shot baselines.

Transformer-Decoder-Based. With the widespread success of large pre-trained models in fields such as NLP and CV, researchers have begun to explore their applications in time series prediction. Two prominent directions have emerged: the Time Series Foundation Model (TSFM) and the Large Language Model for Time Series (LLM4TS) [133, 134].

TSFM refers to large pre-trained models trained from scratch, tailored for time-series data. These models are designed to capture universal features and patterns, such as periodicity, trend, and seasonal variations, from large-scale datasets, which can then be fine-tuned and applied to different data scenarios and downstream tasks. Representative TSFMs include TimeGPT [122] and MOIRAI [124], both discussed earlier, using encoder-decoder and encoder-only Transformer architectures, respectively. Following these architectures, subsequent efforts have centered on decoder-only designs, which have given rise to several influential models. One of the first foundation models is Lag-Llama [123], a general univariate probabilistic forecasting model based on the LLaMA architecture, which leverages lag features within time windows. In contrast to TimeGPT, which employs conformal prediction, Lag-Llama's final layer outputs a probabilistic distribution using the Student's t-distribution. Similarly producing probabilistic outputs, Toto [125] extends the approach to multivariate observability metrics, employing proportional factorized attention for efficient modeling of temporal and channel dependencies, and a Student-T mixture head for robust capture of complex dynamics. TimesFM [126]

introduces a patch-based approach for TSFMs that pairs causal self-attention with output patches longer than the input, reducing autoregressive steps and supporting arbitrary forecasting horizons. Pretrained on a mixture of real and synthetic data, it demonstrates enhanced generalization. Motivated by data-scarce and heterogeneous scenarios, Timer [32] unifies multiple time series into a single-sequence format and adopts a GPT-style architecture. It further reformulates various time series tasks within a unified generative framework, enabling broad applicability across different tasks.

(Large) Pre-Trained Model-Based. TSFMs are well-suited to handle diverse characteristics and dependencies, leading to more accurate predictions. However, this approach typically requires significant upfront investment, including access to high-quality, large-scale datasets, substantial computational resources, and extended training periods, among other demands. In contrast, LLM4TS mitigates these challenges by aligning pre-existing large language models with time series tasks. To enhance the ability to handle out-of-distribution data, LLM4TS adopts strategies such as input transformations [127], prompt engineering [128], and parameter-efficient fine-tuning (PEFT) [56, 129], including approaches like layer normalization tuning, LoRA, and prompt tuning, as well as the reprogramming technique [57].

LLMTIME [127] completes the time series prediction task with digital preprocessing. Mimicking natural language, it tokenizes each digit separately by inserting spaces between them and uses commas to separate each time step in the sequence. Additionally, values are scaled using a MinMaxScaler to constrain the range before input. LLMTIME, to some extent, demonstrates that LLMs can serve as zero-shot time series forecasters. Furthermore, PromptCast [128] highlights the potential of prompt engineering. It embeds numerical inputs and outputs into a structured prompt template, effectively transforming the time series prediction task into a natural language generation problem, a task in which LLMs excel.

However, relying solely on the zero-shot capability of LLMs is insufficient for addressing more complex scenarios or intricate data patterns. In such cases, PEFT helps a lot. Using GPT-2 as its backbone, LLM4TS [129] incorporates two PEFT strategies, Layer Normalization Tuning and LoRA, when facing data-efficiency scenarios. To better align with the time series forecasting task, LLM4TS implements a two-stage fine-tuning process. The first stage aligns LLMs with the unique characteristics of time series data, while the second stage adapts the model specifically for time series forecasting tasks. In addition to inheriting the channel-independent and patching designs in PatchTST [120], LLM4TS also introduces a temporal encoding layer, allowing the model to process multi-scale temporal information effectively. TEMPO [56] employs learnable prompts to extract comprehensive features from raw data. Initially, TEMPO decomposes the input time series into three key components: seasonal, trend, and residual terms, which are then mapped into the hidden space to form input embeddings. Subsequently, through prompt tuning, TEMPO constructs a learnable prompt pool that assigns tailored prompts to each decoupled component, guiding LLMs in representation learning and forecasting process.

Unlike PEFT, which involves adjusting model parameters or adding trainable components into the backbone model, the reprogramming technique directly alters data inputs to induce different behaviors within a frozen language model during inference. TIME-LLM [57], for example, employs a learnable patch reprogram to convert time series data into text-based prototype representations, thereby enhancing the comprehension and reasoning abilities of pre-trained language models. Additionally, the Prompt-as-Prefix mechanism in TIME-LLM provides supplementary information by incorporating dataset context, task instructions, and input statistics.

5.2.2 Time Series Classification. Time series data exhibit diverse and rich patterns. For instance, common fluctuation patterns include sudden increase, level shifts downward, steady growth, spike, and dip [113]. The classification of

time series data reveals the operational condition. Achieving this, however, requires effective modeling of temporal representations.

Transformer-Encoder-Based. The TF-Encoder-Based Framework [135] introduces a representation learning approach for multivariate time series, leveraging the Transformer encoder architecture: unsupervised pre-training is carried out by constructing an autoregressive denoising task, where a certain percentage of inputs in the multivariate time series data are randomly masked, and the model is trained to predict these masked values. Next, the pre-trained model is fine-tuned for downstream tasks such as classification and prediction. Notably, the framework incorporates learnable positional encoding to enhance the flexible interpretation of sequence information and employs batch normalization to reduce the impact of outliers. This approach allows the model to capture the intrinsic features of the data in a label-less manner, aiding the acquisition of generalized representations.

Considering that multivariate time series contain interrelated channels, Gated Transformer Networks (GTN) [58] utilize an extended two-tower framework to capture both step-wise temporal information and channel-wise spatial information. A gating mechanism is then employed to integrate and optimize the features extracted by the two towers. Extensive experiments on 13 datasets demonstrate the effectiveness of GTN's feature modeling in time series classification.

5.2.3 Time Series Anomaly Detection. Typically, time series anomaly detection involves identifying outliers or abnormal patterns in time series data that deviate from expected norms. This process allows for rapid troubleshooting of system anomalies and facilitates the identification of potential root causes.

Transformer-Based. TranAD [136] is a Transformer-based model for anomaly detection and diagnosis. Traditional encoder-decoder models often struggle to capture short-term trends, leading to the missed detection of anomalies with minor deviations. TranAD addresses this by using focus score-based self-conditioning and adversarial training to amplify anomalies. Specifically, it consists of two encoders and two simplified decoders that predict the reconstruction of input time-series windows in two stages. The first stage generates an approximate reconstruction, with deviations termed as the focus score. In the next stage, the focus score from the first decoder maximizes the reconstruction error of the self-conditioned output, to enhance their impact in subsequent predictions, thereby making minor deviations more pronounced.

Transformer-Encoder-Based. Due to the limited occurrence of anomalies, establishing a strong association between anomalies and the entire sequence is challenging, as anomalies are primarily linked to neighboring time points. This distinction allows for differentiating between normal and abnormal points. The Anomaly Transformer [28] leverages this concept by modeling both local-prior and global-series associations at each time point. It computes the association discrepancy to adjust the self-attention mechanism into an anomaly-attention mechanism. As a result, anomalies generally exhibit smaller discrepancies between prior and series associations compared to normal points, making them more distinguishable.

5.2.4 Time Series Alignment for Understanding and Reasoning. Time series, as non-natural language data, often require complex analysis before their patterns can be translated into human-readable semantics. Bridging this gap entails aligning time series with natural language, enabling models to understand and interpret temporal behaviors, and to support higher-level reasoning and decision-making.

(Large) Pre-Trained Model-Based. In response to the lack of aligned time series with text data, ChatTS[59] introduces an attribute-based framework for synthetic data generation. This framework decomposes time series into four fundamental components (trend, periodicity, noise, and local fluctuation), allowing precise and controllable data

construction. Building on this, the Time Series Evol-Instruct (TSEvol) algorithm produces diverse question–answer pairs through staged attribute evolution. The model also integrates a context-aware encoder with a value-preserving normalization strategy to ensure numerical fidelity during training. ChatTS is fine-tuned from QWen2.5-14B-Instruct in two stages: large-scale alignment for cross-modal grounding, followed by supervised fine-tuning to strengthen reasoning. With its strong capabilities in time series understanding and reasoning, the model is further able to support downstream tasks such as forecasting, anomaly detection, classification, and question answering.

Table 6. Summary of Transformer-based models for metric analysis. We classify the techniques based on targets (TS represents Time Series), principles (TF represents *Transformer-Based*, TFE represents *Transformer-Encoder-Based*, TFD represents *Transformer-Decoder-Based*, LPM represents *(Large) Pre-Trained Model-Based*), approaches (TAB represents *Transformer-Architecture-Based*, PE represents *Prompt Engineering*, FT represents *Fine-Tuning*, FM represents *Foundation Models*), and core methods. Additionally, we list the capabilities or benefits of the Transformer (SEQ represents *Sequence Modeling*, PARALLEL represents *Parallelization and Non-Recurrent Dependencies*, PTKL represents *Pre-Training and Knowledge Utilization*, ICL represents *In-Context Learning*, TF represents *Transfer Learning and Domain Adaptation*, LANG represents *Language Capacity*).

Target	Technique	Principle				Approach				Core Methods	Capabilities or Benefits					
		TF	TFE	TFD	LPM	TAB	PE	FT	FM		SEQ	PARALLEL	PTKL	ICL	TF	LANG
TS Forecasting	Informer [31]	✓				✓				Probsparse Self-Attention + Self-Attention Distilling + Generative Style Decoder	✓	✓				
	Autoformer [29]	✓				✓				Auto-Correlation + Series Decomposition	✓	✓				
	Non-stationary Transformer [118]	✓				✓				Series Stationarization + De-Stationary Attention	✓	✓				
	Scaleformer [119]	✓				✓				Multi-Scale Framework + Cross-Scale Normalization	✓	✓				
	TimeGPT [122]	✓							✓	Pre-Training + Conformal Prediction + Zero-Shot + Fine-Tuning			✓	✓	✓	
	PatchTST [120]		✓			✓				Subseries-Level Patches + SSL + Channel Independence	✓	✓				
	iTransformer [121]		✓			✓				Inverted Dimensions of Time Series	✓	✓				
	MOIRAI [124]		✓						✓	Multi-Patch Projection + Any-Variate Attention + Mixture Likelihood + Pre-Training + Zero-Shot	✓		✓	✓		
	Lag-LlaMA [123]			✓					✓	LLaMA + Lag Features + Pre-Training + Probabilistic Forecasting + Zero-Shot	✓		✓	✓		
	Toto [125]			✓					✓	Proportional Factorized Space–Time Attention + Student's T Mixture Head + Pre-Training + Zero-Shot	✓	✓	✓	✓		
	TimesFM [126]			✓					✓	Patch Strategy + Pre-Training + Zero-Shot	✓		✓	✓		
	Timer [32]			✓					✓	Sequence Conversion + Windowed Sampling + Pre-Training + Unified Generative Task			✓	✓	✓	
	LLMTIME [127]				✓		✓			GPT/LLaMA + Digital Preprocessing + Zero-Shot			✓	✓		
	PromptCast [128]				✓		✓	✓		GPT Families + Prompt Learning + Fine-Tuning + Zero-Shot			✓	✓	✓	
	LLM4TS [129]				✓			✓		GPT2 + Layer Normalization Tuning + LoRA + Forecasting Fine-Tuning + Few-Shot			✓	✓	✓	
	TEMPO [56]				✓		✓	✓		GPT2 + Seasonal and Trend Decomposition + Prompt Pool + LoRA + Few-Shot			✓	✓	✓	
	TIME-LLM [57]				✓		✓			LLaMA-7B + Patch Reprogram + Prompt-as-Prefix + Zero/Few-Shot			✓	✓		
TS Classification	TF-Encoder Framework [135]		✓			✓		✓		Learnable Positional Encoding + SSL + Fine-Tuning	✓	✓	✓		✓	
	GTN [58]		✓			✓				Channel-Wise and Step-Wise Correlation Modeling + Gating Mechanism	✓	✓				
TS Anomaly Detection	Anomaly Transformer [28]	✓				✓				Prior and Series Association + Anomaly-Attention Mechanism + Minimax Strategy	✓	✓				
	TranAD [136]		✓			✓				Focus Score-Based Self-Conditioning + Adversarial Training + Meta Learning	✓	✓				
TS Alignment	ChatTS [59]				✓			✓	✓	QWen2.5-14B + Data Synthesis + Time Series Evol-Instruct + Fine-Tuning			✓	✓	✓	✓

5.2.5 Summary. In Tab. 6, we summarize the Transformer-based models for metric analysis, providing information on Targets, Principles, Approaches, Core Methods, and Capabilities.

Broadly, the techniques mentioned above can be categorized into two groups: Transformer architectures and foundation models. The first group leverages the Transformer architecture to capture spatial and temporal information. Various enhancements have been introduced to the naive architecture, including attention mechanism variants [28, 29, 31, 118], orthogonal frameworks [119, 136], token input modifications [120, 121], and gating mechanisms [58]. The second group prioritizes generalizable metric analysis over scenario-specific architectural design. Consequently, it focuses on foundation models, which have evolved in two directions: Time Series Foundation Models (TSFM) and Large Language Models for Time Series (LLM4TS). TSFM [32, 122–126] originates from pre-training paradigm and requires substantial computational resources. In contrast, LLM4TS adapts existing large language models for time series tasks without necessitating training from scratch. To accommodate time series data, LLM4TS typically employs input transformation [127], prompt engineering [128], PEFT [56, 129], and reprogramming technique [57]. Further ahead, similar to LogLM [55] in log analysis, ChatTS [59] also transforms multi-task metric analysis into natural language dialogue, paving the way for instruction-based IT operations.

Transformers in the metric domain are capable of tackling various tasks, including time series forecasting, classification, anomaly detection, understanding, and reasoning. To some extent, features such as trends and deviations derived

from forecasting, along with data patterns identified through classification, contribute to effective anomaly detection. Throughout feature extraction and subsequent analysis, Transformers assume distinct roles and exhibit unique strengths. On the one hand, the Transformer architecture is widely utilized for time series modeling, taking full advantage of its *Sequence Modeling* and *Parallelization and Non-Recurrent Dependencies* capabilities [28, 29, 31, 58, 118–121, 135, 136]. On the other hand, some techniques exploit Transformers' *Pre-Training and Knowledge Utilization* capabilities by pre-training on extensive high-quality temporal data to develop foundation models [32, 122–126]. Meanwhile, others leverage existing large language models to analyze time series, showcasing the *Transfer Learning and Domain Adaptation* capabilities of Transformers [56, 57, 59, 127–129].

To summarize, metrics sketch system status through quantitative data and exhibit key characteristics such as temporal dependency, trends, seasonality, periodicity, and non-stationarity, enabling on-call engineers to intuitively monitor software systems. Whether in Transformer architectures or large pre-trained models, the essence of metric analysis lies in uncovering temporal dependencies within individual sequences and spatial dependencies among multidimensional sequences, thereby enhancing downstream tasks of IT operations. This is precisely where Transformers excel. However, as our research indicates, most existing techniques focus on general-purpose time series analysis. In the AIOps domain, metric analysis still faces significant challenges, including data quality issues, inadequate anomalous samples, external influences, deployment costs, real-time processing of large-scale data, and model interpretability. By examining representative techniques in time series analysis and exploring the capabilities of Transformers, we aim to advance metric analysis within AIOps.

