informatics

MDPI

Article

Reducing Trace-Complexity Bias in Microservice Anomaly Scoring via Weighted Residual Service Trace Vectors

Pavan Kumar Ramina and Murali Krishna Enduri *

Department of Computer Science and Engineering, SRM University-AP, Amaravati 522 240, India; pavankumar_ramina@srmap.edu.in

* Correspondence: muralikrishna.e@srmap.edu.in

Abstract

Distributed trace anomaly scoring must distinguish unusual service-path latency behavior from structural differences among traces. However, raw Service Trace Vector (STV) scores may increase with trace size and feature occupancy, potentially prioritizing structurally larger traces even when no anomaly has been injected. We evaluate this structural score dependence on held-out, non-injected traces from three TrainTicket configurations using five train–test splits per configuration. Raw STV + Isolation Forest scores show strong mean Pearson correlations with service count ($r = 0.868$), span count ($r = 0.820$), and the number of observed STV features ($r = 0.960$). Local Outlier Factor closely reproduces this dependence, whereas One-Class SVM reduces its Pearson magnitude, indicating a representation–scorer interaction rather than an effect specific to Isolation Forest. We propose WR-STV, which applies clipped feature-wise standardized residuals and log-frequency reliability weighting. Validation selects $k = 1$; therefore, the final configuration scores each trace using its maximum reliability-weighted residual. WR-STV reduces the mean Pearson correlations with service and span count to 0.296 and 0.238, respectively. Under the original controlled 30×, three-span perturbation setting, WR-STV improves within-dataset ROC-AUC from 0.605 to 0.814. An expanded 1.10×–30× study shows that subtle perturbations remain difficult and that ranking performance generally increases with severity. WR-STV requires approximately 0.022 ms of mean batched scoring time per trace, approximately 16× lower than the evaluated Isolation Forest baselines. These results support WR-STV for reducing structural score dependence in controlled STV-visible latency ranking; they do not establish performance for naturally occurring incidents or non-latency anomaly types.

Keywords: microservices; distributed tracing; anomaly detection; service trace vector; residual normalization; trace complexity bias; explainability

Check for updates

Academic Editors: Weitian Tong, Lei Chen and Lixin Li

Received: 24 August 2026
Revised: 28 September 2026
Accepted: 30 September 2026
Published: 9 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Microservice architectures decompose large applications into collections of independently deployed services that communicate through networked requests [1]. Although this design improves scalability, modularity, and deployment flexibility, it also complicates performance diagnosis because a single user request may traverse many services and generate a distributed execution path. Distributed tracing systems record this execution as a trace composed of spans, where each span captures a service operation, timing information, and parent–child relationships [2]. As microservice systems scale, manual inspection of traces becomes impractical, creating a need for automated methods that can identify

traces whose execution deviates from expected behavior and guide operators toward likely sources of latency or failure.

At the scale of modern microservice deployment, distributed tracing frameworks continuously collect large volumes of execution traces, making manual inspection challenging for human operators. Early industrial approaches often relied on manually maintained rules, including service-specific return-code rules, to identify abnormal executions. As services evolve, however, these rules require continual maintenance because the meaning and usage of service-specific return codes may change over time, reducing their long-term effectiveness [3]. These challenges have motivated learning-based trace anomaly detection methods that model the characteristics of normal trace executions and automatically identify traces whose behavior deviates from the learned patterns, allowing operators to prioritize investigation without continually redesigning rule sets. Learning-based trace anomaly detection depends heavily on how distributed traces are represented for machine learning. Previous studies have explored execution sequences, graph-based models, learned embeddings, and fixed-dimensional vector representations [3–5]. Among these, the Service Trace Vector (STV), introduced in TraceAnomaly [3,6], transforms variable-length distributed traces into fixed-dimensional feature vectors suitable for machine learning. In an STV, each feature corresponds to a specific service reached through a particular call path, while its value represents a latency-related characteristic associated with that service path (Section 3). Because every feature has a direct correspondence to an actual service path, STVs preserve service-level interpretability while providing a compact numerical representation of distributed traces. Furthermore, because STVs are fixed-dimensional feature vectors, they can be analyzed using a wide range of conventional statistical and machine learning techniques, making them a flexible representation for trace anomaly detection. However, the effectiveness of such approaches depends not only on the trace representation itself, but also on how anomaly scores are derived from the resulting feature vectors. Existing STV- and graph-based approaches have demonstrated strong anomaly detection performance across several benchmark microservice datasets. However, comparatively little attention has been given to understanding whether the anomaly scores produced by these methods are influenced by intrinsic structural properties of traces, such as execution size or call-path complexity.

We identify a systematic weakness in STV-based anomaly scoring: when an STV is scored using a generic unsupervised detector trained directly on raw, unnormalized feature values, the resulting anomaly score can become strongly influenced by trace structural characteristics, potentially obscuring feature-relative latency deviations. To better understand this phenomenon, we analyze held-out, non-injected test traces from three configuration variants of the `ts-order-service` microservice in the TrainTicket benchmark [7]. Across these datasets, the Pearson correlation between baseline STV + Isolation Forest anomaly scores and the number of distinct services ranges from $r = 0.968$ to 0.982, while the correlation with the number of spans ranges from $r = 0.854$ to 0.981 (Table 1). Because these measurements are obtained from held-out, non-injected traces, the observed correlations cannot be explained by successful detection of synthetic faults. Instead, they indicate that structural properties of a trace can have a substantial influence on anomaly scores. The revised analysis further shows that this relationship is closely reproduced under Local Outlier Factor and remains present, although weaker, under One-Class SVM, suggesting that the phenomenon is not specific to Isolation Forest. These observations motivate a scoring approach that is less dependent on trace structural occupancy while preserving sensitivity to service-path latency deviations. We propose **WR-STV** (Weighted Residual Service Trace Vector), which replaces each observed raw STV value with a clipped standardized residual, applies a log-frequency reliability weight, and uses top-k scoring. Validation

selects $k = 1$; therefore, the final evaluated configuration is a maximum reliability-weighted standardized-residual scorer. This design preserves service-path interpretability while reducing structural score dependence (Section 4).

Table 1. Pearson correlation between baseline STV + Isolation Forest anomaly score and trace-complexity metrics (held-out, non-injected traces).

	Dataset 1	Dataset 2	Dataset 3
num_services	0.977	0.982	0.968
num_spans	0.854	0.981	0.975
total_duration	0.605	0.698	0.672
max_span_duration	0.550	0.627	0.681
mean_span_duration	0.035	−0.034	0.089

The main contributions of this paper are

- **A cross-detector analysis of structural score dependence** on held-out, non-injected traces: In addition to service and span counts, the revision directly measures STV feature occupancy and evaluates raw STV scoring with Isolation Forest, LOF, and One-Class SVM.
- **WR-STV**, a lightweight residual-standardization and reliability-weighting method: The general formulation supports top-k aggregation, while validation selects $k = 1$, making the final scorer max-based.
- **An expanded controlled latency-perturbation study** covering f in $\{1.10, 1.25, 1.50, 2, 3, 5, 10, 20, 30\}$, one and three modified characterizes the method's sensitivity envelope rather than treating only large perturbations as representative.
- **Quantitative interpretability and failure analysis**: ranked service-path evidence, overlap against known injected paths, and a rule-based descriptive taxonomy of cases in which the injected trace is not ranked above its original.

2. Related Work

This section surveys how prior work represents distributed traces for anomaly detection and how it derives anomaly scores from those representations, then situates the present paper's focus—whether the scoring process itself is biased by trace structure—relative to that literature.

2.1. Distributed Trace Representations

The effectiveness of learning-based trace anomaly detection depends strongly on how distributed traces are represented before analysis. Distributed traces are variable-length execution records with service invocation paths, parent–child relationships, and heterogeneous latency behavior. Prior work has therefore explored several representations for trace analysis. TraceAnomaly represents each trace as a fixed-dimensional Service Trace Vector (STV), where features correspond to service call paths and values encode latency-related behavior [3]. Graph-based approaches instead preserve more of the trace topology. DeepTraLog constructs a trace event graph that combines invocation structure with log events [4,8], while TraceVAE models microservice traces using a graph variational autoencoder designed for tree-structured trace data [5]. Other recent approaches incorporate additional telemetry such as profiling metrics or resource utilization to enrich trace representations [9,10].

