Attribution-Aware Log Anomaly Detection via Index-Preserving Multi-View Fusion

1st Yuan Tian
School of Computer Science
Wuhan University
Wuhan, China
tytian@whu.edu.cn

2nd Shi Ying
School of Computer Science
Wuhan University
Wuhan, China
yingshi@whu.edu.cn

***Abstract*—Log anomaly detection is essential to the reliability of large-scale software systems. In real-world deployment, anomaly alarms should not only be accurate but also reveal when anomalies occur, where they arise, and which components and events are most responsible. However, most existing methods model logs from a single view and lose fine-grained traceability during representation learning, which limits their value for actionable diagnosis. To address this issue, we propose IP-Log, an attribution-aware framework for log anomaly detection based on index-preserving multi-view fusion. IP-Log models system behavior from three complementary views, namely component, event, and time, and structures their interactions according to two principles. One is level-confined information flow, which prevents premature entanglement across views. The other is index-preserving computation, which maintains the correspondence between latent representations and original log entities throughout the model. As a result, anomaly evidence can be decomposed into fine-grained contributions over components, events, and time segments, and the model can further identify top-ranked evidence triplets for diagnosis and troubleshooting. Experiments on public log benchmarks show that IP-Log achieves strong anomaly detection performance while also providing interpretable and index-aligned evidence attribution.**

***Index Terms*—System logs, Log anomaly detection, Multi-view representation, Anomaly attribution**

I. Introduction

Log anomaly detection plays a critical role in ensuring the reliability and availability of large-scale software systems. Modern infrastructures continuously generate massive volumes of system logs that record runtime behaviors such as component activities, execution events, and temporal evolution. Deviations from normal log patterns often indicate latent failures, performance degradation, or security threats, making timely and accurate anomaly detection essential for system diagnosis and maintenance [1].

Beyond detection accuracy, practical deployment increasingly demands attribution-aware anomaly detection. System operators must understand when an anomaly occurs, which components or events contribute most to the alert, and which time segments are most suspicious, in order to support efficient triage, diagnosis, and remediation. However, most existing log anomaly detection methods fall short of this requirement. Many approaches model logs from a single perspective, such as temporal sequences [2], [3], window-based statistics [4], or event graphs [5], [6]. During representation learning, these methods often aggregate or transform log data in ways that break the correspondence between learned representations and original log entities, making fine-grained, evidence-based attribution difficult or unreliable.

A key observation is that system logs are inherently multi-view. Each log message is naturally associated with a component that emits it, an event describing the system action, and a timestamp specifying when it occurs. These three dimensions—component (C), event (E), and temporal (T)—capture complementary aspects of system behavior. While prior work has explored individual views in isolation, jointly modeling their interactions remains underexplored, especially under the requirement that representation units remain traceable to original log entities. As a result, existing methods may suffer from blind spots when anomalies arise from cross-view interactions, such as specific components repeatedly emitting certain events within a short time segment.

To address these limitations, we propose IP-Log, an attribution-aware log anomaly detection framework based on index-preserving multi-view interaction modeling. IP-Log represents system behavior from the component, event, and temporal views, and organizes their dependencies through a hierarchical interaction structure. A central design principle of IP-Log is index-preserving computation, which maintains the correspondence between learned representations and original log entities throughout the modeling process. This design enables anomaly evidence to be traced back to concrete components, events, and time segments, thereby supporting fine-grained and actionable attribution in addition to accurate detection.

Our main contributions are summarized as follows:

- **Attribution-aware multi-view log modeling.** We propose a unified multi-view framework that jointly models the component, event, and temporal perspectives of system logs, enabling structured and attribution-aware log anomaly detection.
- **Hierarchical interaction decomposition.** We introduce a hierarchical modeling paradigm that decomposes cross-

This work was supported in part by the National Natural Science Foundation of China under Grants 62472329 and 62072342, and in part by the National Key Research and Development Program of China under Grant 2022YFB3304300.

view dependencies into (i) intra-view representations, (ii) pairwise cross-view interactions, and (iii) tri-view synergy, yielding expressive modular representations.