5.3 Transformers for Event Analysis

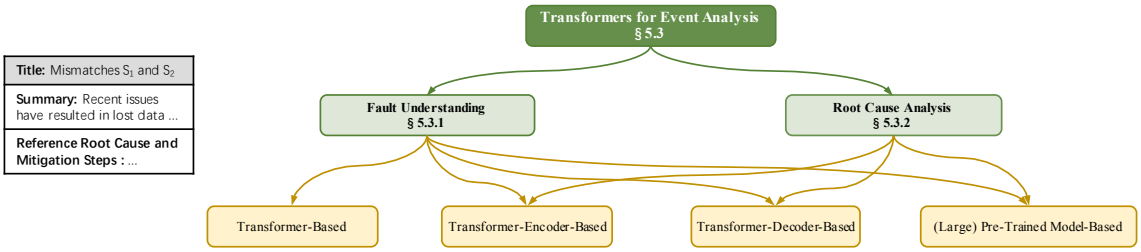

Fig. 7. Categories of Transformers for event analysis.

Event data, also referred to as incidents from the cloud side or tickets from the customer side, records system failures in natural language, providing detailed information such as timestamps, symptoms, affected components, and mitigation steps [37, 60]. This data combines unstructured descriptions in natural language with structured metadata, serving as a critical resource for enhancing system reliability.

A typical event lifecycle includes four stages [6, 10, 137]: (1) Detection: When a failure occurs, internal or external users can report it, or monitoring tools configured by developers can detect it automatically. (2) Triage: Once an event is detected, relevant tickets must be analyzed, aggregated, and summarized. These processed tickets are then assigned to appropriate on-call engineers for resolution according to an initial categorization. (3) Diagnosis: This stage focuses on pinpointing the root cause of the failure, a process that often demands substantial time and effort. (4) Mitigation: After identifying the root cause, suitable mitigation measures are implemented to resolve the issue, restore system

health, and minimize the impact on users. Given that an event encompasses the entire process from failure detection to recovery, while containing rich semantic information, it plays a pivotal role in system operation and maintenance.

As indicated in Fig. 7, current research primarily explores the application of event data in fault understanding and root cause analysis. (1) Fault understanding encompasses activities such as detecting system anomalies, correlating events, classifying tickets, aggregating events, and summarizing incidents (Sec. 5.3.1). These processes aim to provide actionable insights for on-call engineers, corresponding to the detection and triage stages of the event lifecycle. (2) Root cause analysis, on the other hand, involves identifying the fundamental causes of system failures and proposing mitigation strategies, corresponding to the diagnosis and mitigation stages of the event lifecycle (Sec. 5.3.2). Traditional event analysis relies heavily on on-call engineers, demanding significant expertise and labor, and is susceptible to errors. While automated techniques exist, their adoption remains limited due to insufficient model performance, dependence on rigid rules, and poor generalization. To address these challenges, researchers have introduced Transformer-based models, particularly large language models, to harness their advanced linguistic capabilities for fault understanding and root cause analysis in event data.

5.3.1 Fault Understanding. Fault understanding primarily pertains to the detection and triage stages of the event lifecycle. By employing techniques such as anomaly detection [138], event correlation [139], event aggregation [60, 61], ticket classification [140], incident summarization [62], and expertise consolidation [63, 64], event data is preprocessed to enable more efficient root cause analysis and facilitate quicker failure recovery.

Transformer-Based. Traditional anomaly detection techniques often focus on identifying anomalous patterns from sequential data but tend to neglect the semantic information embedded in individual events. To address this limitation, CAT [138] introduces a Transformer-based framework designed to capture rich semantic content. The CAT framework begins with a content-awareness layer that extracts semantic patterns from the events. Subsequently, the encoder processes the sequence of event representations, producing a contextualized encoding, which is then decoded. Additionally, an auxiliary token is prepended to the sequence, functioning as a representation of the overall sequence state. The training process is guided by a generative objective aimed at two tasks: long-sequence forecasting and event sequence anomaly detection. These objectives are optimized using a one-class objective function, ensuring robustness in identifying anomalies across sequences.

Transformer-Encoder-Based. Some techniques leverage the BERT architecture to extract semantic information from event data for downstream tasks. For instance, LinkCM [139] employs semantic embeddings to link customer-reported tickets with system incidents, thereby streamlining the diagnostic process. Similarly, iPACK [60], an incident-aware method for aggregating duplicate tickets, combines failure information from both the customer side (*i.e.*, tickets) and the cloud side (*i.e.*, incidents), leveraging BERT to generate semantic embeddings as vectors. TixFusion [61] also leverages BERT to encode knowledge triples, after which a density-based clustering algorithm is applied to effectively group duplicate tickets from defect reports in mobile operating systems. Notably, in TixFusion, large language models are employed to retrieve knowledge triples from the knowledge base and to generate human-readable explanations for ticket clusters. These efforts mark one of the first attempts to apply Transformers in mobile operating systems. In contrast, Ticket-BERT [140] goes a step further by integrating a fully connected layer on top of its BERT-based embedding module to effectively classify incident tickets.

Transformer-Decoder-Based. Other techniques investigate the application of GPT-style models for event summarization. Oasis [62] highlights the limitations of directly inputting descriptions into large language models, such as inefficiency and potential token limit issues. Therefore, Oasis employs a multi-step process to generate effective

summaries. The method begins by identifying relevant incidents through three types of linkage analysis to evaluate the impact of outages. Next, domain-specific text processing is applied to filter noise and prioritize incidents based on their severity. Finally, the processed context is fed into fine-tuned GPT-3.5 or GPT-4 models, which generate concise, human-readable summaries. This technique has been successfully deployed at Microsoft, demonstrating its practical efficacy.

(Large) Pre-Trained Model-Based. Effective consolidation and application of expertise are critical for enhancing fault understanding. To address the challenges introduced by failure-prone software changes, SCELM [63] presents a change assessment framework that leverages historical knowledge and large language models to generate detailed analysis reports, which identify anomalies, classify failure types, and uncover root causes. SCELM unifies the phases of the software change lifecycle into a single pipeline, thereby streamlining the assessment management and enhancing the overall efficiency. While the aforementioned techniques aim to replace human engineers with Transformers, Hamadanian et al. [64] argue that large language models cannot fully substitute human experts. Instead, it introduces a collaborative framework where LLMs function as assistants to engineers, supporting various stages of event management. These stages include hypothesis formation, hypothesis testing, and mitigation planning. In this framework, the hypothesis former generates potential explanations to assist operators in pinpointing the root cause. The hypothesis tester then suggests steps to validate these explanations systematically. Finally, the mitigation planner leverages the validated hypothesis to develop a mitigation plan, providing actionable insights to address the issue effectively.

5.3.2 Root Cause Analysis. Root cause analysis necessitates a comprehensive integration of natural language text and domain-specific knowledge. By extracting failure-related information from event data, it identifies the root cause and recommends actionable mitigation steps [5, 35, 66, 141–144]. Additionally, it should, where feasible, offer insights into similar historical cases associated with the predicted root cause [65, 141, 145], along with human-readable explanations [146], a transparent reasoning process [146], and an assessment of confidence level [145].

Transformer-Encoder-Based. ICA [141] employs multiple BERT models, including BERT-base, BERT-large, DistilBERT, and RoBERTa, each fine-tuned on open-domain extractive question-answering datasets (*e.g.*, SQuAD [147]) and standard summarization datasets (*e.g.*, CNN-DailyMail [148]). The fine-tuned models are subsequently used to extract root causes and solutions from historical incident reports, which serve as the foundation for building a knowledge graph. In cases of failure, information retrieval techniques are applied to diagnose the issues.

Transformer-Decoder-Based. Similar to Oasis [62] for event summarization, fine-tuning GPT-style models has also become a prevalent approach for performing root cause analysis on event data. A notable example is Ahmed et al. [35], which conducted a rigorous and large-scale study at Microsoft involving over 40,000 incidents. By leveraging fine-tuned GPT-3.x models, Ahmed et al. effectively identified root causes and proposed mitigation measures for these incidents. Through experimental validation and the active involvement of production incident owners, the study highlighted the significant promise of large language models for improving incident management practices.

(Large) Pre-Trained Model-Based. Fine-tuning models requires substantial data and computational resources, with performance heavily reliant on the quality of fine-tuning. To bypass these limitations, some researchers focus on prompt engineering, a gradient-free alternative. RCACopilot [146] combines expert rules with the linguistic capabilities of large language models to develop an innovative framework for root cause analysis. After an incident occurs, RCACopilot applies specific rules to filter information relevant to the failure and then uses GPT-4 to synthesize and condense contextual data into concise, actionable insights. The processed data is subsequently re-entered into GPT-4 to classify potential root causes and deliver explanatory details. Notably, RCACopilot utilizes a chain-of-thought prompting

strategy to guide the LLM through intermediate reasoning steps, enhancing its effectiveness in predicting incident categories. Additionally, Zhang et al. [65] propose in-context example-based prompting, incorporating historical incident summaries and root causes as examples in prompt templates to steer large models toward identifying potential root causes. This technique significantly improves correctness and readability while mitigating the substantial computational and maintenance burdens of fine-tuning.

Some techniques adopt a ReAct-style paradigm [149] to enable autonomous agents, where reasoning, tool invocation, and decision updates are performed iteratively based on observations. Specifically, RCAgent [66] introduces a self-governing framework that integrates data collection and analysis tools, comprising an expert agent and a controller agent. The expert agent is responsible for acquiring and processing multimodal data, while the controller agent coordinates the expert's activities and synthesizes results. In consideration of privacy-sensitive industrial applications, RCAgent employs an internally hosted model and has been seamlessly integrated into Alibaba Cloud's issue discovery and diagnosis pipeline. ITBench [144] further evaluates this paradigm in practical settings, including site reliability engineering, compliance and security operations, and financial operations. Its agents interact with the environment through tool invocation and state feedback, demonstrating the applicability of such agent designs across diverse IT automation tasks. OpenRCA [142] implements a similar agent for root cause identification, but differs from RCAgent and ITBench that utilize predefined tool lists. Instead, it relies on the agent to synthesize and execute Python code for telemetry retrieval and analysis. While this design improves flexibility, it also imposes stronger requirements on the model's code generation capability. Complementary to reactive interaction, some techniques emphasize explicit planning capabilities by incorporating standard operating procedures (SOPs). For example, FlowXpert [5] extends the agent-driven troubleshooting paradigm by automating workflow orchestration in large-scale cloud datacenters. It builds a domain-specific knowledge base centered on incident-aware nodes and implements a co-evolving dual-agent system, consisting of a Planner and a Scorer. The Planner generates troubleshooting workflows, while the Scorer delivers multi-dimensional feedback; the two agents are optimized through reinforcement learning, enabling iterative improvements in workflow generation. Deployed in real-world operations at Huawei Cloud, FlowXpert contributes a lot to both on-call engineers and AI executors. Moreover, Flow-of-Action [143] introduces an SOP-enhanced multi-agent framework, where multiple specialized agents collaborate under the guidance and constraints of matched or generated SOPs. This represents a hybrid design that bridges flexible ReAct-style reasoning and structured operational knowledge. Additionally, LLM-based root cause analysis faces a critical dilemma: while blindly trusting AI introduces significant risks, disregarding its outputs forfeits its potential benefits. To address this challenge, LM-PACE [145] employs an LLM as a black-box generator of root cause hypotheses and concurrently utilizes the model to evaluate the confidence of these predictions. Specifically, it applies semantic vector analysis to identify historical events and root causes analogous to the current incident. The relevant information is then incorporated into the prompts to facilitate an assessment of the confidence in its current predictions.

5.3.3 Other Targets. Beyond fault understanding and root cause analysis, certain Transformer-based techniques emphasize additional dimensions of event management. For instance, to resolve incidents efficiently, on-call engineers are required to analyze telemetry data through domain-specific language (DSL) queries. KQL is a DSL employed for incident management in a large-scale cloud management system at Microsoft. To automate the recommendation of KQL queries and enhance the efficiency of incident management, XPert [150] has developed an end-to-end framework. The process begins by retrieving and integrating historical incidents resembling the new incident ticket into the prompt. Subsequently, the LLM generates tailored KQL queries. Notably, XPert introduces the Xcore metric, which

holistically assesses the quality of the generated queries through syntax and semantic check, sub-component matching, and output-schema consistency.

Table 7. Summary of Transformer-based models for event analysis. We classify the techniques based on targets, principles (TF represents *Transformer-Based*, TFE represents *Transformer-Encoder-Based*, TFD represents *Transformer-Decoder-Based*, LPM represents *(Large) Pre-Trained Model-Based*), approaches (TAB represents *Transformer-Architecture-Based*, PE represents *Prompt Engineering*, FT represents *Fine-Tuning*, FM represents *Foundation Models*), and core methods. Additionally, we list the capabilities or benefits of the Transformer (SEQ represents *Sequence Modeling*, PARALLEL represents *Parallelization and Non-Recurrent Dependencies*, PTKL represents *Pre-Training and Knowledge Utilization*, ICL represents *In-Context Learning*, TF represents *Transfer Learning and Domain Adaptation*, LANG represents *Language Capacity*).

Target	Technique	Principle				Approach				Core Methods	Capabilities or Benefits					
		TF	TFE	TFD	LPM	TAB	PE	FT	FM		SEQ	PARALLEL	PTKL	ICL	TF	LANG
Fault Understanding	CAT [138]	✓				✓				Content-Awareness Layer + Generative Inference + One-Class Objective	✓	✓				✓
	LinkCM [139]		✓						✓	BERT + Text Encoding + Feature Combination + Transfer Learning			✓		✓	
	iPACK [60]		✓						✓	BERT + Alert Parsing + Graph Profiling + Attentive Interaction Layer			✓			
	TixFusion [61]		✓				✓		✓	BERT + GLM + Multi-Round Retrieval + DBSCAN			✓	✓		✓
	Ticket-BERT [140]		✓					✓		BERT + Prompt-as-Prefix + Active Learning	✓		✓		✓	
	Oasis [62]			✓				✓		GPT-3.x + Heuristic Lookup + Domain-Specific Text Processing + Fine-Tuning			✓	✓	✓	✓
	SCELM [63]				✓		✓			Multimodal Processing + RAG			✓	✓		✓
	Hamadanian et al. [64]				✓		✓			GPT-4 + Prompt Engineering			✓	✓		✓
Root Cause Analysis	ICA [141]		✓					✓		Fine-Tuned BERT + Causal Knowledge Graph + Retrieval Based RCA			✓		✓	✓
	Ahmed et al. [35]			✓				✓		GPT-3.x + Fine-Tuning			✓	✓	✓	✓
	RCACopilot [146]				✓		✓			GPT-4 + CoT + Nearest Neighbor Search + Few-Shot			✓	✓		✓
	Zhang et al. [65]				✓		✓			GPT Families + RAG			✓	✓		✓
	RCAgent [66]				✓		✓			Multi-Agent System + Tool Invocation + Trajectory-Level Self-Consistency			✓	✓		✓
	ITBench [144]				✓		✓			ReAct + Reflection + LLM-as-a-Judge + Tool Invocation			✓	✓		✓
	OpenRCA [142]				✓		✓			ReAct + Code Generation			✓	✓		✓
	FlowXpert [5]				✓			✓		Ontology-Enhanced Knowledge Base + Multi-Agent System + RL			✓	✓	✓	✓
	Flow-of-Action [143]				✓		✓			GPT Families + ReAct + Multi-Agent System + SOP Enhancement + Tool Invocation			✓	✓		✓
	LM-PACE [145]				✓		✓			GPT Famalies + RAG + Confidence Estimation			✓	✓		✓
Other	XPert [150]				✓		✓			GPT Famalies + RAG + Query Generation + Post-Processor			✓	✓		✓

5.3.4 Summary. In Tab. 7, we summarize the Transformer-based models for event analysis, providing information on Targets, Principles, Approaches, Core Methods, and Capabilities.