These representations involve different trade-offs. Fixed-dimensional vectors such as STVs are lightweight and directly interpretable because each feature maps to a concrete service path, but they may collapse structural information. Graph-based and multi-modal

models preserve more execution structure and operational context, but typically require more complex learning architectures and may reduce transparency of individual feature contributions. Our work builds on the interpretability and simplicity of STVs, but studies a different question: whether anomaly scores derived from raw STV features are systematically influenced by trace structural complexity.

2.2. Trace-Based Anomaly Detection and Localization

Early approaches to microservice anomaly detection often relied on manually defined rules or service-specific heuristics to identify abnormal executions [3,11]. As microservice systems increased in scale and complexity, these rule-based techniques became harder to maintain, motivating learning-based methods that model normal trace behavior directly from data. TraceAnomaly learns the distribution of normal STVs using service-level deep Bayesian networks and assigns higher anomaly scores to traces that deviate from the learned normal behavior [3]. Because STV features correspond to service paths, this representation also provides service-path-level diagnostic information.

Subsequent methods improve trace anomaly detection by using richer representations or more expressive models. DeepTraLog combines trace and log information in a graph neural architecture with a deep support vector data description objective [4,12]. TraceGra uses graph deep learning to model trace-based anomaly detection in microservice systems [13]. ServiceAnomaly combines distributed traces with profiling metrics in an annotated graph to detect abnormal microservice executions [9]. Recent work has also explored context-aware trace graph embeddings and resource-metric autoencoders for anomaly detection and localization [10,14,15].

These methods primarily focus on improving anomaly detection accuracy, capturing a richer trace structure, or enhancing fault localization. In contrast, our work focuses on the behavior of the anomaly scoring process itself. Specifically, we examine whether STV-based anomaly scores can be biased by structural properties of traces, such as the number of spans or services, even when the traces are operationally normal.

2.3. Research Gap: Bias in STV-Based Anomaly Scoring

Existing trace anomaly detection studies have advanced the state of the art through improved trace representations, graph neural architectures, deep generative models, and multi-modal telemetry fusion. However, they typically evaluate detection accuracy or localization quality, while the relationship between anomaly score and trace structural complexity is rarely examined directly. This leaves open an important question for STV-based anomaly scoring: does a high anomaly score reflect abnormal service-path behavior, or can it also reflect the fact that a trace is structurally larger than other traces?

To the best of our knowledge, prior STV-based trace anomaly detection studies have not explicitly quantified whether raw STV anomaly scores are correlated with trace-complexity measures such as span count and service count. Our work addresses this gap by measuring the dependency directly, proposing a lightweight residual scoring method designed to reduce it, and evaluating the resulting effect on both trace-complexity bias and controlled latency-anomaly scoring performance.

3. Background and Motivation

This section introduces the distributed-trace and Service Trace Vector concepts needed to define trace-complexity bias, then presents the empirical baseline evidence for that bias that motivates the WR-STV design in Section 4.

3.1. Distributed Traces

A distributed trace records the end-to-end execution of a single user request as it propagates through multiple interacting microservices [2]. Each trace consists of a collection of spans, where every span represents an individual service invocation together with execution metadata, including the service name, operation name, start time, duration, and parent–child relationship. The parent–child relationships among spans allow a trace to be interpreted as a causal execution structure, from which request paths across services can be reconstructed. Because traces vary in both execution path and structural complexity, they provide a rich source of information for performance analysis and anomaly detection.

3.2. Service Trace Vectors

Although distributed traces capture detailed execution behavior, their variable length and hierarchical structure make them unsuitable for direct use by conventional machine learning algorithms that require fixed-dimensional inputs. TraceAnomaly [3] addresses this limitation through the Service Trace Vector (STV), which maps each distributed trace into a fixed-dimensional feature vector while preserving the correspondence between features and service call paths.

Following TraceAnomaly, each feature $j = (\text{service}, \text{service_path})$ corresponds to a unique service reached through a particular root-to-node execution path observed during training. For a trace i, the feature value is defined as

$$x_{i,j} = \log(1 + d_{i,j}), \tag{1}$$

where $d_{i,j}$ denotes the maximum observed duration for feature j within trace i. When the same service-path feature appears multiple times in a trace, the maximum duration is retained. Features that are absent from a trace are assigned a sentinel value of -1, indicating that the corresponding service path was not executed. The logarithmic transformation reduces the dynamic range of span durations while preserving their relative ordering. Because every feature corresponds to a specific service path, the resulting representation remains directly interpretable and enables anomaly scores to be associated with concrete execution paths rather than latent embedding dimensions.

3.3. Motivation: Trace-Complexity Bias

The STV representation provides an interpretable numerical description of distributed traces, but it also introduces an important characteristic: structurally larger traces naturally populate more STV features than smaller traces. Consequently, traces containing more services or spans may appear more unusual to an anomaly scoring method, even when the underlying system behavior is treated as normal under the experimental protocol.

To investigate whether this structural difference influences anomaly scoring, we analyze the relationship between baseline STV anomaly scores and two measures of trace complexity: the number of distinct services and the total number of spans contained in each trace. The analysis is performed on real, non-injected test traces collected from three TrainTicket configurations [7].

As shown in Table 1, baseline anomaly scores exhibit consistently strong positive correlations with both complexity measures across all three datasets. Because these measurements are obtained from traces without injected anomalies, the observed correlations cannot be attributed to the successful detection of synthetic faults. Instead, they indicate that the structural properties of a trace can substantially influence anomaly scores even when the observed executions are traces treated as normal under the experimental protocol. The concern is not that all association with complexity must be zero, but that structural occupancy can dominate the score and thereby obscure feature-relative latency deviations.

Figure 1 visualizes this relationship by plotting baseline anomaly scores against trace structural count for held-out, non-injected traces. The overall trend demonstrates that larger traces tend to receive higher baseline anomaly scores, motivating the need for an anomaly scoring strategy that is less sensitive to structural complexity.

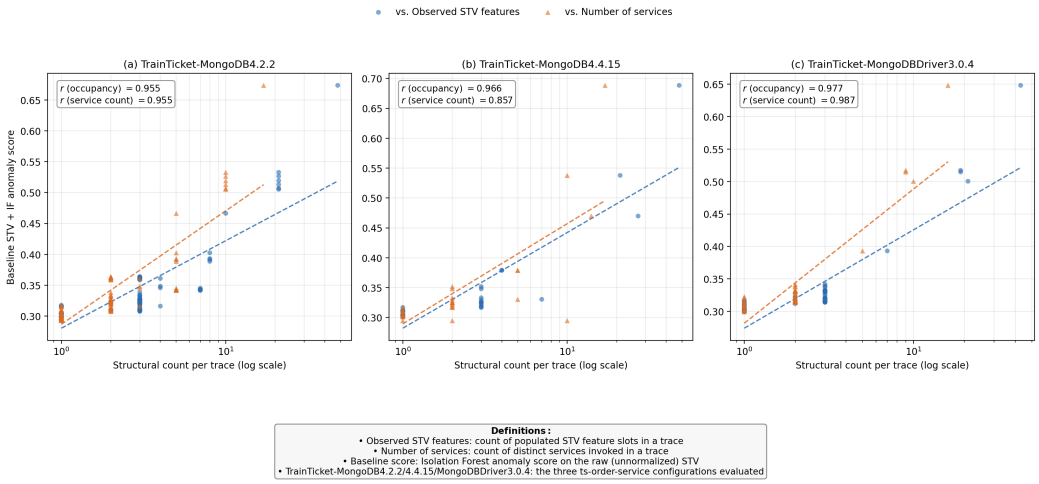


Figure 1. Baseline STV + Isolation Forest anomaly score versus trace structural count (observed STV features and number of services; dashed lines are ordinary-least-squares trend fits) on held-out, non-injected traces from the three TrainTicket `ts-order-service` configurations evaluated in this paper (representative seed-0 split; see Table 4 for the 5-seed mean). Each point represents one trace. The observed positive relationships indicate substantial structural score dependence in the baseline scoring configuration, even in the absence of injected anomalies.