- **Index-preserving aggregation for interpretability.** We design an index-preserving aggregation mechanism that enables fine-grained, node-level attribution across the component, event, and temporal dimensions, providing actionable evidence attribution for detected anomalies.

II. Related Work

System log anomaly detection has been extensively studied as a fundamental technique for diagnosing failures in large-scale distributed systems. A common pipeline first parses raw logs into structured event types and then models system behavior based on these structured representations, either through temporal or window-based statistics [7], or through explicit modeling of discrete event dependencies [8]. Early approaches predominantly rely on clustering or invariant mining over parsed logs, such as LogCluster, which offers a certain degree of interpretability but suffers from limited robustness under evolving workloads and heterogeneous components [9]. More recent neural detectors adopt sequence-centric modeling, learning normal patterns of event sequences using recurrent or attention-based architectures. Representative methods include DeepLog [2] and LogBERT [3]. Despite their strong detection performance, these methods may lose fine-grained traceability when logs from concurrent services interleave, or when single-view sequential signals are insufficient to characterize complex system behavior. To better capture contextual relations, graph-based methods such as Log2Graphs [6] encode event dependencies with graph structures for unsupervised anomaly detection. However, the evidence provided by these approaches is often indirect, since latent graph nodes do not necessarily preserve a stable one-to-one correspondence with the original entities, which limits evidence-level attribution. In parallel, studies on multi-view learning and interaction factorization have shown that explicitly organizing cross-view interactions can improve both performance and interpretability, as exemplified by factorization machines [10] and tensor decomposition frameworks [11]. Building on these insights, our work fuses component, event, and temporal views under index-preserving constraints, enabling evidence-attributable anomaly detection that pinpoints where, what, and when each anomaly occurs.

III. Problem Setup

As illustrated in Fig. 1, we first transform raw logs into a structured tensor representation and then derive index-preserved multi-view entities for subsequent interaction modeling. System logs record runtime behaviors as discrete messages, each associated with a timestamp, a component identifier, and an event type obtained from log parsing. Given a chronologically ordered log stream, we segment the logs into consecutive windows using a fixed-span sliding window. Within each window, log messages are grouped by component, and the occurrences of each event type are counted, yielding a component–event count matrix. By stacking these matrices

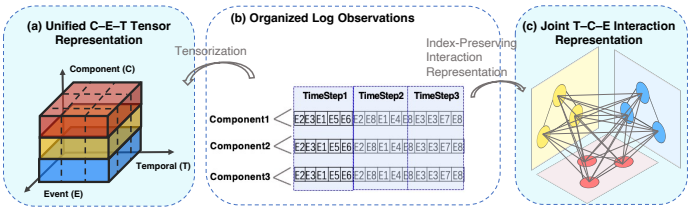

Fig. 1. From Raw Logs to Index-Preserving Multi-View Representation. (a) Unified component–event–temporal organization of raw logs. (b) Tensorization into a structured C–E–T representation. (c) Index-preserved multi-view entities for subsequent interaction modeling.

along the temporal dimension, we obtain a three-dimensional tensor $\mathbf{X} \in \mathbb{R}^{N_C \times N_E \times N_T}$, where N_C, N_E, and N_T denote the numbers of components, event types, and time segments, respectively. This process, illustrated in Fig. 1(a) and Fig. 1(b), converts raw log streams into a unified representation over components, events, and time.

The tensor $\mathbf{X}$ naturally supports three index-orthogonal views, as shown in Fig. 1(c), by fixing one dimension and varying the other two. Specifically, each component slice $\mathbf{X}_{c,:,:}$ represents the event distribution of component c over time, each event slice $\mathbf{X}_{:,e,:}$ captures how event e evolves across components and time, and each temporal slice $\mathbf{X}_{:,:,t}$ describes the system state at time segment t. These three types of slices encode the *who*, *what*, and *when* aspects of system behavior, respectively. Importantly, each slice is anchored to a specific component, event, or time index, thereby preserving the correspondence between representation units and original log entities. These index-preserved view-specific slices serve as the fundamental entities for subsequent multi-view interaction modeling, providing a traceable representation basis for analyzing cross-view dependencies.