The aforementioned techniques reveal that the core ideas of Transformer-based approaches to event analysis contain three levels: feature extractors, copilots, and autonomous agents. First, several techniques [60–62, 138–140] turn to Transformers for semantic information extraction, which adopt specific practices such as architecture design, pre-trained models, and fine-tuning. Second, Transformers increasingly function as copilots in various techniques [35, 63–65, 141, 145, 146]. In these cases, large pre-trained models are integrated with contextual knowledge and representative examples to support diverse tasks, providing valuable insights to on-call engineers. Finally, some techniques [5, 66, 142–144] extend Transformers into autonomous agents capable of not only executing tasks but also interacting dynamically with real-world environments. The ultimate objective is to develop self-governing agents for IT operations, enabling more efficient and intelligent automation.

In the event domain, Transformers navigate the entire event lifecycle, *i.e.*, detection, triage, diagnosis, and mitigation, demonstrating the unique capabilities in fault understanding and root cause analysis. On the one hand, effective fault understanding relies on semantic information extraction. The Transformer architecture, with its strengths in *Sequence Modeling, Parallelization and Non-Recurrent Dependencies*, is widely utilized to model event sequences and allocate attention to critical semantic content [138, 140]. Additionally, LLMs' *Pre-Training and Knowledge Utilization* and *Transfer Learning and Domain Adaptation* capabilities contribute to the smooth switching of semantic understanding from the pre-trained corpus to the domain-specific event data [60–64, 139]. On the other hand, root cause analysis primarily benefits from Transformers' capabilities of *In-Context Learning* and *Language Capacity*. By integrating contextual examples, large pre-trained models can identify potential root causes, suggest mitigation steps, generate

interpretations, and provide confidence estimates [5, 35, 65, 66, 141–146, 150]. This collaboration between on-call engineers and Transformer-based copilots or autonomous agents significantly accelerates the decision-making process, enhancing the efficiency of IT operations.

Unlike logs and metrics, which outline system status and signal failures indirectly, events provide direct, natural language descriptions of the incidents from the cloud side or tickets from the customer side. These descriptions typically include high-level attributes such as symptoms, impact scope, and severity. The rich semantic nature of events creates an ideal application scenario for Transformers, particularly large pre-trained models. By analyzing event data, Transformers develop a comprehensive, in-depth, and human-centric understanding of failures, enhancing the ability to localize root causes and suggest effective solutions. This, in turn, provides valuable insights to on-call engineers.

6 Evaluation and Benchmark for Transformers in AIOps Domain

High-quality benchmarks offer comprehensive experimental scenarios and evaluation standards that drive algorithmic innovation and technological advancement, fostering field growth. Similar to other data-driven fields, benchmarks also play a crucial role in the context of Transformers in the AIOps domain. We categorize key benchmarks into three types based on dataset forms: offline datasets (Sec. 6.1), standardized question banks (Sec. 6.2), and real-time simulations (Sec. 6.3). The following sections provide a detailed analysis of these benchmarks, including subfields, dataset construction processes, scales, experimental perspectives, evaluation strategies, and other key insights.

6.1 Offline Dataset

Offline datasets form the foundation of static evaluations under controlled conditions. Given access to relevant data, the effectiveness of Transformers could be assessed across diverse tasks, including time-series forecasting [7], multi-task log analysis [151, 152], and root-cause analysis based on multimodal telemetry [142], *etc.* Furthermore, large-scale, high-quality datasets constitute essential resources for both pre-training and fine-tuning stages.

6.1.1 CloudOps. Due to the scarcity of publicly available large-scale datasets in time series, systematic studies on the impact of pre-training and model scaling remain challenging. Consequently, time series research has fallen behind fields like natural language processing (NLP) and computer vision (CV) in the era of pre-training and transfer learning. To fill in this gap, the CloudOps [7], introduced by Salesforce AI Research and Singapore Management University, provides a valuable resource. It consists of three large-scale time series forecasting datasets, with the largest containing billions of observations. These datasets cover metrics such as CPU and memory utilization percentages, originating from Azure virtual machines, Borg clusters, and Alibaba clusters, which reflect cloud service quality. During the data preprocessing phase, duplicate timestamps are removed, missing values are imputed, and time series are filtered based on specific characteristics, with timestamps adjusted. The datasets are then split into pre-training and train-test sets to support various stages of model training and evaluation.

The study proposes a generic Transformer architecture, pre-trained as a powerful zero-shot forecaster. It features a probabilistic head based on a simple Student-T distribution, RoPE for positional encoding, and scalability across model sizes and training durations, all aimed at enhancing prediction accuracy without extensive hyperparameter tuning. Comprehensive benchmark tests compare the zero-shot forecaster with naive methods (*e.g.*, AutoTheta [153, 154], VAR [155]), deep learning models (*e.g.*, Autoformer [29], PatchTST [120]), and pre-training methods (*e.g.*, TS2Vec [156], CoST [157]). Point forecasting performance is measured using the symmetric mean absolute percentage error (sMAPE), while probabilistic forecasting is evaluated with the continuous ranked probability score (CRPS). When pre-trained, the

proposed architecture outperforms the aforementioned baselines across the three large-scale datasets. Moreover, further investigation into scaling behavior shows performance improvements as both dataset size and model parameters scale. Additionally, the billions of observations offer a large-scale foundation for evaluating the aforementioned Transformers in time-series forecasting [29, 31, 32, 56, 57, 118–123, 127–129].

6.1.2 Loghub-2.0. Loghub-2.0 [151], developed by The Chinese University of Hong Kong, addresses the limitations of existing log datasets in both scale and representativeness. The dataset comprises logs from 14 real-world software systems, including distributed systems, supercomputers, and operating systems. Each subset contains approximately 3.6 million log messages on average, representing a 1875-fold increase in size compared to the widely used Loghub-2k [108]. Data annotation follows a multi-stage pipeline: preprocessing, grouping, template annotation, template matching, and refinement. Each log message is labeled with its corresponding template and parameter fields, supporting structured parsing.

This benchmark offers a diverse dataset that captures both common and rare log patterns, enabling robust and scalable evaluation of log parsing techniques under realistic conditions [36, 51, 102–105]. Beyond data coverage, the benchmark facilitates fine-grained evaluation, particularly for infrequent templates and parameter-rich messages. Specifically, the study introduces a template-level metric, the F1-score of Group Accuracy (FGA), which complements message-level metrics such as Group Accuracy (GA) and Parsing Accuracy (PA) by mitigating sensitivity to imbalanced template distributions. The study also presents a systematic reassessment of 15 log parsers, covering both statistical and semantic techniques. The results reveal substantial performance degradation, particularly in handling rare or complex logs. By combining a high-quality annotated corpus with a rigorous evaluation protocol, Loghub-2.0 sets a new standard in log parsing and lays the foundation for future evaluation and refinement of the aforementioned Transformer-based parsers.

6.1.3 LogEval. LogEval [152], developed by Nankai University, Tsinghua University, and Huawei, is a benchmark suite for evaluating large language models in log analysis. It covers four key tasks, including log parsing, anomaly detection, fault diagnosis, and summarization, based on a collection of 4,000 real-world log entries sourced from diverse platforms such as LogHub [108], LogPub [158], LogPAI [159], Alibaba Cloud, China Mobile datasets, and LogSummary [160]. Each task is paired with 15 diverse prompts in both Chinese and English to reflect practical variability and reduce prompt-specific bias. The benchmark incorporates both objective and subjective evaluation formats. Objective tasks are scored using accuracy and F1-score, while subjective tasks are assessed by semantic similarity, ROUGE, and task-specific metrics such as edit distance and summarization accuracy. In addition, to evaluate model efficiency, the benchmark reports average inference time and token count.

Compared to LogHub 2.0, which focuses primarily on parsing and structured data extraction, LogEval covers the full lifecycle of log data in IT operations. This broader scope supports a more comprehensive evaluation of Transformers for log analysis [30, 36, 51–55, 96, 97, 102–105, 107, 109–111]. The benchmark tests examine the performance of eighteen leading large language models, including GPT-4, LLaMA2, and ChatGLM, under both zero-shot and few-shot settings. The results show that GPT-4 consistently outperforms others, particularly in parsing and fault diagnosis. LLaMA2-70B exhibits strong zero-shot performance in anomaly detection. However, most models struggle with anomaly detection in few-shot scenarios, highlighting the limitations of general-purpose LLMs and the need for anomaly-aware Transformers [30, 53–55, 96, 97, 109–111].

6.1.4 OpenRCA. The lack of large-scale, real-world datasets for root cause analysis remains a major barrier to evaluating the related capabilities of large language models. To bridge this gap, OpenRCA [142], conducted by Microsoft, the Chinese University of Hong Kong, and Tsinghua University, presents the first publicly available benchmark for root cause analysis. It comprises 335 failure cases and over 68GB of multi-modal telemetry data, encompassing metrics, logs, and traces. Additionally, the datasets span a broad spectrum of systems, including telecom databases, banking applications, and e-commerce platforms. This diversity reflects the inherent complexity of modern production environments.

Notably, OpenRCA enables integrated evaluation across heterogeneous telemetry modalities, supporting the assessment of both single-source diagnostic performance and cross-modal correlation reasoning. Therefore, it could serve as a testing ground for applying Transformers to root cause analysis [35, 55, 59, 65, 66, 145, 146]. Diagnostic accuracy, defined as the proportion of correctly identified components, timestamps, and root causes, is adopted as the primary evaluation metric. The study further proposes a multi-agent framework equipped with program synthesis and execution capabilities. A controller agent is responsible for planning the reasoning steps, while an execution agent handles heterogeneous telemetry data by generating and executing Python code. This modular division enhances scalability and alleviates the limitations imposed by context window constraints. Experimental results demonstrate that execution-based approaches substantially improve diagnostic performance, especially when paired with large models optimized for code generation. Nonetheless, the overall accuracy remains low, particularly in complex scenarios characterized by multiple root cause elements.

6.2 Standardized Question Bank

Question banks, composed of various standardized formats including multiple-choice, short-answer, and fill-in-the-blank questions, are typically designed to directly assess the potential of large language models in IT operations. Compared with offline datasets that evaluate task-specific performance using corresponding data sources, these benchmarks tend to place greater emphasis on evaluating the internal knowledge and domain expertise embedded within the models.

6.2.1 MSQA. MSQA [161] is a cloud-domain QA dataset introduced by the Microsoft research team, focusing on Microsoft products and IT technical issues encountered by customers. It contains 32K QA pairs compiled from textual responses collected on the Microsoft Q&A forum between October 2019 and May 2023, covering a wide range of Microsoft technologies and products, including Azure and Microsoft 365. During data collection, only single-turn QA pairs labeled as "Accepted" are retained, while responses containing images are discarded to maintain focus on textual content. Additionally, post-processing is applied to remove unwanted noise, such as user-related information and irregular decorative symbols added by users, to ensure data quality.

The benchmark tests show that large language models, when utilizing domain knowledge from a smaller language model through instruction tuning, outperform those that rely on retrieval-based methods like BM25 [162] and Dense Passage Retrieval (DPR) [163]. In the experiments, a wide range of metrics are used to evaluate long-form generated answers, including Lexical-Overlap-Based Metrics (BLEU, ROUGE-1, ROUGE-2, ROUGE-L), Semantic-Overlap-Based Metrics (BERTScore), Keyword/Span-Hit-Rate (KHR), Can-Answer-Rate (CAR), LLM-Based Metrics, and Human Evaluation. As the results suggest, lexical-overlap-based metrics are not well-suited for evaluating long-form LLM-generated answers, particularly in the cloud domain. This is because the generated answers show poor correlation with grounded human-written answers, demonstrating limited variations across methods and generally lower scores.

6.2.2 NetEval. Network operations (NetOps) encompass several common application scenarios, including network monitoring, device management, troubleshooting, and performance optimization, all of which are essential components

of IT operations. To assess the capabilities of large language models in commonsense knowledge and inference within the NetOps domain for further advancement, Zhongguancun Laboratory develops the NetEval dataset [164]. This dataset consists of 5,269 multiple-choice questions and 463 non-multiple-choice questions, such as fill-in-the-blank and question-answering tasks, which are sourced from a variety of documents, including certification exam question banks, network textbooks, and Wikipedia articles. The construction process involves a combination of rule-based filtering, input-output extraction with GPT-4, and manual verification.

NetEval evaluates six representative model families, with approximately three models tested from each family (*e.g.*, GPT-3.5-turbo and GPT-4 for the GPT series). The evaluation primarily uses accuracy for multiple-choice questions, while BLEU and ROUGE scores are employed to assess the quality of responses to non-multiple-choice questions. A more detailed evaluation is conducted for certain models based on their parameter scale (*e.g.*, LLaMA2 7B, 13B, and 70B). The study also systematically examines the impact of different prompting strategies, including zero-shot, few-shot, and Chain-of-Thought (CoT) approaches. The results show that both increasing model size and adopting a few-shot setting improve response accuracy. Additionally, for relatively easier multiple-choice questions, GPT-4 achieves an accuracy of 81% in the few-shot setting, making it the only model capable of passing NetOps-related certification exams, while other models exhibit significantly lower accuracy. For non-multiple-choice questions, however, all models perform poorly, with BLEU and ROUGE scores remaining below 0.3. These experimental results suggest that LLMs tailored for the NetOps domain still have a long way to go.

6.2.3 DevOps-Eval. DevOps-Eval [165], developed by the CodeFuse team, is a comprehensive benchmark specifically designed to evaluate large language models in the DevOps domain. The benchmark covers eight key sub-domains that span the entire software development lifecycle: planning, coding, building, testing, releasing, deploying, operations, and monitoring. The dataset consists of 7,486 multiple-choice questions in a single-choice format, with 2,840 AIOps-related samples addressing scenarios such as log parsing, time series anomaly detection, time series classification, time series forecasting, and root cause analysis. Additionally, 1,509 ToolLearning samples are included, covering 59 domains and 239 tool categories. The dataset is derived from various sources, including multiple-choice questions from professional certification exams, AIOps-related samples obtained from open-source datasets, scenario-based questions generated by ChatGPT using relevant keywords, and synthetic data produced through data simulation.

The experiments evaluate the performance of various general-purpose and domain-specific large language models in the DevOps domain, exploring the strengths and limitations of these models. The evaluation primarily focuses on multiple-choice question accuracy, with particular emphasis on zero-shot and one-shot performance in the context of AIOps-related queries. The results show that, in most cases, the models perform better in the one-shot setting than in the zero-shot setting. Among the zero-shot models, Qwen-14B-Base achieves the highest average accuracy, while in the one-shot setting, DevOpsPal-14B-Chat outperforms all other models. Overall, the models exhibit varying levels of performance across different AIOps tasks, highlighting the adaptability differences of each model to specific tasks, while providing valuable insights for selecting the most appropriate model for particular use cases.