3.4. Design Intuition

The observations above suggest that anomaly scoring should depend on how unusual each feature is relative to its own expected behavior rather than on its absolute magnitude alone. Features whose latency values are consistent with their historical behavior should contribute little to the final anomaly score, even when they occur in a large trace. In contrast, features exhibiting unusually large deviations from their own learned normal behavior should receive greater emphasis. This intuition motivates the Weighted Residual Service Trace Vector (WR-STV) method introduced in the next section, which preserves the interpretability of the original STV representation while reducing the influence of trace structural complexity on anomaly scoring.

4. Proposed Method

This section defines WR-STV in the order it is computed: STV construction, residual normalization, reliability weighting, and top-k aggregation, followed by the request-aware ablation variant.

4.1. Overview

WR-STV is an STV-based anomaly scoring framework that transforms raw service-path latency features into reliability-weighted residual contributions. The method computes an anomaly score for each trace in five stages, all derived from statistics learned from the training split only: (1) construct the baseline Service Trace Vector (STV), (2) compute per-feature training statistics, (3) convert observed values into clipped standardized residuals, (4) weight each residual by a reliability term reflecting how often that feature was observed during training, and (5) aggregate the top-k weighted residuals into a single trace-level score. Figure 2 summarizes the full pipeline.

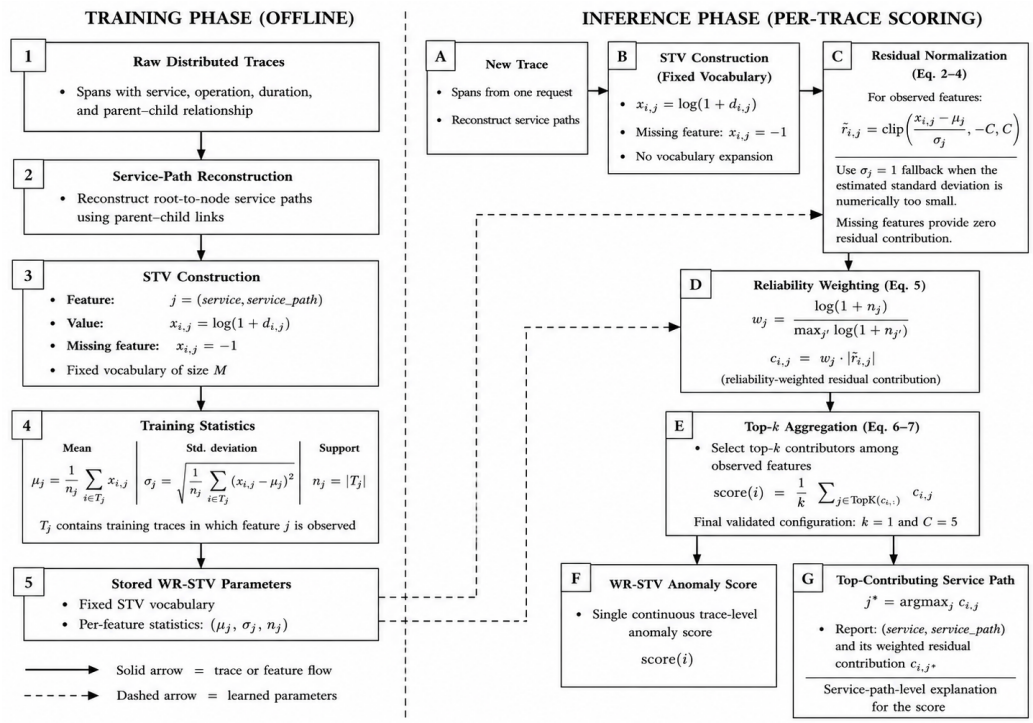


Figure 2. Overview of the WR-STV anomaly scoring pipeline. Feature statistics and reliability weights are learned from training traces only and applied to each test trace using a fixed STV vocabulary. Top-k weighted residual aggregation produces a trace-level anomaly score and service-path-level explanatory evidence.

Given a distributed trace, WR-STV produces a single anomaly score that quantifies the degree to which the trace deviates from the statistical behavior learned from training traces. Rather than modifying the underlying trace representation, WR-STV retains the original STV representation and instead modifies the anomaly scoring process through feature-wise residual standardization, reliability weighting, and top-k aggregation. This design preserves the interpretability of STV features while reducing the influence of trace structural complexity on the final anomaly score.

4.2. Service Trace Vector Construction

Following the Service Trace Vector (STV) representation introduced by TraceAnomaly [3], each trace is mapped to a fixed-vocabulary vector, where feature $j = (\text{service}, \text{service_path})$ is valued by $x_{i,j} = \log(1 + d_{i,j})$. A sentinel value of -1 is used when feature j is not observed in trace i. The feature vocabulary and all statistics described in the following subsections are learned exclusively from the training split. During inference, the same vocabulary is reused without modification, and service paths not observed during training are treated as absent rather than added as new features. Using a fixed vocabulary ensures that training and inference operate over a consistent feature space.

4.3. Residual Normalization

Raw STV values are not directly comparable across service-path features because different paths naturally exhibit different latency scales and variability. WR-STV therefore normalizes each observed feature relative to its own training-time behavior. For each

feature j, we compute the training-time mean and standard deviation over the training traces in which j is observed:

$$\mu_j = \frac{1}{n_j} \sum_{i \in \mathcal{T}_j} x_{i,j}, \tag{2}$$

$$\sigma_j = \sqrt{\frac{1}{n_j} \sum_{i \in \mathcal{T}_j} (x_{i,j} - \mu_j)^2}, \tag{3}$$

where $\mathcal{T}_j$ is the set of training traces in which feature j is present and $n_j = |\mathcal{T}_j|$. If a feature has insufficient observations or a numerically small standard deviation, the implementation uses a unit standard deviation fallback to avoid unstable division.

For an observed value $x_{i,j}$ at inference time, the clipped standardized residual is

$$\tilde{r}_{i,j} = \text{clip}\left(\frac{x_{i,j} - \mu_j}{\sigma_j}, -C, C \right), \tag{4}$$

where C denotes the residual clipping threshold. The clipping operation improves numerical stability for features with very small standard deviations and prevents individual feature deviations from disproportionately influencing the final anomaly score. Features absent from a trace are assigned zero residual contribution and therefore do not increase the anomaly score.

4.4. Reliability Weighting

Features observed infrequently during training provide fewer samples for estimating their underlying latency distribution, resulting in greater uncertainty in the estimated mean and standard deviation. Consequently, standardized residuals associated with such features are more susceptible to sampling variability. To reduce the influence of these uncertain estimates, each feature is assigned a reliability weight based on its training-time observation frequency:

$$w_j = \frac{\log(1 + n_j)}{\max_{j'} \log(1 + n_{j'})}. \tag{5}$$

The logarithmic transformation compresses large count differences while preserving the ordering of feature support. Thus, $w_j \in (0, 1]$, with the most frequently observed feature receiving weight 1. Features observed more frequently during training receive larger weights, whereas infrequently observed features are down-weighted, reducing their influence on the final anomaly score.

4.5. Top-k Weighted Residual Scoring

The weighted contribution of feature j in trace i is defined as

$$c_{i,j} = w_j |\tilde{r}_{i,j}|. \tag{6}$$

Larger standardized deviations from learned normal behavior produce larger contributions, moderated by the feature reliability weight.