IV. Method

Given the preprocessed component–event–temporal tensor $\mathbf{X} \in \mathbb{R}^{N_C \times N_E \times N_T}$, we construct three complementary views: component (C), event (E), and temporal (T). Built on the index-preserved view-specific entities defined in Section III, IP-Log learns their joint representation through a unified modeling framework that combines *Hierarchical Interaction Decomposition* with *Index-Preserving Aggregation*. This design enables structured cross-view interaction modeling while preserving traceability to the original log entities.

As illustrated in Fig. 2, IP-Log focuses on a core hierarchical interaction process consisting of three stages: *Intra-view Enhancement*, *Pairwise Cross-view Interaction*, and *Tri-view Synergy*. These stages progressively capture view-specific patterns, pairwise dependencies across views, and higher-order tri-view correlations under the index-preserving constraint. The resulting view representations are further aligned and fused into a joint representation, which is then used for anomaly detection and evidence attribution.

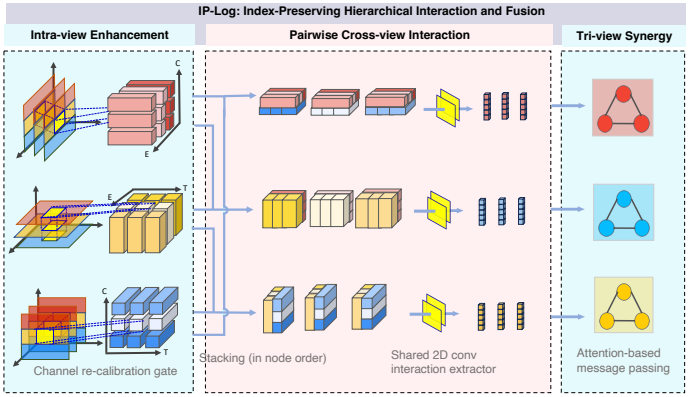

Fig. 2. Hierarchical Multi-View Interaction Modeling in IP-Log. The framework focuses on three core stages: intra-view enhancement, pairwise cross-view interaction, and tri-view synergy.

A. Hierarchical Interaction Decomposition

To capture the cross-view nature of log anomalies, IP-Log employs a hierarchical,

$$\begin{aligned}\Phi(C,E,T) = \varphi(C) \oplus \varphi(E) \oplus \varphi(T) \oplus \psi(C,E) \\ \oplus \psi(C,T) \oplus \psi(E,T) \oplus \rho(C,E,T)\end{aligned} \tag{1}$$

where $\varphi(\cdot)$ denotes intra-view representations, $\psi(\cdot,\cdot)$ models pairwise cross-view interactions, and $\rho(\cdot,\cdot,\cdot)$ captures tri-view synergy. The operator $\oplus$ denotes a learnable composition.

Based on this decomposition, we define a contextualized representation for each target view $p \in \{C,E,T\}$ with complementary views (q,r) as:

$$\Phi(p;q,r) \triangleq \varphi(p) \oplus \psi(p,q) \oplus \psi(p,r) \oplus \rho_p \tag{2}$$

where ρ_p is the view-specific instantiation of the tri-view term, aggregating higher-order dependencies while remaining aligned with the indices of view p. After constructing contextualized representations for all three views, the final joint representation is obtained by:

$$\Phi(C,E,T) \triangleq \Phi(C;E,T) \oplus \Phi(E;C,T) \oplus \Phi(T;C,E) \tag{3}$$

Eqs. (1)–(3) provide a unified blueprint for the entire pipeline. All subsequent modules can be viewed as concrete, learnable instantiations of φ, ψ, and ρ. We implement this blueprint progressively by interaction order, from intra-view to pairwise and then tri-view.