6.2.4 Owl-Bench. Owl-Bench [166], developed by Beihang University, provides a benchmark for the operation and maintenance(O&M) domain to measure LLMs' capabilities, which covers nine O&M-related domains (information security, application, system architecture, software architecture, middleware, network, operating system, infrastructure, and database), enabling systematic assessment of how well models understand specialized operational knowledge across diverse areas. This comprehensive structure reflects the diversity of LLMs' capabilities in a hierarchical manner. As a bilingual benchmark, Owl-Bench includes 317 QA pairs and 1,000 multiple-choice questions. The dataset is sourced from

skill assessment exams for IT operations professionals, supplemented by approximately 200 questions contributed by experts. Notably, the dataset retains its real-world relevance, as it originates from scenario-based examination questions, with no modifications or expansions made using large language models.

The experiment evaluates the performance of six language models on Owl-Bench, using two distinct evaluation methods for the QA questions. The first method, referred to as the single-score mode, involves GPT-4 assigning a numerical score between 1 and 10 to the LLM-generated responses. The second method, the pairwise-score mode, adopts a comparative approach in which GPT-4 evaluates the responses from two models to the same question and determines which model produces a higher-quality answer. For multiple-choice questions, accuracy is assessed across various O&M domains. Additionally, the study introduces a supervised fine-tuned model, trained on synthetic single- and multi-turn dialogues derived from seed data, which demonstrates the highest performance on QA questions. In contrast, ChatGPT achieves the best average score in multiple-choice questions.

6.2.5 OpsEval. OpsEval [167], developed by Tsinghua University, is a comprehensive benchmark suite that includes an Ops-oriented evaluation dataset, a corresponding evaluation benchmark, and a specialized QA evaluation method designed for Ops. The research team compiles data from various open AIOps communities, incorporating materials such as company documents, certification exams, and operation textbooks. These data sources are globally recognized and reviewed by Ops collaborators. The collected data undergoes several stages of processing, including deduplication, dependency filtering, question categorization, and manual review. Finally, the obtained dataset consists of 1,736 question-answer pairs and 7,184 multiple-choice questions, available in both Chinese and English, spanning eight distinct task scenarios and incorporating three ability levels. The 20% of the dataset is released to the public, while the remaining 80% is withheld to safeguard against potential test set leakage, which forms an up-to-date leaderboard.

OpsEval is designed to assess the performance of LLMs in IT operations, providing valuable guidance for selecting the most suitable models within this field. The benchmark evaluates 21 latest LLMs and introduces 4 reasoning strategies: self-consistency (SC), chain-of-thought (CoT), a combination of CoT and SC, and few-shot in-context learning. These strategies enable a more thorough assessment, going beyond simple, naive responses. For multiple-choice questions, OpsEval uses accuracy as the primary evaluation metric. For question-answering tasks, two distinct evaluation metrics are employed. The first is based on lexical overlap, incorporating ROUGE and BLEU scores, while the second focuses on semantic similarity, utilizing LLM-based evaluation (FAE-Score) and expert assessment (Expert-Evaluation), which involves fluency, accuracy, and supporting evidence. Experimental results show that closed-source models, such as GPT-4 and GLM-4, consistently outperform other models. Additionally, few-shot in-context learning and chain-of-thought reasoning significantly improve model performance.

6.3 Real-Time Simulation

Through real-time simulation, the agent under evaluation could interact dynamically with the system to perform a series of operations, including observation, reasoning, decision-making, tool invocation, and result analysis. This type of benchmark effectively serves as a sandbox environment for agents to practice and evolve.

6.3.1 AIOPSLAB. The AIOPSLAB [8], proposed by UIUC, UC Berkeley, Microsoft, and IISc, features a comprehensive framework for automating the entire end-to-end evaluation process of AIOps solutions, which includes deploying services, injecting faults, generating workloads, orchestrating agent cloud interaction, and analyzing results. In the AIOPSLAB playground, AI agents could observe, act, and receive feedback, facilitating dynamic and interactive problem-solving within complex environments. These functionalities are supported by an Agent-Cloud Interface, which consists

Table 8. Summary of evaluation and benchmark for Transformers in AIOps. We identify the benchmarks based on dataset formats, scale, domains, and evaluation metrics.

Benchmark	Dataset Formats and Scale	Domain	Evaluation Metrics	Link
CloudOps [7]	Billions of Time Series Observations	Cloud Operations	sMAPE + CRPS	[171]
Loghub-2.0 [151]	50M Log Messages	Log Analysis	GA + PA + FGA + FTA	[172]
LogEval [152]	4K Log Messages	Log Analysis	Accuracy + F1 + Semantic Similarity + ROUGE + Edit Distance + Inference Time + Token Length	[173]
OpenRCA [142]	335 Failure Cases With Over 68 GB of Telemetry Data	Root Cause Analysis	Proportion of Correctly Identified Root Cause Elements	[174]
MSQA [161]	32K QA	Operation Issues	Lexical-Overlap-Based + Semantic-Overlap-Based + KHR + CAR + LLM-Based + Human Evaluation	[175]
NetEval [164]	5,269 Multi-Choice, 463 Filling-Blanks and QA	Network Operations	Accuracy + BLEU + ROUGE	[176]
DevOps-Eval [165]	7,486 Multi-Choice	Development and Operations	Accuracy	[165]
Owl-Bench [166]	1K Multi-Choice, 317 QA	Operation and Maintenance	LLM as a Judge (Single-Score Mode, Pairwise-Score Mode)	[177]
OpsEval [167]	7,184 Multi-Choice, 1,736 QA	IT Operations	Accuracy + FAE-Score + Expert-Evaluation	[178]
AIOPSLAB [8]	AI Agent Playground Featuring a Pool of 48 Problems	Ops-Oriented Agents	Correctness + Time + Steps + Tokens	[179]
ITBench [144]	Runtime Environment with 102 Real-World Scenarios	IT Operations	Pass@1 + NTAM + MTTD + MTTR + TTP + F1 + Ranking	[180]

of a set of APIs enabling agents to obtain multimodal data from the cloud's observability tools and make meaningful progress towards operational objectives. To evaluate the performance of LLM-driven agents within the AIOps domain, the framework incorporates a problem pool featuring 48 problems (*e.g.*, RevokeAuth, PodFailure, NetworkLoss) spanning four levels of complexity: detection, localization, root cause analysis, and mitigation. Furthermore, the framework allows users to easily extend and create new problem instances.

Unlike offline datasets or question banks limited to static evaluations, AIOPSLAB highlights real-time interaction and adaptive decision-making, establishing a unique benchmark for agent-oriented AIOps solutions [5, 54, 64, 66, 142]. Through a comprehensive problem pool with multimodal telemetry and fault injection capabilities, it enables rigorous assessment of agents' performance in detecting, localizing, and mitigating failures under real-time conditions. In its evaluation, AIOPSLAB registers four AI agents: naive agents without fine-tuning or modifications using GPT-3.5-TURBO and GPT-4-TURBO, ReAct [149], an agent that integrates reasoning and acting in an interleaved manner, and FLASH [168], a cloud operation-specific agent. Additionally, non-LLM-based AIOps algorithms, such as MKSMC [169] for detection and PDiagnose [170] for localization, are also included as baseline comparisons. Regarding evaluation metrics, correctness is used to measure whether the agent successfully resolves the problems, while time (e.g., Time-to-Detect, Time-to-Mitigate) and steps are employed to assess the efficiency of problem resolution. The number of tokens generated by both agents and the environment serves as an indicator of the computational cost. The results demonstrate that FLASH achieves the best overall performance across all baselines. However, GPT-3.5-TURBO, despite being the fastest in task completion, achieves a poor accuracy of only 15.25%. Moreover, due to varying task complexities, even FLASH, the top-performing agent, shows performance limitations when tackling more challenging tasks, such as mitigation. Agent behaviors contributing to suboptimal performance, including unnecessary steps, data overload, invalid API usage, and false positive detection issues, are identified and discussed in detail. These insights pave the way for further improvement towards self-healing cloud systems.

6.3.2 ITBench. ITBench [144], conducted by IBM and UIUC, is a benchmarking framework and runtime environment for evaluating agents on practical IT automation tasks across multiple personas, including site reliability engineering (SRE), compliance and security operations (CISO), and financial operations (FinOps). Evaluation scenarios are formalized as structured tuples $\langle M, E, T, D \rangle$, encompassing scenario specifications, environment configurations, triggering events, and desired outcomes. ITBench includes 102 scenarios and supports both vertical expansion (*i.e.*, adding new scenarios) and horizontal expansion (*i.e.*, introducing additional personas), enabling broad coverage of IT operational tasks.

Similar to AIOPSLAB, ITBench drives agents to interact with the testbed environment, allowing end-to-end evaluation of their ability to solve real-world tasks. For SRE tasks, it adopts pass rate, a normalized topology-aware metric for

root cause and fault propagation analysis, as well as mean time to diagnosis (MTTD) and mean time to repair (MTTR). For CISO tasks, pass rate and time to process (TTP) are used to measure effectiveness and efficiency. For FinOps tasks, the evaluation considers pass rate, F1-score, and ranking metrics. The reported results indicate that current agents achieve limited performance on complex, multi-step scenarios, highlighting the gap between prototype capabilities and real-world operational demands.

6.4 Summary

In Tab. 8, we summarize the evaluations and benchmarks for Transformers in AIOps, highlighting key aspects including Dataset Formats and Scale, Domains, and Evaluation Metrics. The datasets used for evaluation can be classified into three categories based on their data format. First, the offline dataset [7, 142, 151, 152] consists of pre-collected and processed static data. This type of data is stable and reliable, making it ideal for training and testing models, particularly for tasks that require extensive historical data, such as time-series analysis. In contrast, real-time simulation [8, 144] creates real-world, dynamic scenarios, allowing the model under evaluation to interact with and respond to continuously changing data. This approach is particularly valuable for assessing the analytical and decision-making capabilities of AI agents in a live environment. Last but not least, standardized question banks [161, 164–167] evaluate the strengths and weaknesses of models through a set of structured questions, including multiple-choice, QA, and fill-in-the-blank formats. These question banks are versatile and applicable to a wide range of use cases, enabling quick construction, evaluation, sharing, and comparison. Their straightforward structure and evaluation methods make them the most common type of dataset in the field.

Despite recent progress in evaluation strategies, several gaps remain. First, current settings differ from real-world environments, as reflected in the limited coverage of system heterogeneity, data modalities, and task types, as well as the reliance on static datasets that cannot evolve to accommodate emerging data patterns. Moreover, operational problems are often simplified into a downstream task in offline datasets or a single QA instance in question banks, whereas real-world operations typically involve multi-step reasoning with progressively revealed information. Although real-time simulation brings evaluation closer to practice, simulated environments still fall short of production systems in scale and complexity, despite the substantial cost of building such infrastructures. Second, evaluation reliability may also be affected by noisy annotations, potential data leakage, prompt sensitivity, and subjective scoring mechanisms (*e.g.*, semantic matching or LLM-as-judge). These issues compromise the reproducibility of benchmarking results and hinder fair comparisons across different methods. Finally, most benchmarks emphasize one-shot task performance, while long-term operational stability and human-in-the-loop collaboration remain largely underexplored. These gaps highlight important directions for future benchmark design, including more realistic environments, more robust evaluation protocols, and better support for long-term scenarios.

7 Related Work

Surveys on Traditional Techniques in AIOps. AIOps serves as a cornerstone of modern operational practices, offering critical solutions to complex issues such as anomaly detection, root cause analysis, failure classification, impact evaluation, and system recovery. With sustained attention from researchers, an increasing number of techniques and methodologies have been proposed, and several surveys and empirical studies have been successfully published [1, 2, 13, 14, 37, 69, 181–188]. On the one hand, some surveys summarize and categorize research efforts tailored for specific tasks in AIOps. For instance, Gao et al. [181, 182] systematically classify nearly a decade of fault diagnosis and fault-tolerant techniques into four main approaches: model-based, signal-based, knowledge-based, and hybrid or

active methods. Similarly, Wong et al. [183] concentrate on fault localization techniques for single software systems, emphasizing methods that guide programmers to fault locations with minimal human intervention. Sol et al. [184] provide a comprehensive overview of root cause analysis techniques and offer guidance on selecting the optimal strategies based on factors such as scalability, time complexity, and other factors. Beyond root cause analysis, Soldani et al. [185] explore anomaly detection techniques for service-based cloud applications. Additionally, Zhang et al. [2] conduct a comprehensive survey on the failure diagnosis in microservice systems, covering root cause analysis and failure classification through multimodal data. Notaro et al. [1] examine the role of AIOps in failure management, summarizing application requirements and presenting quantitative evaluations of approximately 100 distinct techniques. Meanwhile, Remil et al. [37] provide a systematic review of the complete AIOps workflow for incident management, proposing a unified taxonomy and comparing existing studies to offer methodological guidance for research and practice. On the other hand, several surveys concentrate on research related to specific data modalities. For example, Garg et al. [186] evaluate and compare various multi-dimensional time series anomaly detection and fault diagnosis techniques on a unified dataset. Oliner et al. [187], He [69], and Arya et al. [188] provide research overviews focused on log data, while Li et al. [13] and Zhou et al. [14] highlight the significance of distributed tracing as an essential component of microservice infrastructure, particularly from an industrial practice perspective. Unlike these previous reviews, our study adopts a multimodal data perspective and systematically tracks the advancements in diverse tasks. Moreover, prior studies have not thoroughly updated or summarized the current state of AIOps research, especially the latest developments involving Transformers. Our work fills this critical gap.

Surveys on Transformers in AIOps. The Transformer architecture, along with its variants such as the BERT and GPT families, has been widely adopted across diverse fields due to its exceptional capabilities. It has gradually established itself as a foundational architecture, alongside CNN and RNN. Notably, the late 2022 emergence of ChatGPT has further intensified researchers' interest in large pre-trained models. Lin et al. [189] provide a comprehensive overview of the vanilla Transformer and its variants, discussing architectural modifications, pre-training strategies, and diverse applications. Zhao et al. [190] focus on large pre-trained language models based on the Transformer architecture, summarizing advancements across four key aspects: pre-training, adaptation tuning, utilization strategies, and capacity evaluation. Several surveys have explored the applications of Transformers in domains such as natural language processing [191–193] and computer vision [26, 194, 195]. However, in the AIOps domain, comprehensive surveys on Transformer-based techniques remain limited. Wen et al. [34] systematically review the use of Transformers in time series analysis, highlighting both their advantages and disadvantages. The study takes a dual approach: First, it examines various structural modifications to Transformers tailored for domain-specific challenges; second, it categorizes existing techniques into prediction, anomaly detection, and classification. Additionally, it empirically investigates how Transformers work, providing practical insights. Also, Su et al. [38] present a review of Transformers for time series forecasting and anomaly detection, highlighting their strengths and limitations in handling large-scale heterogeneous data, enabling real-time predictions, and identifying anomalies, while underscoring the need for cross-modal integration and model explainability. Le et al. [36] explore the effectiveness of large language models in log analysis, particularly under different prompting strategies. They also identify potential opportunities and challenges associated with employing ChatGPT for log analysis tasks. Similarly, Ahmed et al. [35] evaluate the performance of large language models in diagnosing root causes and recommending mitigation steps. Their study analyzes over 40,000 incidents from more than 1,000 cloud services at Microsoft, using six semantic and lexical metrics to demonstrate the models' potential in resolving cloud incidents. Moreover, Zhang et al. [3] review techniques for failure management in the era of large language models, from the perspective of multimodal data, task objectives, and LLM-based approaches.