Rather than summing all feature contributions, which can reintroduce dependence on the number of populated features in a trace, WR-STV aggregates only the strongest residual evidence. The final anomaly score is computed as the average of the k largest weighted contributions:

$$\text{score}(i) = \frac{1}{k} \sum_{j \in \text{Top}_k(c_{i,:})} c_{i,j}. \tag{7}$$

This aggregation emphasizes the most significant service-path deviations while limiting the influence of numerous low-magnitude residuals that are unlikely to indicate anomalous behavior. The value of k is selected through validation, as described in Section 5.6. In the final configuration, we use $k = 1$. Under this setting, the anomaly score corresponds to the largest weighted residual within the trace, and the associated service-path feature is reported as the explanation for the score in the interpretability analysis presented in Section 6.6.

4.6. Request-Aware Calibration (Ablation Variant)

As an ablation study, we also evaluate a request-aware variant in which the statistics in Equations (2) and (3) are computed separately for each root operation when sufficient training samples are available. When insufficient data exist for a particular request type, the corresponding global statistics are used instead.

Although this variant further reduces the observed correlation between anomaly score and trace complexity (Table 4), it does not consistently improve anomaly scoring performance. Therefore, we report it as an ablation rather than the primary WR-STV method.

5. Experimental Setup

This section describes the datasets, preprocessing, synthetic injection procedure, baseline methods, metrics, and hyperparameter selection used to produce the results in Section 6.

5.1. Datasets

We evaluate WR-STV using three trace sets collected from the TrainTicket benchmark microservice system [7]. Each trace set corresponds to a distinct configuration of `ts-order-service`:`ts-order-service_mongodb_4.2.2`, `ts-order-service_mongodb_4.4.15`, and `ts-order-service_3.0.4-mongodb-driver`. We refer to these configurations as Dataset 1, Dataset 2, and Dataset 3, respectively. TrainTicket is a benchmark microservice system designed for fault analysis and debugging studies, making it suitable for controlled experiments on microservice observability and anomaly scoring.

The three configurations vary the service version and MongoDB driver while preserving broadly similar service-level functionality. This makes them useful for both within-dataset evaluation and cross-dataset transfer analysis. In particular, the related but non-identical configurations allow us to study how WR-STV behaves when training and test traces share similar service structures but differ in observed latency distributions and service-path coverage.

Table 2 summarizes trace, span, and service counts, along with trace-graph quality checks performed before modeling. Across all three datasets, each trace contains exactly one root span, no non-root span has a missing parent, and no duplicate (traceID, spanID) pairs are present.

Table 2. Dataset statistics and trace-graph quality checks.

	Dataset 1	**Dataset 2**	**Dataset 3**
Traces	269	86	160
Spans	3788	1306	1993
Distinct services	32	33	33
Exactly 1 root span/trace	✓	✓	✓
Non-root spans w/missing parent	0	0	0
Duplicate (traceID, spanID) pairs	0	0	0

5.2. Preprocessing Pipeline

Spans are grouped by trace identifier, and each span's root-to-node service path is reconstructed by following parent–child relationships from the trace root. Following Equation (1), spans that share the same $(\text{service}, \text{service_path})$ feature within a single trace are aggregated into one STV feature by retaining the maximum observed duration. We use maximum aggregation because the slowest occurrence of a repeated service path is often the most actionable signal for latency anomaly triage, whereas mean aggregation may dilute localized delays.

For every experimental split, the STV vocabulary, feature-wise normalization statistics, and reliability weights are estimated only from the training traces. The learned vocabulary and statistics are then applied unchanged to validation and test traces. This protocol prevents test-set leakage and ensures that unseen service paths at inference time are treated as absent rather than being used to expand the feature space.

5.3. Synthetic Latency Injection

Because the trace sets do not provide per-trace ground-truth anomaly labels, we evaluate anomaly scoring using controlled synthetic latency injection. For a normal test trace, an injected copy is created by multiplying the duration of selected STV-visible spans by an injection factor f. A span is considered STV-visible if its $(\text{service}, \text{service_path})$ feature appears in the training vocabulary. The original trace is assigned label 0, while the injected copy is assigned label 1. We evaluate two span-selection strategies:

- **Largest-visible injection:** selects up to m STV-visible spans with the largest original durations. This is the primary controlled latency-injection setting.
- **Random-visible injection:** selects up to m STV-visible spans uniformly at random, regardless of original duration. This setting tests whether WR-STV's advantage depends on always perturbing high-duration spans.

Unless otherwise specified, experiments use $f = 30$ and $m = 3$. To evaluate sensitivity to perturbation magnitude, we additionally perform injection-strength sensitivity analysis using $f \in \{1.10, 1.25, 1.50, 2, 3, 5, 10, 20, 30\}$ and $m \in \{1, 3\}$, as reported in Section 6.4. The smaller factors ($1.10\times$ to $3\times$) represent subtle-to-moderate controlled latency changes, whereas the larger factors ($5\times$ to $30\times$) are retained as stress conditions. Both injection strategies are restricted to STV-visible dimensions, meaning that they evaluate controlled latency deviations on service paths already observed during training. Consequently, the evaluation characterizes controlled STV-visible latency perturbations rather than anomalies involving novel service paths, missing spans, dropped calls, incorrect responses, or other structural trace changes. This is a deliberate scope limitation and is discussed further in Section 7.

5.4. Baseline Methods

We compare WR-STV against both raw-STV and residual-STV baselines:

- **Baseline STV + Isolation Forest:** the raw STV representation from Equation (1), without feature normalization, scored using Isolation Forest [16]. This is the primary baseline used to measure trace-complexity bias.
- **Enhanced STV + Isolation Forest:** z-score-normalized residual features from Equations (2)–(4), scored using Isolation Forest. This baseline isolates the effect of residual normalization independent of WR-STV's top-k scoring rule and reliability weighting.
- **Max-Z Residual STV:** the maximum absolute residual from Equation (4), using $k = 1$ and no reliability weighting. This isolates the contribution of the reliability weight in Equation (5).

- **Top-3 Residual STV:** the mean of the top three unweighted absolute residuals. This tests whether aggregating multiple residual contributors changes ranking performance without reliability weighting.
- **Residual STV + LOF / One-Class SVM:** the standardized residual representation scored using Local Outlier Factor [17] and One-Class SVM [18]. This compares WR-STV's simple top-k aggregation against classical detectors applied to the same residual feature space.
- **Raw STV + LOF / One-Class SVM:** the same two detectors applied directly to the unnormalized STV representation. This tests whether the behavior observed with Isolation Forest is specific to one detector family or also appears under other classical unsupervised methods.

5.5. Evaluation Metrics

We report ROC-AUC, average precision (AUPRC), F1 at an oracle threshold, precision@N_+, and pairwise win rate. Here, N_+ denotes the number of injected traces in an evaluation split. The oracle-threshold F1 is computed by labeling the top N_+ scored traces as anomalous and is therefore used only as a ranking diagnostic, not as a deployment-ready thresholding result.

Pairwise win rate is computed over matched $(\text{original}, \text{injected})$ trace pairs. A pair is counted as a win when the injected trace receives a strictly higher anomaly score than its corresponding original trace. This metric directly measures whether a scoring method ranks a controlled latency perturbation above the normal trace from which it was generated.

5.6. Hyperparameter Selection

The WR-STV top-k parameter and residual clip bound C are selected using an inner validation split. We search over $k \in \{1, 3, 5\}$ and $C \in \{3.0, 5.0, 10.0\}$. The validation split is drawn from the outer training set, with 30% of the outer training traces reserved for validation. Test traces are not used during hyperparameter selection. This procedure selects $k = 1$ and $C = 5.0$, which are used for all final WR-STV results (Table 3).

Table 3. Hyperparameter selection on a held-out validation split (independent of all test-set metrics reported elsewhere in this paper).

Hyperparameter	Selected Value
k (top-k features aggregated)	1
C (residual clip bound)	5.0

5.7. Implementation Details

All experiments are implemented in Python 3.13.15 using NumPy 2.1.3, pandas 2.2.3, and scikit-learn 1.6.1. Isolation Forest uses 200 estimators in all experiments. Local Outlier Factor is evaluated in novelty-detection mode, and One-Class SVM uses the radial-basis-function kernel with $\nu = 0.1$ and $\gamma = \texttt{scale}$. Unless otherwise specified, each dataset is split 65:35 into training and test traces. The split is performed uniformly at random at the trace level using the specified random seed, ensuring that individual traces appear in only one partition. Within-dataset experiments are repeated over five random seeds for each dataset, resulting in $3 \times 5 = 15$ runs.