B. Index-Preserving Aggregation

Eq. (2) defines a p-anchored representation $\Phi(p;q,r)$, aggregating information from complementary views (q,r) into the target view $p \in \{C,E,T\}$. To keep this anchoring both operational and traceable, we enforce two constraints: (i) *level-confined information flow*, which introduces interactions progressively by order to prevent unintended cross-view semantic entanglement; and (ii) *index-preserving per-node computation*, which ensures that the i_p-th latent node always corresponds to the same underlying entity without mixing different p-indices. The resulting view representations are then projected into a shared semantic space and fused to form the final joint representation.

C. Hierarchical Multi-View Interaction Modeling

This module instantiates the intra-view and pairwise terms in Eq. (1) under the index-preserving constraints introduced above. Let $\varphi(\cdot)$ denote an intra-view encoder, $\psi(\cdot,\cdot)$ a pairwise interaction module, and $\rho(\cdot)$ a tri-view synergy operator. All three operators are index-preserving: output representations retain the same node indices as their inputs, enabling traceable, progressive construction of contextualized view representations.

1) Intra-view Enhancement: We begin with the intra-view encoder $\varphi(\cdot)$. Given the log tensor $\mathbf{X} \in \mathbb{R}^{N_C \times N_E \times N_T}$, it is first reshaped so that the target view p serves as the channel dimension, while the complementary views (q,r) define the spatial axes, yielding a view-specific tensor $\mathbf{X}^{(p)} \in \mathbb{R}^{N_q \times (N_r \times N_T)}$. To characterize channel-wise importance, a channel descriptor $\mathbf{z}^{(p)}$ is computed by applying global average pooling (GAP) over the spatial dimensions (q,r):

$$\mathbf{z}^{(p)} = \mathrm{GAP}\left(\mathbf{X}^{(p)}\right) \in \mathbb{R}^{N_p} \tag{4}$$

The resulting descriptor summarizes the global response of each channel and provides the basis for subsequent attention estimation. It is then passed through a one-dimensional convolutional layer $\mathrm{Conv1D}^{(p)}_{\mathrm{attn}}$ with an adaptive kernel size $k^{(p)} = \lceil (\log_2 N_p + 1)/2 \rceil$, following [12], and a sigmoid activation is applied to obtain the channel attention weights:

$$\mathbf{a}^{(p)} = \sigma\left(\mathrm{Conv1D}^{(p)}_{\mathrm{attn}}\left(\mathbf{z}^{(p)}\right)\right) \in \mathbb{R}^{N_p} \tag{5}$$

This operation assigns an importance weight to each channel according to its aggregated contribution. Then, the attention weights $\mathbf{a}^{(p)}$ are used to recalibrate $\mathbf{X}^{(p)}$ via channel-wise multiplication with broadcasting along the spatial dimensions:

$$\tilde{\mathbf{X}}^{(p)} = \mathbf{a}^{(p)} \odot \mathbf{X}^{(p)} \in \mathbb{R}^{N_p \times N_q \times N_r} \tag{6}$$

where $\odot$ denotes element-wise multiplication with broadcasting. In this way, informative channels are emphasized while less relevant ones are suppressed.

Subsequently, we employ a one-dimensional convolutional block $f^{(p)}_{\mathrm{Conv1D}}$ along the channel dimension to extract local patterns. This block preserves the channel count and consists of a padded Conv1D layer followed by Batch Normalization, ReLU activation, and Dropout:

$$f^{(p)}_{\mathrm{Conv1D}} = \mathrm{Dropout} \circ \mathrm{ReLU} \circ \mathrm{BN} \circ \mathrm{Conv1D}^{(p)} \tag{7}$$

This design enables local dependency modeling within the target view without altering the underlying index structure. The output of this block forms the intra-view representation:

$$\varphi(p) \triangleq f^{(p)}_{\mathrm{Conv1D}}\left(\tilde{\mathbf{X}}^{(p)}\right) \in \mathbb{R}^{N_p \times N_q \times N_r} \tag{8}$$

The resulting representation captures enhanced intra-view patterns while remaining aligned with the original view indices.

In summary, this stage produces $\varphi(p)$, which enhances the target view p while keeping the complementary views (q,r) aligned with their native indices.