Although relevant to our work, these studies suffer from several limitations (detailed in Sec. 1), including insufficient coverage and analysis of data modalities [3, 34–36, 38], a narrow scope of Transformers [3, 34–36, 38], the absence of structural principles [3, 34–36, 38], and inadequate discussion of methodological approaches [34–36]. In contrast, our study provides a systematic investigation of multimodal data, AIOps tasks, Transformers, and their interconnections reflected in the principles and approaches. We further distill the distinctive capabilities of Transformers, offering researchers a more holistic and in-depth understanding of related solutions. In addition, the summarized open resources and evaluation strategies are expected to advance the application of Transformers in AIOps further.

Quantitative Analysis of AIOps Techniques. Significantly, numerous empirical studies and evaluations have quantitatively examined research within the AIOps domain. For instance, Garg et al. [186] train 11 deep learning models on 7 multivariate time series datasets, assessing the effectiveness of various techniques for anomaly detection and diagnosis. Furthermore, this study reveals that the choice of scoring function often outweighs the importance of the underlying model by combining 10 models with 4 different scoring functions. Zhou et al. [14] conduct an empirical investigation using a trace visualization tool, ShiViz, to document the debugging process. Their study qualitatively and quantitatively demonstrates the efficacy of distributed traces in fault analysis, underscoring the potential for significant improvement over existing techniques. Arya et al. [188] evaluate multiple state-of-the-art causal inference techniques such as regression-based methods, independence testing-based approaches, and event models, using log data from the publicly available TrainTicket microservice benchmark [14]. Lyu et al. [196] perform a case study on two commonly studied AIOps applications, highlighting that time-based splitting of training and validation datasets significantly reduces data leakage compared to random splitting. Their findings also emphasize that regularly updating AIOps models is an effective strategy to address concept drift, albeit with additional computational overhead. As highlighted earlier, several works [35, 36, 104, 151, 152] investigate the potential of state-of-the-art large language models in cloud event management and log analysis through empirical studies. Such efforts provide valuable insights for researchers to intuitively compare performance, overhead, and other aspects across different techniques. However, they either overlook recent advancements in Transformers [14, 151, 186, 188, 196] or fail to provide a qualitative analysis of AIOps techniques leveraging multimodal data from the perspective of Transformers' unique capabilities [35, 36, 104, 152]. This challenge is precisely what our study aims to address.

8 Open Challenges and Future Directions

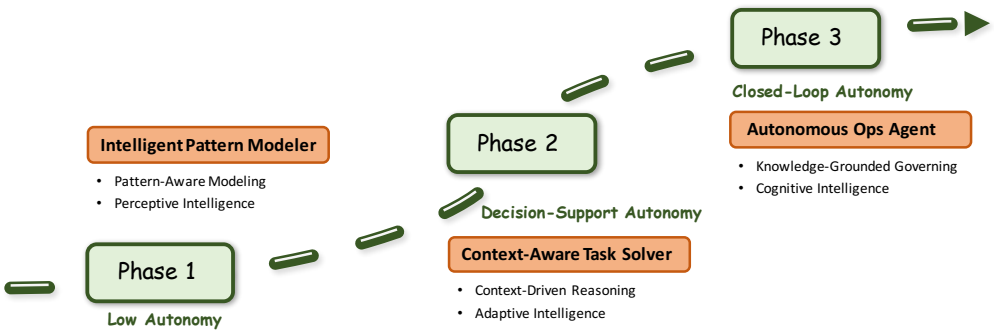

Fig. 8. Evolving Roles of Transformers in AIOps.

Building on the recent advancements and corresponding benchmarks outlined in previous sections, the role of Transformers in AIOps has undergone a notable evolution, as illustrated in Fig. 8. These roles represent a continuum of increasing capability, and their boundaries may therefore overlap in practice. We distinguish them primarily by the degree of operational autonomy, ranging from pattern analysis to decision support and autonomous action. (1) In their early applications, Transformers serve as intelligent pattern modelers [28–31, 52, 53, 58, 60, 61, 96, 97, 107, 109, 112, 118–121, 135, 136, 138–141], analyzing operational data to complete foundational tasks such as anomaly detection, failure classification, and root cause localization. They primarily extract patterns from historical data with limited contextual reasoning or decision-making capabilities. (2) Over time, Transformers have progressed into context-aware task solvers that integrate diverse sources of contextual information [32, 36, 50, 51, 55–57, 59, 62, 63, 65, 100, 102–105, 110, 111, 122–129, 145, 146, 150], such as observability data, configuration changes, service topology, and historical cases, to address more complex tasks like impact assessment and incident summarization. Compared with pattern modelers, they leverage richer context to perform reasoning in natural language and support operators' decision-making. However, Transformers at this stage remain modules within the operational workflow rather than end-to-end solutions. (3) Most recently, Transformers are emerging as autonomous agents in operations [5, 54, 64, 66, 142–144], interacting with environments and making decisions with a certain degree of autonomy. Beyond analyzing incidents and recommending mitigation strategies, these agents can execute actions within governed frameworks, thereby closing the operational loop.

This trajectory reflects not only the evolving demands of real-world deployments of AIOps systems, *i.e.*, progressing from perception to reasoning and ultimately to self-governing actions, but also highlights the expanding capabilities of Transformers. However, considerable challenges still lie ahead on the journey toward truly intelligent and autonomous operations. These include Hallucinations, Domain Knowledge, and Interpretability (Sec. 8.1); Generalization and Out-of-Distribution Data Handling (Sec. 8.2); Computational Costs and Path to Efficiency Optimization (Sec. 8.3); and Data Fusion and Model Collaboration (Sec. 8.4). Moreover, Sec. 8.5 recommends Best Practices in response to these challenges, whereas Sec. 8.6 looks ahead to Emerging Neural Architectures that may herald the next wave of AIOps evolution.

8.1 Hallucinations, Domain Knowledge, and Interpretability

The phenomenon of "hallucination" in large language models refers to the generation of inaccurate or nonsensical content, often deviating from the provided context or factual information [197]. This phenomenon poses significant challenges in the practical application of Transformers, particularly in rigorous industrial domains like AIOps. Unstable or imprecise model outputs can result in severe consequences, including misleading decisions, compromised system stability and security, and harm to economic interests and reputation. Mitigating such hallucinations to ensure the stability and accuracy of model outputs is crucial in effectively developing and applying Transformers in AIOps. Currently, few techniques address the hallucination issue in the AIOps domain. While some general-domain techniques, such as retrieval-augmented generation [198], fact-checking mechanisms [199], and adversarial fine-tuning [200], may help models resist inaccurate or nonsensical outputs, their effectiveness in AIOps requires further investigation.

From one perspective, domain knowledge equips the model with specific contextual information, enabling it to supplement business logic, professional details, historical experience, standard operating procedures, and other aspects not covered by the pre-trained corpora. This is particularly vital for knowledge-intensive tasks in AIOps. Fine-tuning allows the integration of domain knowledge into the model's parameters, fostering the development of specialized models. However, this approach demands significant expertise, high-quality data, and substantial computational resources. Consequently, some techniques bypass the relatively complex fine-tuning process. One such approach

involves constructing knowledge bases and combining them with retrieval-augmented generation, facilitating the accumulation and utilization of domain knowledge [5, 61, 141]. Additionally, as domain knowledge may evolve alongside the software or scenario, maintaining continuous updates is crucial for adapting to dynamic systems. Thus, ensuring the ongoing refresh of domain knowledge is a critical consideration when designing model fine-tuning processes or knowledge bases.

From another perspective, interpretable model outputs help build trust with on-call engineers by enabling them to understand the rationale behind the model's decisions and evaluate its reliability. This transparency makes it easier to accept the recommendations or correct erroneous outputs. Enhancing model interpretability can be achieved through various techniques. For instance, visualization techniques like heatmaps and attention maps highlight the areas or features the model focuses on [201, 202]; decision trees and rule extraction display the decision-making process directly [10]; backpropagation methods assess the influence of different layers in a neural network [203]; and simpler models can be trained to approximate the behavior of more complex ones [204, 205]. As Transformers grow in complexity, they increasingly become black boxes that are difficult to understand. By clarifying the effectiveness of these interpretability techniques within AIOps, we can facilitate their broader deployment and adoption.

8.2 Generalization and Out-of-Distribution Data Handling

In the AIOps domain, tasks are frequently accompanied by software evolution or scenario migration, including system iterations, scaling of microservice instances, and changes in online service business. These dynamics place high demands on a model's generalization capability, particularly its adaptability to system changes and portability across diverse tasks and scenarios. Traditional methods, such as meta-learning, incremental learning, continuous data collection, and periodic model updates, or even designing new model architectures, have been employed to tackle this challenge. However, these approaches often struggle to effectively and swiftly transfer to complex, evolving scenarios or tasks. In contrast, pre-trained models, which benefit from the rich knowledge captured from a large-scale corpus, have demonstrated strong generalization performance in natural language tasks [47, 101, 106, 206]. We anticipate that pre-trained Transformers, leveraging this extensive pre-learning, will exhibit similar generalization capabilities within the AIOps domain.

Theoretically, Transformers have the potential to internalize general knowledge during pre-training, embedding it into the extensive parameters to cover a wide range of scenarios and tasks. Additionally, fine-tuning with high-quality data can further refine these parameters to better align with the data patterns of specific cases. However, despite this theoretical promise, empirical evaluations suggest that current pre-trained models have not yet fully realized their generalization potential, especially in the AIOps domain. Specifically, in certain benchmarks, the model's performance under zero-shot and few-shot settings remains suboptimal [8, 142, 152, 164, 167]. This indicates that even a naive pre-trained model, or one based on reasonable extrapolation from similar cases, struggles to effectively handle out-of-distribution data. Even with fine-tuning, challenges like insufficient high-quality data or a lack of accumulated cases could hinder performance. As a result, additional data and customization may still be necessary. In practice, on-call engineers are particularly concerned with the model's ability to adapt to dynamic changes. Thus, key research directions for deploying Transformers in AIOps will include guiding the model to handle out-of-distribution data better, refining the fine-tuning process for specific tasks or scenarios, and improving the overall generalization capability.

8.3 Computational Costs and Path to Efficiency Optimization

Transformers have demonstrated exceptional performance across a broad spectrum of AIOps tasks, particularly in addressing complex challenges such as root cause analysis and event summarization. This effectiveness is primarily attributed to their deep network architectures and large parameter scales. However, Transformers are not a silver bullet for all AIOps solutions, as their advantages often come with substantial computational costs. In practice, the feasibility of Transformer-based techniques should be assessed from three perspectives: hardware requirements, training costs, and inference latency.

Hardware Requirements. Transformers typically rely on GPU acceleration due to the computational intensity of the self-attention mechanism and their large parameter sizes. Training and deploying such models often require high-memory GPUs or even multi-GPU clusters, particularly when processing long log sequences or massive observability data. In contrast, classical models such as CNNs and RNNs generally have smaller memory footprints and often need a single GPU or CPU-based environment. Therefore, in resource-constrained settings or edge deployment scenarios, classical models fundamentally offer a more practical option.

Training Costs. Training Transformers is typically more expensive than training classical architectures. In addition to the hardware requirements discussed above, large pre-trained Transformers usually require extensive datasets and prolonged training. For example, Microsoft's Oasis model [62], designed for fault understanding, is trained on large volumes of incident data collected from diverse cloud platforms and optimized through multiple training iterations. The entire pipeline, including data preprocessing, model training, and hyperparameter tuning, places a significant burden on the supporting infrastructure. By comparison, classical models generally require fewer resources to achieve reasonable performance. However, for tasks requiring strong generalization and semantic understanding, large pre-trained Transformers remain highly beneficial. Recent advances, such as parameter-efficient fine-tuning and transfer learning, have partially alleviated the training burden [53, 56, 110, 129]. These techniques retain much of the expressive power of Transformer architectures while reducing data and computational requirements, thereby enabling more accessible adaptation to novel tasks and operational environments.

Inference Latency. Inference efficiency is another important consideration for production AIOps systems, while the tolerance for latency varies across tasks. Transformers are often associated with longer inference times, limiting their applicability in latency-sensitive scenarios such as real-time anomaly detection or online alert filtering, where responses are expected within seconds. In such cases, classical models with simpler architectures typically provide faster inference and are therefore more suitable. However, not all AIOps tasks impose strict latency requirements. For example, root cause analysis is generally less time-sensitive but often requires more complex reasoning over observability data. In these scenarios, the stronger capability of Transformers can justify their higher cost. It is also worth noting that deploying end-to-end Transformers may be overkill in relatively simple scenarios, where smaller models or even traditional data analysis methods, possibly assisted by human experts, can already deliver acceptable performance at significantly lower computational cost [207, 208].

Improving the efficiency of Transformer-based AIOps solutions remains an important research challenge. In practice, the choice between Transformers and classical models involves trade-offs between performance and computational costs. The following checklist summarizes key recommendations that practitioners may use to assess feasibility.

- **Recommendation 1: Check resource availability.** When computational resources are limited (*e.g.*, CPU-only environments or small GPU budgets), classical models such as CNNs or RNNs are generally more practical due to their lower training costs and inference overhead.

- **Recommendation 2: Examine data scale.** For tasks involving large-scale datasets, Transformer-based models can better leverage the available data and provide stronger representation capability.
- **Recommendation 3: Evaluate task requirements.** For latency-sensitive tasks such as online anomaly detection or alert filtering, lightweight models are typically preferable. Conversely, tasks requiring stronger generalization or more complex reasoning may benefit from Transformers.
- **Recommendation 4: Assess task complexity.** In relatively simple scenarios, Transformers may be overkill; classical models, possibly with human assistance, can also achieve acceptable performance.

8.4 Data Fusion and Model Collaboration

Current Transformers in AIOps still operate with a marked degree of separation at both the data and model levels, lacking sufficient integration and collaboration, respectively.

At the data level, different types of telemetry data offer distinct and complementary views of system behavior: logs provide semi-structured textual records of runtime events [69], metrics deliver intuitive indicators of system performance and trends [70], and events summarize failure information in natural language [3]. Each data type plays an indispensable role in supporting AIOps tasks. Previous deep learning-based research has validated the feasibility and necessity of multimodal data fusion in AIOps, demonstrating the advantages of integrating heterogeneous data through various fusion techniques [10–12, 201, 209, 210]. Among them, ART [10] stands out as an early attempt to apply attention mechanisms to capture inter-modal dependencies, pioneering the use of Transformers for multimodal fusion in AIOps. However, most existing Transformers continue to focus on single-modality analysis, which may lead to a fragmented view when certain modalities are missing, thereby constraining performance gains.

Notably, distributed traces constitute one of the three pillars of observability, alongside logs and metrics. A trace consists of spans corresponding to service invocations, each recording execution metadata such as duration, status codes, and other annotations [211]. By linking spans across services, traces capture invocation paths and component dependencies, providing fine-grained visibility into request execution in distributed systems. However, Transformer-based trace analysis remains relatively underexplored. A key reason lies in the structural characteristics of trace data. While logs, metrics, and events naturally form sequential inputs suitable for Transformer-based sequence modeling, traces primarily encode invocation relationships that are inherently graph-structured (*i.e.*, directed acyclic graphs). Accordingly, existing trace analysis works mainly adopt graph-oriented techniques, including statistical methods (*e.g.*, spectrum-based fault localization [212–215]), graph-based ranking algorithms (*e.g.*, PageRank [216, 217]), and graph representation learning (*e.g.*, graph neural networks [218]). Due to the structural differences and the currently limited amount of relevant works, trace analysis is not the primary focus of our main analysis. Nevertheless, we include this discussion to provide a more complete view. Looking forward, integrating trace information into Transformer-based AIOps systems remains a promising direction, calling for structure-aware Transformers that can capture graph dependencies while jointly reasoning over trace data and other observability signals.