The overall experimental protocol is as follows. First, STVs are constructed from the training split, and the training vocabulary is fixed. Second, WR-STV feature statistics and reliability weights are estimated from the same training split only. Third, the fitted scoring procedure is applied unchanged to held-out validation or test traces. Finally, synthetic

latency anomalies are injected only into evaluation traces, and metrics are computed over the resulting normal-vs-injected trace pairs.

6. Results and Discussion

This section reports results for eight research questions in turn: whether structural score dependence exists (RQ1), whether residual normalization and detector choice affect it (RQ2), whether WR-STV improves detection (RQ3), its sensitivity to injection strength (RQ4), cross-dataset generalization (RQ5), explainability (RQ6), failure modes (RQ7), and computational cost (RQ8).

6.1. RQ1: Does Trace-Complexity Bias Exist?

The baseline results indicate that trace-complexity bias is substantial. Table 1 shows that anomaly scores produced by the baseline STV representation combined with Isolation Forest are strongly correlated with trace structural complexity. Across the three datasets, the Pearson correlation between anomaly score and the number of services ranges from $r = 0.968$ to 0.982, while the correlation with the number of spans ranges from $r = 0.854$ to 0.981. These consistently high correlations indicate that structurally larger traces receive systematically higher anomaly scores even in the absence of injected anomalies. The revised analysis additionally measures the number of observed STV features, which directly represents STV feature occupancy. Averaged across the three datasets and five train–test splits per dataset, raw STV + Isolation Forest has a mean Pearson correlation of $r = 0.960$ with the number of observed STV features.

These results indicate substantial complexity-related score dependence in the baseline scoring process and provide the primary motivation for WR-STV. They do not imply that every association between trace complexity and operational abnormality is spurious. Rather, the concern is that structural occupancy can substantially influence the score among held-out, non-injected traces and may obscure feature-relative latency deviations.

6.2. RQ2: Does Residual Normalization Reduce It?

Residual normalization substantially reduces, but does not eliminate, trace-complexity bias. Table 4 shows that WR-STV considerably reduces the dependence of anomaly scores on trace structural complexity. Averaged across datasets and seeds, the Pearson correlation with the number of services decreases from $r = 0.868$ under the baseline detector to $r = 0.296$ using WR-STV and further to $r = 0.274$ under the request-aware variant. Similarly, the correlation with the number of spans decreases from $r = 0.820$ to $r = 0.238$ and $r = 0.218$, respectively.

Table 4. Mean Pearson correlation between anomaly score and trace structural measures over three datasets and five train–test splits per dataset (15 runs), evaluated on held-out, non-injected traces using the sentinel representation for absent raw-STV features.

Method	Corr. Num_Services	Corr. Num_Spans
Baseline STV + IF	0.868	0.820
Raw STV + LOF	0.872	0.805
Raw STV + One-Class SVM	0.464	0.331
Enhanced STV + IF	0.835	0.779
Residual STV + LOF	0.699	0.629
Residual STV + One-Class SVM	0.394	0.327
Global Residual STV (unweighted)	0.383	0.324
WR-STV (weighted, $k = 1$)	**0.296**	**0.238**
Request-Aware WR-STV	0.274	0.218

The additional detector analysis shows that the observed relationship is not specific to Isolation Forest. Raw STV scored with Local Outlier Factor exhibits similarly strong correlations with service and span counts, whereas One-Class SVM produces weaker, but still positive, Pearson correlations. Detector choice therefore affects the magnitude of the relationship. The evidence is best interpreted as a representation–scorer interaction rather than as a property caused exclusively by either raw STV or Isolation Forest.

The number of observed STV features provides a direct measure of representation occupancy. Its mean Pearson correlation is $r = 0.960$ for raw STV + Isolation Forest, $r = 0.935$ for raw STV + LOF, and $r = 0.517$ for raw STV + One-Class SVM. WR-STV reduces this value to $r = 0.368$. A zero-filled missing-feature robustness analysis produces the same qualitative conclusion as the sentinel-policy analysis reported here.

The remaining positive correlation is expected because larger traces naturally provide more opportunities for at least one extreme feature residual to occur. Consequently, WR-STV should be interpreted as substantially reducing, rather than completely removing, complexity-related score dependence while preserving sensitivity to path-visible latency deviations.

6.3. RQ3: Does WR-STV Improve Anomaly Scoring and How Does It Compare with Baselines?

Tables 5 and 6 report the primary anomaly scoring comparison. Compared with baseline STV + Isolation Forest, WR-STV improves the within-dataset ROC-AUC from 0.605 to 0.814. The cross-dataset ROC-AUC also improves from 0.622 to 0.803. The within-dataset comparison has a large paired effect size across the evaluated dataset–seed rows. The cross-dataset comparison also shows a large paired effect, but its associated p-values are exploratory because the six directed pairs arise from only three underlying datasets.

Table 5. Main detection results: baseline vs. WR-STV, largest-visible injection ($f = 30$, $m = 3$), mean over datasets and seeds. F1 uses an oracle top-N_+ threshold and is reported only as a ranking diagnostic; it does not represent performance under an operationally selected threshold.

	ROC-AUC	AUPRC	F1
Within-dataset			
Baseline STV + IF	0.605	0.574	0.583
WR-STV (k = 1, C = 5)	**0.814**	**0.812**	**0.738**
Cross-dataset (six directed pairs)			
Baseline STV + IF	0.622	0.573	0.602
WR-STV (k = 1, C = 5)	**0.803**	**0.804**	**0.722**

Table 6. Paired comparisons between WR-STV and baseline STV + Isolation Forest. The reported p-values are exploratory and unadjusted for multiple comparisons. The 15 within-dataset rows are repeated splits of three datasets and are not equivalent to 15 independent datasets. The six directed transfer pairs are derived from three underlying datasets and are not independent observations; cross-dataset inference is therefore treated as exploratory. F1 uses an oracle top-N_+ threshold and is a ranking diagnostic.

	Gain	Cohen's d_z	Paired-t p	Wilcoxon p
Within-dataset ($n = 15$ seed–dataset rows)				
ROC-AUC	+0.209	2.78	3.68×10^{-8}	6.1×10^{-5}
AUPRC	+0.237	3.79	6.87×10^{-10}	6.1×10^{-5}
F1	+0.155	2.70	5.34×10^{-8}	9.82×10^{-4}
Cross-dataset ($n = 6$ directed pairs)				
ROC-AUC	+0.182	1.92	0.0053	0.03125
AUPRC	+0.231	2.67	0.00126	0.03125
F1	+0.120	1.69	0.00893	0.03125

These results indicate that the primary performance improvement arises from feature-wise residual normalization rather than from replacing one unsupervised detector with another. Standardizing each service-path feature relative to its learned latency distribution produces a scoring space that is substantially less influenced by trace structural complexity, allowing latency deviations to be identified more consistently across heterogeneous traces. The ablation results should be interpreted as evidence for residual normalization as the primary component: the additional reliability weighting is metric-dependent and is not statistically significant for ROC-AUC or F1 in the reported comparison. The reported p-values are exploratory and are not adjusted for the multiple comparisons across metrics, methods, and transfer pairs.

Table 7 further compares WR-STV with classical baselines operating on the same residual representation. Although WR-STV achieves the highest overall ROC-AUC (0.811), the margin over the unweighted Max-Z Residual STV (0.798) is relatively small.

Table 7. Detection performance vs. classical baselines, random-visible injection ($f = 30$), mean ROC-AUC.