2) Inter-view Interaction: Pairwise interactions between the target view p and its complementary views (q, r) are derived from the intra-view representations $\varphi(q) \in \mathbb{R}^{N_q \times N_p \times N_r}$ and $\varphi(r) \in \mathbb{R}^{N_r \times N_p \times N_q}$. We first reshape them into the p-reference frame to obtain $\tilde{\varphi}(q) \in \mathbb{R}^{N_p \times (N_q \times N_r)}$ and $\tilde{\varphi}(r) \in \mathbb{R}^{N_p \times (N_r \times N_q)}$. The reshaped representations are then concatenated along the channel dimension:

$$\mathbf{\Psi}^{(p)} = \left[\tilde{\varphi}(q) \,\|\, \tilde{\varphi}(r)^\top\right] \in \mathbb{R}^{N_p \times (2 \times N_q \times N_r)} \tag{9}$$

where $\|$ denotes concatenation. In this representation, $\mathbf{\Psi}^{(p)}$ contains N_p slices along the p-axis, each denoted as $\mathbf{\Xi}_i^{(p)} \in \mathbb{R}^{2 \times N_q \times N_r}$.

To extract local q–r co-occurrence patterns, a two-dimensional convolutional block is defined, followed by global average pooling and a fully connected layer:

$$f_{\text{Conv2D}}^{(p)} = \text{FC} \circ \text{GAP} \circ \text{ReLU} \circ \text{Conv2D}^{(p)} \tag{10}$$

For each node i in view p, the block is applied to the corresponding slice $\mathbf{\Xi}_i^{(p)}$ to obtain a d_f-dimensional interaction embedding:

$$\mathbf{m}_i^{(p)} = f_{\text{Conv2D}}^{(p)}\left(\mathbf{\Xi}_i^{(p)}\right) \in \mathbb{R}^{d_f} \tag{11}$$

Finally, stacking $\{\mathbf{m}_i^{(p)}\}_{i=1}^{N_p}$ in index order yields the pairwise interaction representation:

$$\psi(p, q) \oplus \psi(p, r) \triangleq \{\mathbf{m}_i^{(p)}\}_{i=1}^{N_p} \in \mathbb{R}^{N_p \times d_f} \tag{12}$$

By executing all operations slice-wise along the p-axis, the process preserves the native indices and per-node independence in the target view p.

3) Tri-view Synergy: In the target view p, the pairwise interaction features $\{\mathbf{m}_i^{(p)}\}_{i=1}^{N_p}$ have already aggregated information from the complementary views. Building on this, propagating these features across nodes in view p further integrates dependencies within p and yields the tri-view synergy specific to p.

To this end, we construct a view-p graph with N_p nodes, where node features are given by $\mathbf{m}i^{(p)} i = 1^{N_p} \in \mathbb{R}^{N_p \times d_f}$. A fully connected topology with a static edge set is used. For the temporal view, temporal order is preserved by augmenting node features with fixed sinusoidal positional encodings, as in [13]. On the view-p graph, a Graph Attention Network (GAT) propagates information across nodes to produce context-aware embeddings:

$$\mathbf{g}_i^{(p)} = \text{GAT}\left(\mathbf{m}_i^{(p)}, \{\mathbf{m}_j^{(p)}\}_{j=1}^{N_p}\right) \in \mathbb{R}^{d_g} \tag{13}$$

Collecting these node embeddings yields the tri-view synergy representation for view p:

$$\boldsymbol{\rho}_p \triangleq \{\mathbf{g}_i^{(p)}\}_{i=1}^{N_p} \in \mathbb{R}^{N_p \times d_g} \tag{14}$$

Finally, the contextualized representation is formed by concatenating, for each node, its pairwise interaction feature with the corresponding graph embedding:

$$\Phi(p; q, r) = \{[\mathbf{m}_i^{(p)} \| \mathbf{g}_i^{(p)}]\}_{i=1}^{N_p} \in \mathbb{R}^{N_p \times (d_f + d_g)} \tag{15}$$

In this way, the resulting representation jointly captures intra-view enhancement, pairwise interaction, and tri-view synergy, while remaining strictly index-preserving along the component, event, and temporal axes.