At the model level, the collaboration mechanisms among models of varying scales remain underexplored in a systematic manner. Large pre-trained models demonstrate robust semantic comprehension and generalization capabilities, whereas smaller models offer superior inference speed and resource efficiency. As discussed in Sec. 8.3, in certain scenarios, smaller models, when supported by human experts, can already deliver satisfactory results [208]. By establishing effective collaboration mechanisms, models of different scales or even with heterogeneous characteristics could be assigned to tackle specialized, fine-grained tasks, ultimately integrating their outputs to achieve a unified operational objective, much like assembling a puzzle [208, 219]. In such a framework, large pre-trained models may

serve as the central intelligence, orchestrating complex global reasoning and decision-making, while lightweight deep learning, machine learning, or data analysis models function as agile sensors and actuators, rapidly processing data, generating preliminary inferences, and delivering real-time responses. This task-oriented division and collaboration not only enhances efficiency but also enables comprehensive monitoring and precise control over complex system states. Thus, advancing research into model collaboration is pivotal to facilitating the transition of Transformers from experimental settings to production environments and accelerating the real-world deployment of AIOps systems.

8.5 Recommended Best Practices

The preceding discussion outlines open challenges and future directions. Building on these insights, we summarize several recommended best practices for applying Transformers in AIOps.

- **Best Practice 1: Mitigating hallucination through knowledge integration.** To mitigate hallucination issues, it is recommended to incorporate external knowledge bases or historical case repositories to strengthen domain expertise [5, 61, 63, 65, 111, 141, 143, 146, 150]. Another effective strategy is to embed expert knowledge and reasoning patterns into model design [58, 59, 62, 136, 146]. In addition, interpretability modules can be introduced to enhance the credibility and transparency of model outputs [145, 150].
- **Best Practice 2: Improving generalization through adaptive learning strategies.** To handle out-of-distribution data, pre-training and post-training alignment with diverse data patterns are widely adopted [32, 52, 55, 59, 100, 107, 122–126]. Transfer learning across systems [53] and few-shot incremental learning [104] further support rapid adaptation to dynamic environments and evolving data distributions.
- **Best Practice 3: Balancing efficiency and performance with lightweight architectures.** To balance computational cost, system efficiency, and model performance, lightweight architectures and modular designs are increasingly explored [29, 31, 57, 102, 103, 127, 128]. Parameter-efficient fine-tuning and rapid model transfer can further reduce dependence on large-scale training resources [53, 56, 110, 129]. Notably, streaming-oriented techniques [112] are particularly suitable for real-time operational scenarios.
- **Best Practice 4: Leveraging multimodal data fusion and collaborative model frameworks.** At the data level, integrating logs, metrics, events, and the relatively underexplored modality of traces enables more comprehensive observability and complementary insights [10, 59, 61, 142]. At the model level, task-oriented division and collaboration have emerged as a common paradigm. Several techniques [5, 54, 64, 66, 142] distribute subtasks across specialized models and coordinate interactions among small models, large models, and multiple agents to optimize overall performance.

These practices provide practical guidance for developing more intelligent and efficient AIOps systems.

8.6 Beyond Transformers and Emerging Neural Architectures

As discussed above, Transformers, with their powerful modeling capacity and general intelligence potential, have become the core architecture for many AIOps techniques. However, the self-attention mechanism still struggles with interpretability, generalization, and computational costs, *etc.* Meanwhile, a wave of emerging neural architectures is driving deep learning toward a "Beyond Transformers" stage, opening fresh possibilities for AIOps evolution. Among these, Mamba [220] introduces state-space modeling that enables linear-complexity handling of long-range dependencies. By replacing quadratic attention with structured state transitions, it improves efficiency and scalability in sequential data processing, particularly under high-throughput and long-horizon settings [221]. KAN [222], on the other

hand, is grounded in the Kolmogorov-Arnold representation theorem. It utilizes learnable univariate function families instead of conventional weights to explicitly capture functional relationships among variables, thereby producing more interpretable representations. This structured and symbolic formulation provides a pathway to uncover analytical expressions of system dynamics and supports knowledge extraction and transfer [223, 224].

From an AIOps task perspective, these architectures exhibit distinct advantages. **(1) Efficient Modeling for Streaming Tasks.** Mamba-style models enable linear-time sequence modeling, making them well-suited for continuous processing of large-scale data streams of logs, metrics, and events. Compared to the relatively heavy Transformer architectures, they are more efficient in latency-critical scenarios such as real-time monitoring, high-frequency metric forecasting, and online anomaly detection. **(2) Interpretable Modeling for Diagnostic Tasks.** KAN is particularly promising for interpretability-sensitive tasks such as root cause analysis and failure diagnosis. By modeling variable relationships through functional compositions, it facilitates explicit characterization of anomaly dynamics and potential causal patterns. In contrast, while Transformers can generate reasoning-like explanations, their internal mechanisms remain largely implicit, limiting their ability to provide directly interpretable insights.

Overall, emerging architectures offer more specialized improvements in efficiency and interpretability. Nevertheless, Transformers remain indispensable in AIOps, supported by a mature pretraining ecosystem, robust representation learning, and the emerging paradigm of natural language interaction. Yet this is far from the end. The rise of architectures beyond Transformers marks a shift from scale-driven to structure-driven intelligence, and from empirical stacking to principle-based modeling, signaling the dawn of a new era for AIOps.

9 Conclusion

This survey provides a comprehensive exploration of the question "Why Transformers", highlighting its value in AIOps through three key perspectives. First, in addressing "What Problem", we introduce a structured taxonomy from a multimodal perspective, reviewing the current applications of Transformers in analyzing log, metric, and event data, and subsequently arranging them within each modality according to task objectives, structural principles, and methodological approaches. The role changes of Transformers across phases also show an evolutionary trajectory. Second, in response to "Why Problem", we define a capability framework that outlines two fundamental and four advanced capabilities, revealing the unique advantages of Transformers in handling the complexity and diversity of AIOps tasks. Third, regarding "How Problem", we summarize common benchmarks, metrics, and evaluation strategies, offering comparative resources for further research and practice. While significant progress has been made in applying Transformers to AIOps, several challenges still remain in deployment. These include issues such as hallucination, limited explainability, generalization gaps, computational overhead, multimodal data fusion, and model collaboration. We further discuss the probable best practices for these challenges as actionable guidance. In summary, Transformers in AIOps present both promising opportunities and critical challenges. Its continuous advancement is essential for enhancing the intelligence and autonomy of large-scale software systems. We expect this survey to serve as a systematic reference for researchers and practitioners, fostering broader adoption of AIOps techniques in real-world scenarios and driving ongoing innovation in this evolving field.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62272249, 62302244), the Fundamental Research Funds for the Central Universities (XXX-63253249), and the Tianjin Key Research and Development Program (Grant No. 25YFYFFG00690).

References

[1] Paolo Notaro, Jorge Cardoso, and Michael Gerndt. 2021. A survey of aiops methods for failure management. *ACM Transactions on Intelligent Systems and Technology (TIST)* 12, 6 (2021), 1–45.

[2] Shenglin Zhang, Sibo Xia, Wenzhao Fan, Binpeng Shi, Xiao Xiong, Zhenyu Zhong, Minghua Ma, Yongqian Sun, and Dan Pei. 2024. Failure diagnosis in microservice systems: A comprehensive survey and analysis. *ACM Transactions on Software Engineering and Methodology* (2024).

[3] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Aiwei Liu, Yong Yang, Zhonghai Wu, Xuming Hu, Philip Yu, and Ying Li. 2025. A Survey of AIOps in the Era of Large Language Models. *Comput. Surveys* (2025).

[4] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenzhuo Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Saverese, and Steven CH Hoi. 2023. Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges. *arXiv preprint arXiv:2304.04661* (2023).

[5] Binpeng Shi, Yu Luo, Jingya Wang, Yongxin Zhao, Shenglin Zhang, Bowen Hao, Chenyu Zhao, Yongqian Sun, Zhi Zhang, Ronghua Sun, et al. 2025. FlowXpert: Expertizing Troubleshooting Workflow Orchestration with Knowledge Base and Multi-Agent Coevolution. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 4839–4850.

[6] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 674–688.

[7] Gerald Woo, Chenghao Liu, Akshat Kumar, and Doyen Sahoo. 2023. Pushing the limits of pre-training for time series forecasting in the cloudops domain. *arXiv preprint arXiv:2310.05063* (2023).

[8] Yinfang Chen, Manish Shetty, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Jonathan Mace, Chetan Bansal, Rujia Wang, and Saravan Rajmohan. 2025. AIOpsLab: A Holistic Framework for Evaluating AI Agents for Enabling Autonomous Cloud. In *MLSys '25*.

[9] Juan Pablo Ospina Herrera and Diego Botia. 2023. Cloud-Native Architecture for Distributed Systems that Facilitates Integration with AIOps Platforms. In *Colombian Conference on Computing*. Springer, 318–329.

[10] Yongqian Sun, Binpeng Shi, Mingyu Mao, Minghua Ma, Sibo Xia, Shenglin Zhang, and Dan Pei. 2024. ART: A Unified Unsupervised Framework for Incident Management in Microservice Systems. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 1183–1194.

[11] Shenglin Zhang, Pengxiang Jin, Zihan Lin, Yongqian Sun, Bicheng Zhang, Sibo Xia, Zhengdan Li, Zhenyu Zhong, Minghua Ma, Wa Jin, et al. 2023. Robust failure diagnosis of microservice system through multimodal data. *IEEE Transactions on Services Computing* 16, 6 (2023), 3851–3864.

[12] Guangba Yu, Pengfei Chen, Yufeng Li, Hongyang Chen, Xiaoyun Li, and Zibin Zheng. 2023. Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 553–565.

[13] Bowen Li, Xin Peng, Qilin Xiang, Hanzhang Wang, Tao Xie, Jun Sun, and Xuanzhe Liu. 2022. Enjoy your observability: an industrial survey of microservice tracing and analysis. *Empirical Software Engineering* 27 (2022), 1–28.

[14] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Wenhai Li, and Dan Ding. 2018. Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study. *IEEE Transactions on Software Engineering* 47, 2 (2018), 243–260.

[15] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)* 54, 2 (2021), 1–38.

[16] Zahra Zamanzadeh Darban, Geoffrey I Webb, Shirui Pan, Charu Aggarwal, and Mahsa Salehi. 2024. Deep learning for time series anomaly detection: A survey. *Comput. Surveys* 57, 1 (2024), 1–42.

[17] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of big Data* 8, 1 (2021), 53.

[18] Manjot Kaur and Aakash Mohta. 2019. A review of deep learning with recurrent neural network. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 460–465.

[19] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. 2016. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems* 29 (2016).

[20] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. 2022. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11963–11975.

[21] Ibomoiye Domor Mienye, Theo G Swart, and George Obaido. 2024. Recurrent neural networks: A comprehensive review of architectures, variants, and applications. *Information* 15, 9 (2024), 517.

[22] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. Pmlr, 1310–1318.

[23] Behnam Pourghassemi, Chenghao Zhang, Joo Hwan Lee, and Aparna Chandramowlishwaran. 2020. On the Limits of Parallelizing Convolutional Neural Networks on GPUs. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures* (Virtual Event, USA) *(SPAA '20)*. Association for Computing Machinery, New York, NY, USA, 567–569. doi:10.1145/3350755.3400266

[24] Xavier Gonzalez, Andrew Warrington, Jimmy T Smith, and Scott W Linderman. 2024. Towards scalable and stable parallelization of nonlinear rnns. *Advances in Neural Information Processing Systems* 37 (2024), 5817–5849.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[26] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)* 54, 10s (2022), 1–41.

[27] Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Nagat Drawel, Gaith Rjoub, and Witold Pedrycz. 2024. A comprehensive survey on applications of transformers for deep learning tasks. *Expert Systems with Applications* 241 (2024), 122666.

[28] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy. In *International Conference on Learning Representations*.

[29] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems* 34 (2021), 22419–22430.

[30] Tong Xiao, Zhe Quan, Zhi-Jie Wang, Yuquan Le, Yunfei Du, Xiangke Liao, Kenli Li, and Keqin Li. 2023. Loader: A log anomaly detector based on transformer. *IEEE Transactions on Services Computing* 16, 5 (2023), 3479–3492.

[31] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.

[32] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2024. Timer: generative pre-trained transformers are large time series models. In *Proceedings of the 41st International Conference on Machine Learning*. Article 1313, 31 pages.

[33] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. 2023. A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. *arXiv preprint arXiv:2305.17473* (2023).

[34] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 6778–6786.

[35] Toufique Ahmed, Supriyo Ghosh, Chetan Bansal, Thomas Zimmermann, Xuchao Zhang, and Saravan Rajmohan. 2023. Recommending root-cause and mitigation steps for cloud incidents using large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1737–1749.

[36] Van-Hoang Le and Hongyu Zhang. 2023. Log parsing: How far can chatgpt go?. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1699–1704.

[37] Youcef Remil, Anes Bendimerad, Romain Mathonat, and Mehdi Kaytoue. 2024. Aiops solutions for incident management: Technical guidelines and a comprehensive literature review. *arXiv preprint arXiv:2404.01363* (2024).

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

[39] Xin Zhou, Sicong Cao, Xiaobing Sun, and David Lo. 2024. Large language model for vulnerability detection and repair: Literature review and the road ahead. *ACM Transactions on Software Engineering and Methodology* (2024).

[40] Barbara Kitchenham and Pearl Brereton. 2013. A systematic review of systematic review process research in software engineering. *Information and software technology* 55, 12 (2013), 2049–2075.

[41] [n. d.]. IEEE Xplore Database. https://ieeexplore.ieee.org

[42] [n. d.]. ACM Digital Library. https://dl.acm.org

[43] [n. d.]. SpringerLink Database. https://link.springer.com

[44] [n. d.]. Web of Science Database. https://www.webofscience.com

[45] [n. d.]. ScienceDirect Database. https://www.sciencedirect.com

[46] [n. d.]. arXiv Database. https://arxiv.org

[47] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[48] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[49] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Miami, Florida, USA, 1107–1128. doi:10.18653/v1/2024.emnlp-main.64

[50] Junjielong Xu, Ziang Cui, Yuan Zhao, Xu Zhang, Shilin He, Pinjia He, Liqun Li, Yu Kang, Qingwei Lin, Yingnong Dang, et al. 2024. UniLog: Automatic Logging via LLM and In-Context Learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–12.

[51] Yilun Liu, Shimin Tao, Weibin Meng, Feiyu Yao, Xiaofeng Zhao, and Hao Yang. 2024. Logprompt: Prompt engineering towards zero-shot and interpretable log analysis. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*. 364–365.

[52] Shimin Tao, Yilun Liu, Weibin Meng, Zuomin Ren, Hao Yang, Xun Chen, Liang Zhang, Yuming Xie, Chang Su, Xiaosong Oiao, et al. 2023. Biglog: Unsupervised large-scale pre-training for a unified log representation. In *2023 IEEE/ACM 31st International Symposium on Quality of Service (IWQoS)*. IEEE, 1–11.