Method	ROC-AUC
WR-STV ($k = 1$, $C = 5$)	**0.811**
Max-Z Residual STV (unweighted)	0.798
Residual STV + LOF	0.785
Residual STV + One-Class SVM	0.775
Top-3 Residual STV (unweighted)	0.771
Raw STV + One-Class SVM	0.761
Raw STV + LOF	0.707
Enhanced STV + IF	0.673
Baseline STV + IF	0.604

Table 8 shows that this difference is not statistically significant for ROC-AUC ($p = 0.108$) or F1 ($p = 0.735$), while only Average Precision reaches statistical significance. This suggests that most of WR-STV's performance gain originates from residual normalization, whereas reliability weighting provides a smaller, metric-dependent incremental improvement.

Table 8. Paired comparisons between WR-STV and its two closest residual-space baselines under random-visible injection. The reported p-values are exploratory and unadjusted. F1 uses an oracle top-N_+ threshold and is reported only as a ranking diagnostic.

Comparison	Metric	Diff.	p (Paired-t / Wilcoxon)
vs. Max-Z Residual STV	ROC-AUC	+0.013	0.108/0.135
	AUPRC	+0.065	5.1×10^{-5}/–
	F1	+0.003	0.735/–
vs. Residual STV + LOF	ROC-AUC	+0.025	0.0026/–
	F1	+0.007	0.32/–

6.4. RQ4: How Robust Is WR-STV to Injection Strength?

In Figure 3, the expanded sensitivity analysis includes perturbation factors from 1.10× to 30×, thereby characterizing both subtle controlled changes and high-severity stress conditions. At 1.10× to 1.50×, ranking is close to chance for several method and injection configurations, indicating that subtle latency changes remain difficult. Performance generally improves as perturbation severity increases, although strict monotonicity is not assumed for every method, strategy, and modified-span count.

The revised analysis also evaluates both one and three modified spans under largest-visible and random-visible selection. Random-visible selection reduces the concern that the experiment always targets spans with the largest original durations and therefore favours a maximum-residual scoring rule. The results characterize sensitivity to controlled STV-visible latency perturbations; they do not establish performance for naturally occurring incidents, novel paths, missing spans, incorrect responses, or structural trace changes.

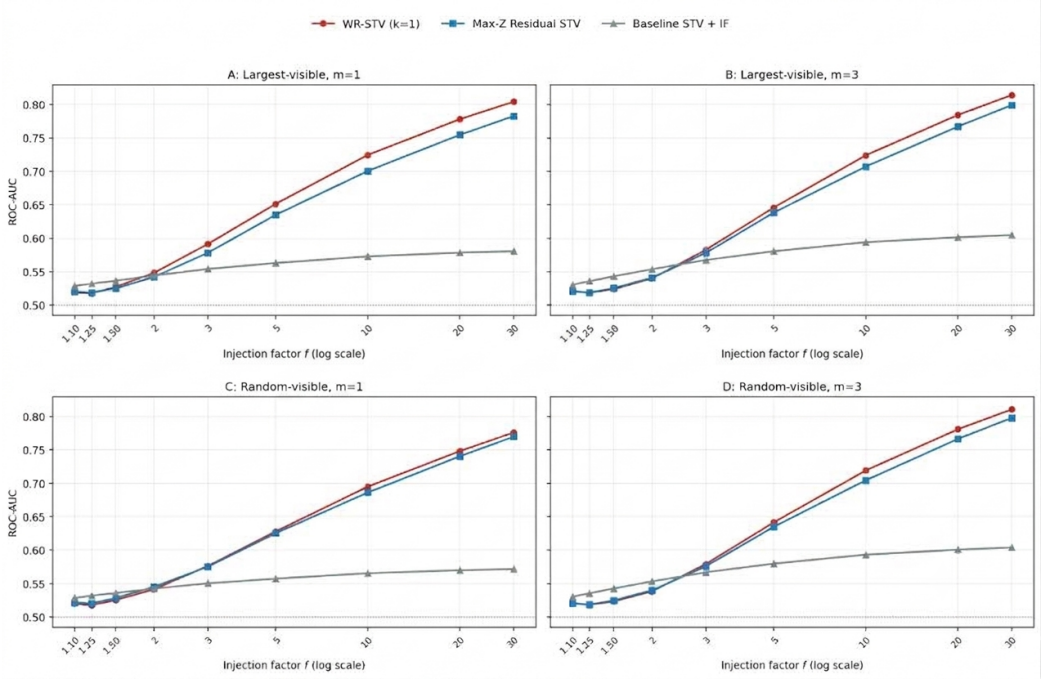


Figure 3. Sensitivity to controlled multiplicative latency perturbations at $f \in \{1.10, 1.25, 1.50, 2, 3, 5, 10, 20, 30\}\times$, under largest-visible and random-visible span selection, with one and three modified spans. Curves show means over three datasets and five train–test splits per dataset.

6.5. RQ5: Does WR-STV Generalize Across Datasets?

Table 9 reports all six directed train–test pairs. Transfer performance varies across pairs, with ROC-AUC ranging from 0.611 to 0.891. The strongest transfer is observed for Dataset 1 → Dataset 2 and Dataset 1 → Dataset 3 (both 0.891), while the weakest transfer occurs for Dataset 2 → Dataset 1 (0.611). We checked whether this degradation is explained by unseen service paths at test time and found that 97.8% of test-trace STV-visible events were covered by the Dataset 2 training vocabulary. This suggests that the weak transfer pair is not primarily a coverage problem.

Table 9. WR-STV cross-dataset transfer, all six directed train→test pairs (largest-visible injection). F1 uses an oracle top-N_+ threshold and is reported only as a ranking diagnostic; it does not represent performance under an operationally selected threshold.

Train → Test	ROC-AUC	AUPRC	F1
Dataset 1 → Dataset 2	0.891	0.888	0.800
Dataset 1 → Dataset 3	0.891	0.892	0.796
Dataset 2 → Dataset 1	0.611	0.644	0.587
Dataset 2 → Dataset 3	0.790	0.789	0.680
Dataset 3 → Dataset 1	0.760	0.752	0.684
Dataset 3 → Dataset 2	0.872	0.861	0.785
Mean (6 pairs)	**0.803**	**0.804**	**0.722**

Table 10 instead shows that this pair has the largest latency distribution shift on shared features (p_{90} standardized shift = 2.74, compared with ≤ 1.71 for the stronger pairs shown). The exploratory correlation across all six directed pairs (Spearman $r = -0.83$, $n = 6$) is more consistent with distribution shift than with vocabulary coverage as an explanation for transfer degradation. This suggests that successful cross-dataset transfer may depend more strongly on preserving the latency distributions of shared service paths than on maintaining identical feature vocabularies. Since this conclusion is based on only six directed transfer pairs, it should be regarded as exploratory rather than definitive.

Table 10. Latency distribution shift vs. cross-dataset transfer performance (shared STV-visible features between train and test). Spearman correlation (p_{90} standardized shift vs. ROC-AUC, all 6 directed pairs): $r = -0.829$, $p = 0.0416$. This correlation is exploratory: the six directed pairs are drawn from only three underlying datasets and are not independent samples, so the p-value above should not be read as confirmatory.

Train → Test	Wtd. Mean Wasserstein	p_{90} std. shift	ROC-AUC
Dataset 2 → Dataset 1	0.973	2.737	0.611
Dataset 1 → Dataset 3	0.872	1.515	0.891
Dataset 1 → Dataset 2	0.772	1.707	0.891

6.6. RQ6: Can WR-STV Explain Anomalies?

Table 11 shows that WR-STV's top-3 contributing service paths recover the actual injected service paths with 93.7% service overlap and 91.5% path overlap, aggregated across all cross-dataset pairs. This result is conditional on the injected anomaly being among the largest residual signals in the trace. The same scoring mechanism used for anomaly scoring is also used for localization, so this result should be interpreted as evidence that WR-STV can point to anomalies it scores highly, not as an independently validated localization capability for anomalies of unknown type.

Table 11. WR-STV explanation overlap against known injected paths (aggregate cross-dataset, largest-visible injection).

Top-k	Service Overlap	Path Overlap
Top-1	0.638	0.500
Top-3	0.937	0.915
Top-5	0.967	0.946
Top-10	0.998	0.992

6.7. RQ7: Where Does WR-STV Fail?

Table 12 categorizes the 279 unique cases, after deduplication from 1395 raw pairwise rows, where WR-STV did not rank an injected trace above its original. The two dominant categories, pre-existing high normal score (44.4%) and very short traces with limited path evidence (37.6%, mean ≈ 1.3 spans), together account for 82% of failures.