D. Cross-View Consistency and Fusion

The tri-view enhanced representations are further aligned into a shared latent space to reduce view-specific bias and improve fusion consistency. A pairwise alignment loss is adopted to encourage consistency across views. The aligned representations are then fused into a unified joint representation $\Phi(C, E, T)$, which is used for subsequent anomaly detection and attribution.

E. Optimization and Anomaly Detection

IP-Log is trained on normal data with a joint objective over the unified representation, combining reconstruction, temporal prediction on the temporal-view part, and cross-view consistency among the contextualized representations of the three views:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \lambda \mathcal{L}_{\text{pred}} + \beta \mathcal{L}_{\text{align}} \tag{16}$$

where $\mathcal{L}_{\text{rec}}$ reconstructs the joint representation, $\mathcal{L}_{\text{pred}}$ predicts subsequent temporal states from the temporal-view representation, and $\mathcal{L}_{\text{align}}$ encourages consistency among the contextualized representations of the component, event, and temporal views.

At inference time, IP-Log computes anomaly scores from both reconstruction error and temporal prediction error, and flags an input instance as anomalous when the combined score exceeds a threshold γ. Owing to the index-preserving design, the anomaly score can be decomposed into node-level contributions:

$$s = \sum_{i \in \mathcal{N}} r_i \tag{17}$$

where $\mathcal{N}$ denotes the set of latent nodes aligned with component, event, and temporal entities. We further decompose the anomaly score into view-specific node scores, rank the nodes within each view, and select the highest-scoring components, events, and time slices as attribution evidence. The returned triplets (c, e, t) indicate where, what, and when the anomaly occurs.

V. Experiments

A. Experiments Setup

Datasets and preprocessing. We evaluate IP-Log on four widely used public log anomaly detection benchmarks: HDFS, BGL, Thunderbird (Tbird), and Spirit. All datasets provide parsed event types and anomaly labels for individual log messages. Following common practice [7], we filter out extremely frequent and extremely rare event types, and retain the top 10 most frequent components to construct compact yet representative views. For HDFS and BGL, logs are grouped into sessions using `block_id` and `node_id`, respectively. For the large-scale datasets (Tbird and Spirit), we uniformly subsample millions of log messages and apply a fixed-span

TABLE I
DATASET STATISTICS

Dataset	#Messages	#Components	#Event Types	Anomaly Rate (%)
Tbird	8,000,000	10	382	18.02
Spirit	8,000,000	10	850	37.37
HDFS	4,999,999	10	44	38.20
BGL	4,747,963	10	192	7.31

TABLE II
ANOMALY DETECTION PERFORMANCE

Method	Metric	Tbird	Spirit	HDFS	BGL	Avg.
LogCluster	F1	69.68	72.08	32.48	31.34	51.39
	Precision	74.61	78.21	20.72	97.37	67.73
	Recall	65.29	66.85	75.28	18.70	56.53
DeepLog	F1	88.32	84.83	80.34	87.45	85.24
	Precision	84.20	92.45	**89.16**	82.99	87.20
	Recall	92.86	78.37	73.01	92.41	84.16
Log2Graphs	F1	95.36	98.86	**88.28**	94.30	94.20
	Precision	93.30	98.73	86.52	95.58	93.53
	Recall	97.59	99.01	**90.08**	93.01	94.92
IP-Log	F1	**98.26**	**99.49**	87.26	**97.30**	**95.58**
	Precision	**97.32**	**98.98**	84.96	95.35	**94.15**
	Recall	**99.25**	**99.95**	89.64	**99.34**	**97.04**

sliding window to construct component–event count matrices; a window is labeled anomalous if it contains at least one abnormal log message. Based on the per-log anomaly labels, we further mark the associated components, events, and time slices of abnormal logs as anomalous, which serve as the attribution benchmark. Table I summarizes the resulting dataset statistics.

Baselines. IP-Log is compared with representative methods from three major paradigms: (i) statistical/traditional detectors, including LogCluster [14]; (ii) sequence-based neural detectors, including DeepLog [2]; and (iii) graph-based detectors, including Log2Graphs [6].