[53] Yicheng Sui, Xiaotian Wang, Tainyu Cui, Tong Xiao, Chenghao He, Shenglin Zhang, Yuzhi Zhang, Xiao Yang, Yongqian Sun, and Dan Pei. 2025. Bridging the Gap: LLM-Powered Transfer Learning for Log Anomaly Detection in New Software Systems. In *2025 IEEE 41st International Conference*

on Data Engineering (ICDE). IEEE Computer Society, 4414–4427.

[54] Jinyang Liu, Junjie Huang, Yintong Huo, Zhihan Jiang, Jiazhen Gu, Zhuangbin Chen, Cong Feng, Minzhi Yan, and Michael R Lyu. 2023. Log-based Anomaly Detection based on EVT Theory with feedback. *arXiv preprint arXiv:2306.05032* (2023).

[55] Yilun Liu, Yuhe Ji, Shimin Tao, Minggui He, Weibin Meng, Shenglin Zhang, Yongqian Sun, Yuming Xie, Boxing Chen, and Hao Yang. 2025. Loglm: From task-based to instruction-based automated log analysis. In *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 401–412.

[56] Defu Cao, Furong Jia, Sercan O Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. 2024. TEMPO: Prompt-based Generative Pre-trained Transformer for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[57] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. 2024. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[58] Minghao Liu, Shengqi Ren, Siyuan Ma, Jiahui Jiao, Yizhou Chen, Zhiguang Wang, and Wei Song. 2021. Gated transformer networks for multivariate time series classification. *arXiv preprint arXiv:2103.14438* (2021).

[59] Zhe Xie, Zeyan Li, Xiao He, Longlong Xu, Xidao Wen, Tieying Zhang, Jianjun Chen, Rui Shi, and Dan Pei. 2025. ChatTS: Aligning Time Series with LLMs via Synthetic Data for Enhanced Understanding and Reasoning. *Proceedings of the VLDB Endowment* 18, 8 (April 2025), 2385–2398. doi:10.14778/3742728.3742735

[60] Jinyang Liu, Shilin He, Zhuangbin Chen, Liqun Li, Yu Kang, Xu Zhang, Pinjia He, Hongyu Zhang, Qingwei Lin, Zhangwei Xu, et al. 2023. Incident-aware duplicate ticket aggregation for cloud systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2299–2311.

[61] Yongqian Sun, Bowen Hao, Xiaotian Wang, Chenyu Zhao, Yongxin Zhao, Binpeng Shi, Shenglin Zhang, Qiao Ge, Wenhu Li, Hua Wei, et al. 2025. LLM-Augmented Ticket Aggregation for Low-cost Mobile OS Defect Resolution. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 215–226.

[62] Pengxiang Jin, Shenglin Zhang, Minghua Ma, Haozhe Li, Yu Kang, Liqun Li, Yudong Liu, Bo Qiao, Chaoyun Zhang, Pu Zhao, et al. 2023. Assess and summarize: Improve outage understanding with large language models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1657–1668.

[63] Yongqian Sun, Tinghua Zheng, Xidao Wen, Weihua Kuang, Heng Liu, Shenglin Zhang, Chao Shen, Bo Wu, and Dan Pei. 2025. A Multimodal Intelligent Change Assessment Framework for Microservice Systems Based on Large Language Models. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 378–388.

[64] Pouya Hamadanian, Behnaz Arzani, Sadjad Fouladi, Siva Kesava Reddy Kakarla, Rodrigo Fonseca, Denizcan Billor, Ahmad Cheema, Edet Nkposong, and Ranveer Chandra. 2023. A Holistic View of AI-driven Network Incident Management. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks*. 180–188.

[65] Xuchao Zhang, Supriyo Ghosh, Chetan Bansal, Rujia Wang, Minghua Ma, Yu Kang, and Saravan Rajmohan. 2024. Automated root causing of cloud incidents using in-context learning with GPT-4. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 266–277.

[66] Zefan Wang, Zichuan Liu, Yingying Zhang, Aoxiao Zhong, Jihong Wang, Fengbin Yin, Lunting Fan, Lingfei Wu, and Qingsong Wen. 2024. Rcagent: Cloud root cause analysis by autonomous agents with tool-augmented large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4966–4974.

[67] Josu Diaz-De-Arcaya, Ana I Torre-Bastida, Gorka Zárate, Raúl Miñón, and Aitor Almeida. 2023. A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey. *Comput. Surveys* 56, 4 (2023), 1–30.

[68] Radoslav Gatev and Radoslav Gatev. 2021. Observability: Logs, metrics, and traces. *Introducing distributed application runtime (dapr) simplifying microservices applications development through proven and reusable patterns and practices* (2021), 233–252.

[69] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. 2021. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–37.

[70] Zhenyu Zhong, Qiliang Fan, Jiacheng Zhang, Minghua Ma, Shenglin Zhang, Yongqian Sun, Qingwei Lin, Yuzhi Zhang, and Dan Pei. 2023. A survey of time series anomaly detection methods in the aiops domain. *arXiv preprint arXiv:2308.00393* (2023).

[71] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

[72] Paul Youssef, Osman Koraş, Meijie Li, Jörg Schlötterer, and Christin Seifert. 2023. Give Me the Facts! A Survey on Factual Knowledge Probing in Pre-trained Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, Houda Bouamor, Juan Pino, and Kalika Bali (Eds.). Association for Computational Linguistics, 15588–15605. doi:10.18653/v1/2023.findings-emnlp.1043

[73] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. 2021. Pre-trained models: Past, present and future. *Ai Open* 2 (2021), 225–250.

[74] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.

[75] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2023. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence* 5, 3 (2023), 220–235.

[76] Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Xiang-Bo Mao, Sitaram Asur, et al. 2024. A comprehensive survey of LLM alignment techniques: RLHF, RLAIF, PPO, DPO and more. *arXiv preprint arXiv:2407.16216* (2024).

[77] Hanyu Lai, Xiao Liu, Junjie Gao, Jiale Cheng, Zehan Qi, Yifan Xu, Shuntian Yao, Dan Zhang, Jinhua Du, Zhenyu Hou, et al. 2025. A Survey of Post-Training Scaling in Large Language Models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2771–2791.

[78] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2790–2799.

[79] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 3045–3059. doi:10.18653/v1/2021.emnlp-main.243

[80] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 4582–4597. doi:10.18653/v1/2021.acl-long.353

[81] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[82] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems* 36 (2023), 10088–10115.

[83] [n. d.]. Hugging Face-PEFT. https://huggingface.co/docs/peft/index

[84] Yuji Cao, Huan Zhao, Yuheng Cheng, Ting Shu, Yue Chen, Guolong Liu, Gaoqi Liang, Junhua Zhao, Jinyue Yan, and Yun Li. 2024. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[85] Shuhe Wang, Shengyu Zhang, Jie Zhang, Runyi Hu, Xiaoya Li, Tianwei Zhang, Jiwei Li, Fei Wu, Guoyin Wang, and Eduard Hovy. 2024. Reinforcement learning enhanced llms: A survey. *arXiv preprint arXiv:2412.10400* (2024).

[86] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3, Feb (2003), 1137–1155.

[87] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *ACM computing surveys (CSUR)* 56, 9 (2024), 1–39.

[88] Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. 2025. A Survey on Large Language Models for Code Generation. *ACM Transactions on Software Engineering and Methodology* (July 2025). doi:10.1145/3747588 Just Accepted.

[89] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*. 149–158.

[90] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. 2016. Logmine: Fast pattern recognition for log analytics. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 1573–1582.

[91] Masayoshi Mizutani. 2013. Incremental mining of system log format. In *2013 IEEE International Conference on Services Computing*. IEEE, 595–602.

[92] Meiyappan Nagappan and Mladen A Vouk. 2010. Abstracting log lines to log event types for mining software system logs. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*. IEEE, 114–117.

[93] Hetong Dai, Heng Li, Che-Shao Chen, Weiyi Shang, and Tse-Hsun Chen. 2020. Logram: Efficient log parsing using n n-gram dictionaries. *IEEE Transactions on Software Engineering* 48, 3 (2020), 879–892.

[94] Zhen Ming Jiang, Ahmed E Hassan, Parminder Flora, and Gilbert Hamann. 2008. Abstracting execution logs to execution events for enterprise applications (short paper). In *2008 The Eighth International Conference on Quality Software*. IEEE, 181–186.

[95] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE international conference on web services (ICWS)*. IEEE, 33–40.

[96] Van-Hoang Le and Hongyu Zhang. 2021. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 492–504.

[97] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. Logbert: Log anomaly detection via bert. In *2021 international joint conference on neural networks (IJCNN)*. IEEE, 1–8.

[98] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. 2017. Generalization in deep learning. *arXiv preprint arXiv:1710.05468* 1, 8 (2017).

[99] Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. 2016. On the depth of deep neural networks: A theoretical view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[100] Antonio Mastropaolo, Luca Pascarella, and Gabriele Bavota. 2022. Using deep learning to generate complete log statements. In *Proceedings of the 44th International Conference on Software Engineering*. 2279–2290.

[101] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[102] Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, and Pinjia He. 2024. Divlog: Log parsing with prompt enhanced in-context learning. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–12.

[103] Zhihan Jiang, Jinyang Liu, Zhuangbin Chen, Yichen Li, Junjie Huang, Yintong Huo, Pinjia He, Jiazhen Gu, and Michael R Lyu. 2024. Lilac: Log parsing using llms with adaptive parsing cache. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 137–160.

[104] Zeyang Ma, An Ran Chen, Dong Jae Kim, Tse-Hsun Chen, and Shaowei Wang. 2024. Llmparser: An exploratory study on using large language models for log parsing. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[105] Junjie Huang, Zhihan Jiang, Zhuangbin Chen, and Michael Lyu. 2025. No More Labelled Examples? An Unsupervised Log Parser with LLMs. *Proceedings of the ACM on Software Engineering* 2, FSE (2025), 2406–2429.

[106] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 4171–4186.

[107] Shaohan Huang, Yi Liu, Carol Fung, He Wang, Hailong Yang, and Zhongzhi Luan. 2023. Improving Log-Based Anomaly Detection by Pre-Training Hierarchical Transformers. *IEEE Trans. Comput.* 72, 9 (Sept. 2023), 2656–2667. doi:10.1109/TC.2023.3257518

[108] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R Lyu. 2023. 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, 355–366.

[109] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. 2023. SwissLog: Robust Anomaly Detection and Localization for Interleaved Unstructured Logs. *IEEE Transactions on Dependable and Secure Computing* 20, 4 (2023), 2762–2780. doi:10.1109/TDSC.2022.3162857

[110] Ting Zhang, Xin Huang, Wen Zhao, Shaohuang Bian, and Peng Du. 2023. LogPrompt: A Log-based Anomaly Detection Framework Using Prompts. In *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[111] Jonathan Pan, Wong Swee Liang, and Yuan Yidi. 2024. RAGLog: Log Anomaly Detection using Retrieval Augmented Generation. In *2024 IEEE World Forum on Public Safety Technology (WFPST)*. IEEE, 169–174.

[112] Khalid Ayedh Alharthi, Arshad Jhumka, Sheng Di, and Franck Cappello. 2022. Clairvoyant: a log-based transformer-decoder for failure prediction in large-scale systems. In *Proceedings of the 36th ACM International Conference on Supercomputing*. 1–14.

[113] Canhua Wu, Nengwen Zhao, Lixin Wang, Xiaoqin Yang, Shining Li, Ming Zhang, Xing Jin, Xidao Wen, Xiaohui Nie, Wenchi Zhang, et al. 2021. Identifying root-cause metrics for incident diagnosis in online service systems. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 91–102.

[114] George EP Box, Gwilym M Jenkins, G Reinsel, et al. 1970. Forecasting and control. *Time Series Analysis* 3, 75 (1970), 1970.

[115] Jongseon Kim, Hyungjoon Kim, HyunGi Kim, Dongjun Lee, and Sungroh Yoon. 2025. A comprehensive survey of deep learning for time series forecasting: architectural diversity and open challenges. *Artificial Intelligence Review* 58, 7 (2025), 1–95.

[116] Ricardo P Masini, Marcelo C Medeiros, and Eduardo F Mendes. 2023. Machine learning advances for time series forecasting. *Journal of economic surveys* 37, 1 (2023), 76–111.

[117] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting?. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 37. 11121–11128.

[118] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems* 35 (2022), 9881–9893.

[119] Mohammad Amin Shabani, Amir H. Abdi, Lili Meng, and Tristan Sylvain. 2023. Scaleformer: Iterative Multi-scale Refining Transformers for Time Series Forecasting. In *The Eleventh International Conference on Learning Representations*.

[120] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

[121] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

[122] Azul Garza and Max Mergenthaler-Canseco. 2023. TimeGPT-1. *arXiv preprint arXiv:2310.03589* (2023).

[123] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Arian Khorasani, George Adamopoulos, Rishika Bhagwatkar, Marin Biloš, Hena Ghonia, Nadhir Hassen, Anderson Schneider, Sahil Garg, Alexandre Drouin, Nicolas Chapados, Yuriy Nevmyvaka, and Irina Rish. 2023. Lag-Llama: Towards Foundation Models for Time Series Forecasting. In *R0-FoMo:Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*.

[124] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified training of universal time series forecasting transformers. In *Forty-first International Conference on Machine Learning*.

[125] Ben Cohen, Emaad Khwaja, Kan Wang, Charles Masson, Elise Ramé, Youssef Doubli, and Othmane Abou-Amal. 2024. Toto: Time series optimized transformer for observability. *arXiv preprint arXiv:2407.07874* (2024).

[126] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. A decoder-only foundation model for time-series forecasting. In *Forty-first international conference on machine learning*.

[127] Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. 2024. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems* 36 (2024).

[128] Hao Xue and Flora D Salim. 2023. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2023).

[129] Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2025. LLM4TS: Aligning Pre-Trained LLMs as Data-Efficient Time-Series Forecasters. *ACM Transactions on Intelligent Systems and Technology* (2025). doi:10.1145/3719207

[130] Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. 2021. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International conference on learning representations*.

[131] Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. 2019. Deep adaptive input normalization for time series forecasting. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3760–3765.

[132] 2025. TimeGPT-1. https://docs.nixtla.io

[133] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. 2024. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*. 6555–6565.

[134] Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, et al. 2023. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196* (2023).

[135] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A transformer-based framework for multivariate time series representation learning. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2114–2124.

[136] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: deep transformer networks for anomaly detection in multivariate time series data. *Proceedings of the VLDB Endowment* 15, 6 (2022), 1201–1214. doi:10.14778/3514061.3514067

[137] Vaibhav Ganatra, Anjaly Parayil, Supriyo Ghosh, Yu Kang, Minghua Ma, Chetan Bansal, Suman Nath, and Jonathan Mace. 2023. Detection is better than cure: A cloud incidents perspective. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1891–1902.

[138] Shengming Zhang, Yanchi Liu, Xuchao Zhang, Wei Cheng, Haifeng Chen, and Hui Xiong. 2022. Cat: Beyond efficient transformer for content-aware anomaly detection in event sequences. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4541–4550.

[139] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Efficient customer incident triage via linking with system incidents. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1296–1307.

[140] Zhexiong Liu, Cris Benge, and Siduo Jiang. 2023. Ticket-bert: Labeling incident management tickets with language models. *arXiv preprint arXiv:2307.00108* (2023).