These categories provide plausible descriptive interpretations of the observed failures rather than causal explanations. Although a failed case may satisfy more than one underlying condition, the reported taxonomy assigns one primary category according to a fixed rule order. An original trace with an already elevated anomaly score can be difficult for its injected counterpart to outrank, while a trace containing only one or two spans provides limited STV-visible evidence from which to identify a feature-relative latency deviation.

The short-trace finding is particularly relevant to the validated $k = 1$ configuration. With $k = 1$, WR-STV scores a trace using its maximum reliability-weighted clipped standardized residual. This avoids accumulating numerous small contributions in structurally

larger traces, but traces with very few observable service-path features provide fewer opportunities for a sufficiently strong and reliable residual to emerge.

Table 12. WR-STV failure taxonomy: cases where the injected trace did not score above its original (279 unique failures after deduplication from 1395 raw pairwise failure rows).

Category	Count	%
Pre-existing high normal score	124	44.44
Very short trace, limited evidence	105	37.63
Score unchanged after injection	19	6.81
Low residual after injection	12	4.30
Uncategorized (distribution/ranking)	10	3.58
Low reliability weight	5	1.79
High train-variance path	4	1.43

These observed failure conditions suggest that many WR-STV ranking failures can arise when the available service-path evidence is limited or when the original trace already receives a high score. In practice, traces with very few observable service-path features may require confidence estimates, abstention rules, or complementary detectors rather than being treated as equally reliable scoring targets.

6.8. RQ8: Is WR-STV Computationally Lightweight?

Although computational efficiency is not the primary contribution of this work, WR-STV also offers practical scoring advantages.

Table 13 shows that WR-STV requires approximately 0.022 ms of mean batched scoring time per trace, compared with approximately 0.36 ms for Isolation Forest-based baselines, corresponding to roughly a 16× reduction in scoring time. This improvement is expected because WR-STV performs lightweight statistical computations using precomputed feature statistics and avoids ensemble-tree inference during deployment. Consequently, WR-STV has favorable computational properties for high-throughput scoring scenarios, although full online deployment would require additional thresholding and operational validation.

Table 13. Runtime comparison, mean over datasets and seeds. Per-trace values represent mean batched scoring time divided by the number of scored traces.

Method	Total (s)	Per-Trace Scoring (ms)
WR-STV	**0.171**	**0.022**
Enhanced STV + IF	0.576	0.3627
Baseline STV + IF	0.607	0.3647

6.9. Summary of Findings

Across the evaluated research questions, the results support the central hypothesis of this work. Raw STV representations scored with Isolation Forest exhibit strong structural score dependence with trace complexity. WR-STV substantially reduces this dependence through feature-wise residual normalization and improves controlled latency-anomaly ranking in within-dataset experiments, with promising but more limited evidence for cross-dataset transfer. The method also provides service-path-level interpretability, predictable failure modes, and lightweight scoring cost. Overall, the results support WR-STV as a bias-aware anomaly scoring framework for controlled STV-visible latency deviations rather than a general-purpose solution to all microservice anomaly types.

7. Threats to Validity

This section discusses internal, construct, external, and statistical-conclusion validity threats to the evaluation reported in Section 6.

7.1. Internal Validity

The main internal-validity risk is optimistic bias from design or hyperparameter choices. The top-k parameter and residual clip bound are selected on an inner validation split independent of the held-out test traces. The vocabulary, residual statistics, and reliability weights are estimated from training traces only. Perturbations are introduced only after splitting and never enter model fitting or hyperparameter selection. The revised sensitivity study evaluates $f \in \{1.10, 1.25, 1.50, 2, 3, 5, 10, 20, 30\}$, $m \in \{1, 3\}$, and both largest-visible and random-visible selection over five splits for each of the three datasets.

7.2. Construct Validity

The evaluation measures controlled synthetic latency anomalies rather than naturally occurring production failures. Injected anomalies are created by multiplying selected span durations, so the evaluation primarily tests sensitivity to path-visible latency deviations. It does not test non-latency anomalies such as incorrect responses, missing calls, dropped spans, novel service paths, or structural changes in the trace graph. In addition, both largest-visible and random-visible injection strategies are constrained to STV-visible service paths, i.e., features already present in the training vocabulary. Therefore, the results characterize WR-STV under controlled STV-visible latency perturbations rather than all possible microservice anomaly types.

The reported F1 score is computed using an oracle threshold that labels the top N_+ scored traces as anomalous, where N_+ is the number of injected traces in the evaluation split. This metric should be interpreted as a ranking diagnostic rather than as expected deployment precision or recall under an unknown operational threshold.

7.3. External Validity

The evaluation is limited to three configuration variants of `ts-order-service` within the TrainTicket benchmark. Although these configurations provide useful version-induced variation, they represent a single benchmark system and a single target service. The results may not generalize to other services, other microservice benchmarks, production workloads, or environments with substantially different tracing quality, traffic patterns, or service-call structures. Additional evaluation on other services, benchmark systems, and real production incident data is required before making broader generalization claims.

7.4. Statistical Conclusion Validity

Within-dataset statistical analyses are based on repeated experiments over five random seeds for each of three datasets, yielding $n = 15$ seed-dataset runs. These repeated runs improve robustness to random train-test splitting, but they are not equivalent to fifteen independent datasets. Cross-dataset transfer analysis is even more limited: it uses only six directed train-test pairs derived from the same three datasets. Consequently, cross-dataset statistical tests should be interpreted as exploratory evidence supporting observed trends rather than definitive statistical confirmation.

8. Conclusions and Future Work

This paper studied structural score dependence in a common microservice anomaly-scoring pattern: Service Trace Vector representations scored directly by an unnormalized unsupervised detector. On held-out, non-injected TrainTicket traces, baseline STV + Isola-

tion Forest anomaly scores correlate strongly with service count and span count, with per-dataset correlations of approximately $r = 0.85$–0.98. The revised analysis further shows a mean correlation of $r = 0.960$ with the number of observed STV features, which directly measures representation occupancy. Raw STV + LOF closely reproduces the strong structural relationship, whereas One-Class SVM produces weaker Pearson correlations. These findings indicate that detector choice affects the magnitude of the relationship and support an interpretation as a representation–scorer interaction rather than an effect attributable exclusively to Isolation Forest. They do not imply that every association between trace complexity and operational abnormality is spurious; instead, the concern is that structural occupancy may substantially influence the score and obscure feature-relative latency deviations.

To address this issue, we proposed WR-STV, a lightweight STV-based scoring method that applies per-feature residual standardization, residual clipping, and reliability weighting. The general method supports top-k scoring, but validation selected $k = 1$ and $C = 5$. Therefore, the final evaluated configuration is a max-based scorer that assigns each trace its maximum reliability-weighted clipped standardized residual. This choice avoids accumulating numerous small contributions in structurally larger traces, although very short traces with few observable service-path features can provide limited evidence for the maximum-residual scorer.

Across the evaluated datasets, WR-STV substantially reduces the measured correlation between anomaly score and trace complexity. The mean Pearson correlation with service count decreases from $r = 0.868$ under raw STV + Isolation Forest to $r = 0.296$ using WR-STV, while the correlation with span count decreases from $r = 0.820$ to $r = 0.238$. The corresponding correlation with the number of observed STV features decreases from $r = 0.960$ to $r = 0.368$. These findings show that WR-STV reduces, but does not eliminate, complexity-related score dependence.

Under the original controlled STV-visible latency-perturbation setting with $f = 30$ and $m = 3$, WR-STV improves within-dataset ROC-AUC over baseline STV + Isolation Forest from 0.605 to 0.814. The expanded sensitivity study evaluates multiplicative factors from $1.10\times$ to $30\times$, one and three modified spans, and both largest-visible and random-visible selection. The results show that subtle perturbations around $1.10\times$–$1.50\times$ remain difficult and that ranking performance generally improves as perturbation severity increases, without implying strict monotonicity for every method and injection configuration.