Evaluation Metrics. Detection performance is measured by Precision, Recall, and F1-score [2], [5]. Precision quantifies the proportion of predicted anomalies that are correct, Recall quantifies the proportion of true anomalies that are detected, and F1 is their harmonic mean.

Evidence attribution is assessed by interpreting the output as a ranked list of candidate entities (components/events/time segments) and reporting top-k ranking metrics: Precision@k, Recall@k, and NDCG@k. Precision@k is the proportion of ground-truth anomalous entities in the top-k list; Recall@k is the proportion of ground-truth entities recovered by the top-k list; and NDCG@k evaluates ranking quality by giving higher credit to correct entities placed earlier. Results are reported with $k \in \{3, 5\}$ to reflect small, actionable evidence sets.

B. Overall Detection Performance

We compare IP-Log against three representative baselines on four public datasets. Table II summarizes the results. LogCluster, a statistical method based on clustering over parsed logs, achieves an average F1 of only 51.39%, indicating that pattern-matching approaches without learned representations struggle to capture complex temporal and structural dependencies in system logs. DeepLog, a sequence-based neural detector, improves average F1 to 85.24% but still falls 10.34% below IP-Log. Under high concurrency, interleaved normal sequences produce unseen patterns that a single-view sequential model may misclassify as anomalies. IP-Log instead models system behavior with a component–event–temporal tensor and explicitly captures cross-view dependencies, reducing reliance on strict event ordering. Log2Graphs encodes event dependencies with graph structures and achieves an average F1 of 94.20%, yet its average recall is 2.12% lower than IP-Log. Because Log2Graphs relies solely on the event view, it introduces view-specific blind spots that increase missed anomalies. In contrast, IP-Log jointly models component, event, and temporal views, enabling it to detect cross-view anomalies (e.g., a specific component continuously emitting certain events within a time segment) that single-view methods overlook, thereby achieving the highest average F1 of 95.58% and recall of 97.04%.

TABLE III
ABLATION RESULTS OF AGGREGATION BLOCKS

Variant	F1	Precision	Recall
Spirit			
No-ϕ	95.37 ($\downarrow$4.12)	97.12 ($\downarrow$1.86)	94.47 ($\downarrow$5.48)
No-ρ	97.28 ($\downarrow$2.21)	97.76 ($\downarrow$1.22)	96.83 ($\downarrow$3.12)
Full	99.49	98.98	99.95
BGL			
No-ϕ	93.52 ($\downarrow$3.78)	93.61 ($\downarrow$1.74)	93.44 ($\downarrow$5.90)
No-ρ	95.25 ($\downarrow$2.05)	94.27 ($\downarrow$1.08)	96.60 ($\downarrow$2.74)
Full	97.30	95.35	99.34

C. Aggregation Block Ablation

Table III reports the ablation results of the two aggregation blocks on Spirit and BGL. Removing the intra-view enhancement block (No-ϕ) leads to the largest performance degradation. Specifically, the F1 score drops by 4.12% on Spirit and 3.78% on BGL, while recall decreases by 5.48% and 5.90%, respectively. These results indicate that intra-view enhancement plays a critical role in capturing view-specific semantics and improving the sensitivity to true anomalies. Removing the tri-view synergy block (No-ρ) also yields consistent performance declines, although to a smaller extent. The F1 score decreases by 2.21% on Spirit and 2.05% on BGL, with precision dropping by 1.22% and 1.08%, respectively. This suggests that the tri-view synergy block provides complementary gains by further integrating cross-node dependencies across views. Overall, the full model achieves the best performance on both datasets, confirming that both aggregation blocks contribute to the effectiveness of IP-Log.