[141] Amrita Saha and Steven CH Hoi. 2022. Mining root cause knowledge from cloud service incident investigations for aiops. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*. 197–206.

[142] Junjielong Xu, Qinan Zhang, Zhiqing Zhong, Shilin He, Chaoyun Zhang, Qingwei Lin, Dan Pei, Pinjia He, Dongmei Zhang, and Qi Zhang. 2025. OpenRCA: Can Large Language Models Locate the Root Cause of Software Failures?. In *The Thirteenth International Conference on Learning Representations*.

[143] Changhua Pei, Zexin Wang, Fengrui Liu, Zeyan Li, Yang Liu, Xiao He, Rong Kang, Tieying Zhang, Jianjun Chen, Jianhui Li, et al. 2025. Flow-of-action: Sop enhanced llm-based multi-agent system for root cause analysis. In *Companion Proceedings of the ACM on Web Conference 2025*. 422–431.

[144] Saurabh Jha, Rohan R. Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, Noah Zheutlin, Saki Takano, Divya Pathak, Felix George, Xinbo Wu, Bekir O Turkkan, Gerard Vanloo, Michael Nidd, Ting Dai, Oishik Chatterjee, Pranjal Gupta, Suranjana Samanta, Pooja Aggarwal, Rong Lee, Jae wook Ahn, Debanjana Kar, Amit Paradkar, Yu Deng, Pratibha Moogi, Prateeti Mohapatra, Naoki Abe, Chandrasekhar Narayanaswami, Tianyin Xu, Lav R. Varshney, Ruchi Mahindru, Anca Sailer, Laura Shwartz, Daby Sow, Nicholas C. M. Fuller, and Ruchir Puri. 2025. ITBench: Evaluating AI Agents across Diverse Real-World IT Automation Tasks. In *Forty-second International Conference on Machine Learning*.

[145] Dylan Zhang, Xuchao Zhang, Chetan Bansal, Pedro Las-Casas, Rodrigo Fonseca, and Saravan Rajmohan. 2024. LM-PACE: Confidence estimation by large language models for effective root causing of cloud incidents. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 388–398.

[146] Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*. 674–688.

[147] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2383–2392. doi:10.18653/v1/D16-1264

[148] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems* 28 (2015).

[149] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*.

[150] Yuxuan Jiang, Chaoyun Zhang, Shilin He, Zhihao Yang, Minghua Ma, Si Qin, Yu Kang, Yingnong Dang, Saravan Rajmohan, Qingwei Lin, et al. 2024. Xpert: Empowering incident management with query recommendations via large language models. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[151] Zhihan Jiang, Jinyang Liu, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu, and Michael R Lyu. 2024. 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*. 223–234.

[152] Tianyu Cui, Shiyu Ma, Ziang Chen, Tong Xiao, Chenyu Zhao, Shimin Tao, Yilun Liu, Shenglin Zhang, Duoming Lin, Changchang Liu, Yuzhe Cai, Weibin Meng, Yongqian Sun, and Dan Pei. 2025. LogEval: A comprehensive benchmark suite for LLMs in log analysis. *Empirical Software Engineering* 30, 6 (Oct. 2025), 32 pages. doi:10.1007/s10664-025-10701-6

[153] Rob J Hyndman and Yeasmin Khandakar. 2008. Automatic time series forecasting: the forecast package for R. *Journal of statistical software* 27 (2008), 1–22.

[154] Cristian Challú Kin G. Olivares Azul Garza, Max Mergenthaler Canseco. 2022. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022. https://github.com/Nixtla/statsforecast

[155] Skipper Seabold and Josef Perktold. 2010. Statsmodels: econometric and statistical modeling with python. *SciPy* 7, 1 (2010), 92–96.

[156] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. 2022. Ts2vec: Towards universal representation of time series. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 8980–8987.

[157] Gerald Woo, Chenghao Liu, Doyen Sahoo, Akshat Kumar, and Steven Hoi. 2022. CoST: Contrastive Learning of Disentangled Seasonal-Trend Representations for Time Series Forecasting. In *International Conference on Learning Representations*.

[158] Zhihan Jiang, Jinyang Liu, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu, and Michael R Lyu. 2023. A large-scale benchmark for log parsing. *CoRR* (2023).

[159] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. 2019. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Ieee, 121–130.

[160] Weibin Meng, Federico Zaiter, Yuzhe Zhang, Ying Liu, Shenglin Zhang, Shimin Tao, Yichen Zhu, Tao Han, Yongpeng Zhao, En Wang, et al. 2023. Logsummary: Unstructured log summarization for software systems. *IEEE Transactions on Network and Service Management* 20, 3 (2023), 3803–3815.

[161] Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Bo Qiao, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2023. Empower Large Language Model to Perform Better on Industrial Domain-Specific Question Answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics, 294–312. doi:10.18653/v1/2023.emnlp-industry.29

[162] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[163] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering.. In *EMNLP (1)*. 6769–6781.

[164] Yukai Miao, Yu Bai, Li Chen, Dan Li, Haifeng Sun, Xizheng Wang, Ziqiu Luo, Yanyu Ren, Dapeng Sun, Xiuting Xu, et al. 2023. An empirical study of netops capability of pre-trained large language models. *arXiv preprint arXiv:2309.05557* (2023).

[165] 2023. DevOps-Eval. https://github.com/codefuse-ai/codefuse-devops-eval

[166] Hongcheng Guo, Jian Yang, Jiaheng Liu, Liqun Yang, Linzheng Chai, Jiaqi Bai, Junran Peng, Xiaorong Hu, Chao Chen, Dongfeng Zhang, xu Shi, Tieqiao Zheng, liangfan zheng, Bo Zhang, Ke Xu, and Zhoujun Li. 2024. OWL: A Large Language Model for IT Operations. In *The Twelfth International Conference on Learning Representations*.

[167] Yuhe Liu, Changhua Pei, Longlong Xu, Bohan Chen, Mingze Sun, Zhirui Zhang, Yongqian Sun, Shenglin Zhang, Kun Wang, Haiming Zhang, et al. 2025. OpsEval: A Comprehensive Benchmark Suite for Evaluating Large Language Models' Capability in IT Operations Domain. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 503–513.

[168] Xuchao Zhang, Tanish Mittal, Chetan Bansal, Rujia Wang, Minghua Ma, Zhixin Ren, Hao Huang, and Saravan Rajmohan. 2024. FLASH: A Workflow Automation Agent for Diagnosing Recurring Incidents. https://www.microsoft.com/en-us/research/wp-content/uploads/2024/10/FLASH_Paper.pdf Microsoft Research Technical Report.

[169] Uzay Çetin and Mursel Tasgin. 2020. Anomaly detection with multivariate k-sigma score using monte carlo. In *2020 5th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 94–98.

[170] Chuanjia Hou, Tong Jia, Yifan Wu, Ying Li, and Jing Han. 2021. Diagnosing performance issues in microservices with heterogeneous data source. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*. IEEE, 493–500.

[171] 2023. CloudOps. https://github.com/SalesforceAIResearch/pretrain-time-series-cloudops

[172] 2024. Loghub-2.0. https://github.com/logpai/Loghub-2.0

[173] 2025. LogEval. https://github.com/LinDuoming/LogEval

[174] 2025. OpenRCA. https://github.com/microsoft/OpenRCA

[175] 2023. MSQA. https://aka.ms/Microsoft_QA

[176] 2023. NetEval. https://huggingface.co/datasets/NASP/neteval-exam

[177] 2024. Owl-bench. https://github.com/HC-Guo/Owl

[178] 2025. OpsEval. https://opseval.cstcloud.cn

[179] 2025. AIOPSLAB. https://github.com/microsoft/AIOpsLab

[180] 2025. ITBench. https://github.com/itbench-hub/ITBench

[181] Zhiwei Gao, Carlo Cecati, and Steven X Ding. 2015. A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches. *IEEE transactions on industrial electronics* 62, 6 (2015), 3757–3767.

[182] Gao Zhiwei, Carlo Cecati, Steven X Ding, et al. 2015. A survey of fault diagnosis and fault-tolerant techniques—Part II: Fault diagnosis with knowledge-based and hybrid/active approaches. *IEEE Trans. Ind. Electron* 62, 6 (2015), 3768–3774.

[183] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A survey on software fault localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.

[184] Marc Solé, Victor Muntés-Mulero, Annie Ibrahim Rana, and Giovani Estrada. 2017. Survey on models and techniques for root-cause analysis. *arXiv preprint arXiv:1701.08546* (2017).

[185] Jacopo Soldani and Antonio Brogi. 2022. Anomaly detection and failure root cause analysis in (micro) service-based cloud applications: A survey. *ACM Computing Surveys (CSUR)* 55, 3 (2022), 1–39.

[186] Astha Garg, Wenyu Zhang, Jules Samaran, Ramasamy Savitha, and Chuan-Sheng Foo. 2021. An evaluation of anomaly detection and diagnosis in multivariate time series. *IEEE Transactions on Neural Networks and Learning Systems* 33, 6 (2021), 2508–2517.

[187] Adam Oliner, Archana Ganapathi, and Wei Xu. 2012. Advances and challenges in log analysis. *Commun. ACM* 55, 2 (2012), 55–61.

[188] Vijay Arya, Karthikeyan Shanmugam, Pooja Aggarwal, Qing Wang, Prateeti Mohapatra, and Seema Nagar. 2021. Evaluation of causal inference techniques for AIOps. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD)*. 188–192.

[189] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI open* 3 (2022), 111–132.

[190] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[191] Narendra Patwardhan, Stefano Marrone, and Carlo Sansone. 2023. Transformers in the real world: A survey on nlp applications. *Information* 14, 4 (2023), 242.

[192] Katikapalli Subramanyam Kalyan, Ajit Rajasekharan, and Sivanesan Sangeetha. 2021. Ammus: A survey of transformer-based pretrained models in natural language processing. *arXiv preprint arXiv:2108.05542* (2021).

[193] Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on computer science and information systems (FedCSIS)*. IEEE, 179–183.

[194] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.

[195] Fahad Shamshad, Salman Khan, Syed Waqas Zamir, Muhammad Haris Khan, Munawar Hayat, Fahad Shahbaz Khan, and Huazhu Fu. 2023. Transformers in medical imaging: A survey. *Medical Image Analysis* 88 (2023), 102802.

[196] Yingzhe Lyu, Heng Li, Mohammed Sayagh, Zhen Ming Jiang, and Ahmed E Hassan. 2021. An empirical study of the impact of data splitting decisions on the performance of AIOps solutions. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 4 (2021), 1–38.

[197] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2025. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *Computational Linguistics* (2025), 1–46.

[198] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=hSyW5go0v8

[199] Potsawee Manakul, Adian Liusie, and Mark Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 9004–9017.

[200] Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. 2024. Efficient adversarial training in llms with continuous attacks. *Advances in Neural Information Processing Systems* 37 (2024), 1502–1530.

[201] Yongqian Sun, Zihan Lin, Binpeng Shi, Shenglin Zhang, Shiyu Ma, Pengxiang Jin, Zhenyu Zhong, Lemeng Pan, Yicheng Guo, and Dan Pei. 2025. Interpretable failure localization for microservice systems based on graph autoencoder. *ACM Transactions on Software Engineering and Methodology* 34, 2 (2025), 1–28.

[202] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology* 15, 2 (2024), 1–38.

[203] Hila Chefer, Shir Gur, and Lior Wolf. 2021. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 782–791.

[204] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[205] Julia Herbinger, Susanne Dandl, Fiona K Ewald, Sofia Loibl, and Giuseppe Casalicchio. 2023. Leveraging model-based trees as interpretable surrogate models for model distillation. In *European Conference on Artificial Intelligence*. Springer, 232–249.

[206] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).

[207] Quentin Fournier, Gaétan Marceau Caron, and Daniel Aloise. 2023. A practical survey on faster and lighter transformers. *Comput. Surveys* 55, 14s (2023), 1–40.

[208] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small Language Models are the Future of Agentic AI. *arXiv preprint arXiv:2506.02153* (2025).

[209] Chenyu Zhao, Minghua Ma, Zhenyu Zhong, Shenglin Zhang, Zhiyuan Tan, Xiao Xiong, LuLu Yu, Jiayi Feng, Yongqian Sun, Yuzhi Zhang, et al. 2023. Robust multimodal failure detection for microservice systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5639–5649.

[210] Cheryl Lee, Tianyi Yang, Zhuangbin Chen, Yuxin Su, and Michael R Lyu. 2023. Eadro: An end-to-end troubleshooting framework for microservices on multi-source data. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1750–1762.

[211] Sasho Nedelkoski, Jorge Cardoso, and Odej Kao. 2019. Anomaly detection from system tracing data using multimodal deep learning. In *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE, 179–186.

[212] Zihao Ye, Pengfei Chen, and Guangba Yu. 2021. T-rank: A lightweight spectrum based fault localization approach for microservice systems. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, Los Alamitos,CA, 416–425.

[213] Zeyan Li, Junjie Chen, Rui Jiao, Nengwen Zhao, Zhijun Wang, Shuwei Zhang, Yanjun Wu, Long Jiang, Leiqin Yan, Zikai Wang, et al. 2021. Practical root cause localization for microservice systems via trace analysis. In *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. IEEE, Los Alamitos,CA, 1–10.

[214] Vijayaraghavan Murali, Edward Yao, Umang Mathur, and Satish Chandra. 2021. Scalable statistical root cause analysis on app telemetry. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, Los Alamitos,CA, 288–297.

[215] Chenxi Zhang, Zhen Dong, Xin Peng, Bicheng Zhang, and Miao Chen. 2024. Trace-based Multi-Dimensional Root Cause Localization of Performance Issues in Microservice Systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. IEEE, Los Alamitos,CA, 1–12.

[216] Guangba Yu, Pengfei Chen, Hongyang Chen, Zijie Guan, Zicheng Huang, Linxiao Jing, Tianjun Weng, Xinmeng Sun, and Xiaoyun Li. 2021. Microrank: End-to-end latency issue localization with extended spectrum analysis in microservice environments. In *Proceedings of the Web Conference 2021*. ACM, New York,NY, 3087–3098.

[217] Guangba Yu, Zicheng Huang, and Pengfei Chen. 2023. TraceRank: Abnormal service localization with dis-aggregated end-to-end tracing data in cloud native systems. *Journal of Software: Evolution and Process* 35, 10 (2023), e2413.

[218] Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. 2021. Sage: practical and scalable ML-driven performance debugging in microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, New York,NY, 135–151.

[219] Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. 2024. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089* (2024).

[220] Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*.

[221] Shriyank Somvanshi, Md Monzurul Islam, Mahmuda Sultana Mimi, Sazzad Bin Bashar Polock, Gaurab Chhetri, and Subasish Das. 2025. From S4 to Mamba: A Comprehensive Survey on Structured State Space Models. *arXiv preprint arXiv:2503.18970* (2025).

[222] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. 2025. KAN: Kolmogorov–Arnold Networks. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=Ozo7qJ5vZi

[223] Nataly R Panczyk, Omer F Erdem, and Majdi I Radaideh. 2025. Opening the AI black-box: Symbolic regression with Kolmogorov-Arnold Networks for advanced energy applications. *Energy and AI* (2025), 100595.

[224] Thomas R Harvey, Fabian Ruehle, Kit Fraser-Taliente, and James Halverson. 2025. Symbolic Regression with Multimodal Large Language Models and Kolmogorov Arnold Networks. *arXiv preprint arXiv:2505.07956* (2025).