D. Anomaly Attribution Performance

Beyond detection, IP-Log decomposes the anomaly score into component-, event-, and time-specific contributions to support evidence-based attribution. As shown in Fig. 3,

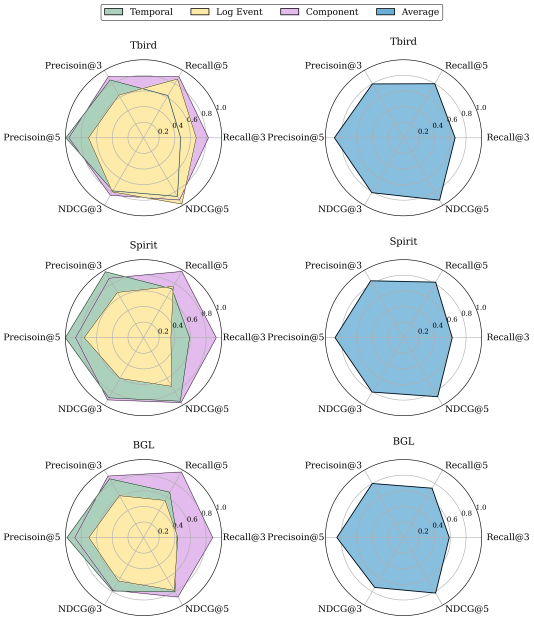

Fig. 3. Anomaly attribution performance of IP-Log

IP-Log achieves strong evidence retrieval performance across datasets under top-k attribution metrics (e.g., Precision@k/Recall@k/NDCG@k). Increasing the evidence budget from Top-3 to Top-5 consistently improves the retrieval quality, suggesting that the ranked evidence remains stable when operators request a slightly larger actionable set. Across views, the component view is typically the most informative, highlighting the importance of component-level context for tracing abnormal behavior in distributed systems. The temporal view is more challenging under bursty workloads (e.g., Spirit), where anomalous patterns may span adjacent windows; nevertheless, its performance improves with a larger k once key time segments are retrieved. Overall, these results indicate that index-preserving modeling enables fine-grained, index-aligned evidence attribution in addition to accurate detection.

VI. Conclusion

IP-Log is an attribution-aware log anomaly detection framework based on index-preserving multi-view fusion. Starting from a unified component–event–temporal tensor representation, it jointly models system behavior from three complementary views and captures cross-view dependencies through hierarchical interaction modeling over intra-view patterns, pairwise interactions, and tri-view synergy. By enforcing index-preserving computation with level-confined information flow, the proposed framework maintains the correspondence between latent representations and original entities throughout the modeling process, thereby supporting fine-grained evidence attribution over components, events, and time segments. Experimental results on four public log benchmarks show that IP-Log achieves competitive anomaly detection performance while additionally providing index-aligned evidence for anomaly interpretation. The ablation results further verify the effectiveness of the proposed hierarchical aggregation design for improving detection and attribution quality. These results demonstrate that preserving entity-level traceability during representation learning is beneficial for building more actionable log anomaly detection models. In future work, we will extend IP-Log to online and streaming settings to support real-time anomaly detection and evidence retrieval under continuously evolving log streams.

References

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

[2] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2017, pp. 1285–1298.

[3] H. Guo, S. Yuan, and X. Wu, "Logbert: Log anomaly detection via bert," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

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

[5] Y. Xie, H. Zhang, and M. A. Babar, "LogGD: Detecting anomalies from system logs with graph neural networks," in *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, 2022, pp. 299–310.

[6] C. Wang, D. Xu, and Z. Li, "Log2graphs: An unsupervised framework for log anomaly detection with efficient feature extraction," *arXiv preprint*, 2024.

[7] A. J. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, 2012.

[8] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. Chen, and F. Pianese, "System log parsing: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, pp. 8596–8614, 2023.

[9] J.-G. Lou, Q. Fu, S. Yang, J. Ye, and J. Li, "Mining invariants from console logs for system problem detection," in *Proceedings of the 2010 USENIX Annual Technical Conference (USENIX ATC)*, 2010.

[10] S. Rendle, "Factorization machines," in *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 995–1000.

[11] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[12] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "Eca-net: Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010. [Online]. Available: https://dl.acm.org/doi/10.5555/3295222.3295349

[14] R. Vaarandi, M. Kont, and M. Pihelgas, "Event log analysis with the LogCluster tool," in *MILCOM 2016 – 2016 IEEE Military Communications Conference*, 2016, pp. 982–987.