Cross-dataset experiments show dataset-dependent transfer behavior, improving the six-pair mean ROC-AUC from 0.622 to 0.803, with pairwise ROC-AUC values ranging from 0.611 to 0.891. These results should be interpreted cautiously because they are based on only six directed train–test pairs derived from three underlying datasets and therefore do not constitute independent observations. In particular, Dataset 2 $\rightarrow$ Dataset 1 is weak (0.611), despite stronger transfer when Dataset 1 is the training source. Accordingly, the cross-dataset findings should be treated as exploratory evidence of transfer behavior rather than confirmatory statistical evidence.

The ablation results indicate that most of the anomaly-ranking improvement comes from feature-wise residual normalization, while the proposed reliability weighting provides a smaller, metric-dependent refinement. WR-STV also provides interpretable service-path-level evidence, with top-3 contributing service paths recovering more than 90% of known injected paths in the evaluated setting. The top-3 explanation list is a diagnostic ranking of contributors rather than the set of features jointly used by the final $k = 1$ anomaly score. Runtime analysis further shows that WR-STV requires approximately 0.022 ms of mean batched scoring time per trace, corresponding to approximately $16\times$ lower mean batched scoring time per trace than the evaluated Isolation Forest-based baselines.

Overall, the results support WR-STV as a lightweight and interpretable approach for reducing structural score dependence in controlled STV-visible latency ranking. They do not establish general-purpose microservice anomaly detection or performance on naturally occurring production incidents. The evaluation is limited to three configurations of one benchmark service and does not cover incorrect responses, dropped calls, missing spans, novel service paths, or structural changes in the trace graph.

Future work will extend this study in four directions: evaluating anomalies beyond STV-visible latency perturbations, including incorrect responses, dropped calls, novel call paths, missing spans, and other structural trace anomalies; testing the generalization of the complexity-related score-dependence pattern on additional services, benchmark systems, and production-like traces beyond `ts-order-service`; attaching confidence estimates or abstention rules for very short, low-evidence traces, which the failure analysis identifies as an observed limitation; and validating WR-STV against real-world fault-injection or incident-labeled datasets to assess its behavior on anomalies not generated by this paper's evaluation procedure.

Author Contributions: Conceptualization, P.K.R.; methodology, P.K.R. and M.K.E.; formal analysis, P.K.R.; investigation, M.K.E. and P.K.R.; software, P.K.R.; validation, M.K.E.; data curation, M.K.E.; writing—original draft preparation, P.K.R. and M.K.E.; writing—review and editing, P.K.R. and M.K.E.; visualization, M.K.E.; supervision, P.K.R.; project administration, P.K.R.; funding acquisition, M.K.E. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by SRM University-AP, Andhra Pradesh, India. The authors gratefully acknowledge the financial support provided by the university.

Data Availability Statement: The source code, datasets, and reproduction scripts are openly available at https://github.com/pVnpNdu02/Reducing-Trace-Complexity-Bias-WR-STV (accessed on 23 August 2026).

Acknowledgments: During the preparation of this manuscript the authors used Claude Sonnet 5.5 (Anthropic) to assist with language, LaTeX preparation, and verification code for the algorithm and experiments. The authors reviewed and edited the content and take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dragoni, N.; Giallorenzo, S.; Lluch Lafuente, A.; Mazzara, M.; Montesi, F.; Mustafin, R.; Safina, L. Microservices: Yesterday, Today, and Tomorrow. In *Present and Ulterior Software Engineering*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 195–216. [CrossRef]
2. Sigelman, B.H.; Barroso, L.A.; Burrows, M.; Stephenson, P.; Plakal, M.; Beaver, D.; Jaspan, S.; Shanbhag, C. *Dapper, a Large-Scale Distributed Systems Tracing Infrastructure*; Technical Report dapper-2010-1; Google Research: Mountain View, CA, USA, 2010.
3. Liu, P.; Xu, H.; Ouyang, Q.; Jiao, R.; Chen, Z.; Zhang, S.; Yang, J.; Mo, L.; Zeng, J.; Xue, W.; et al. Unsupervised Detection of Microservice Trace Anomalies through Service-Level Deep Bayesian Networks. In *Proceedings of the IEEE International Symposium on Software Reliability Engineering*; ISSRE; IEEE: New York, NY, USA, 2020; pp. 48–58. [CrossRef]
4. Zhang, C.; Peng, X.; Sha, C.; Zhang, K.; Fu, Z.; Wu, X.; Lin, Q.; Zhang, D. DeepTraLog: Trace-Log Combined Microservice Anomaly Detection through Graph-Based Deep Learning. In *Proceedings of the 44th International Conference on Software Engineering*; ICSE; IEEE: New York, NY, USA, 2022; pp. 623–634. [CrossRef]
5. Xie, Z.; Xu, H.; Chen, W.; Li, W.; Jiang, H.; Su, L.; Wang, H.; Pei, D. Unsupervised Anomaly Detection on Microservice Traces through Graph Variational Autoencoders. In Proceedings of the ACM Web Conference WWW 2023, Austin, TX, USA, 30 April–4 May 2023.. [CrossRef]
6. Wei, X.; Sun, C.A.; Yang, P.; Zhang, X.Y.; Towey, D. TraLogAnomaly: A microservice system anomaly detection approach based on hybrid event sequences. *Sci. Comput. Program.* **2025**, *245*, 103303. [CrossRef]
7. Zhou, X.; Peng, X.; Xie, T.; Sun, J.; Ji, C.; Li, W.; Ding, D. Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study. *IEEE Trans. Softw. Eng.* **2021**, *47*, 243–260. [CrossRef]

8. Barata, L.M.; Sequeira, S.; Lopes, E.; Inácio, P.R.; Freire, M.M. Anomaly detection and root-cause identification in microservices: A survey. *Clust. Comput.* **2026**, *29*, 309. [CrossRef]
9. Panahandeh, M.; Hamou-Lhadj, A.; Hamdaqa, M.; Miller, J. ServiceAnomaly: An Anomaly Detection Approach in Microservices Using Distributed Traces and Profiling Metrics. *J. Syst. Softw.* **2024**, *209*, 111917. [CrossRef]
10. Al-Omari, T. Context-Aware Anomaly Detection in Microservices Using GCN-Encoded Trace Graphs and LSTM-AE Metrics with Local and Global Embeddings. *Eng. Technol. Appl. Sci. Res.* **2025**, *15*, 29277–29283. [CrossRef]
11. Khanahmadi, M.; Shameli-Sendi, A.; Jabbarifar, M.; Fournier, Q.; Dagenais, M. Detection of microservice-based software anomalies based on OpenTracing in cloud. *Softw. Pract. Exp.* **2023**, *53*, 1681–1699. [CrossRef]
12. Xu, H.; Pang, G.; Wang, Y.; Wang, Y. Deep isolation forest for anomaly detection. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 12591–12604. [CrossRef]
13. Chen, J.; Liu, F.; Jiang, J.; Zhong, G.; Xu, D.; Tan, Z.; Shi, S. TraceGra: A Trace-Based Anomaly Detection for Microservice Using Graph Deep Learning. *Comput. Commun.* **2023**, *204*, 109–117. [CrossRef]
14. Xing, S.; Wang, Y.; Liu, W. Multi-Dimensional Anomaly Detection and Fault Localization in Microservice Architectures: A Dual-Channel Deep Learning Approach with Causal Inference for Intelligent Sensing. *Sensors* **2025**, *25*, 3396. [CrossRef]
15. Kohyarnejadfard, I.; Aloise, D.; Azhari, S.V.; Dagenais, M.R. Anomaly detection in microservice environments using distributed tracing data analysis and NLP. *J. Cloud Comput.* **2022**, *11*, 25. [CrossRef]
16. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422. [CrossRef]
17. Breunig, M.M.; Kriegel, H.P.; Ng, R.T.; Sander, J. LOF: Identifying Density-Based Local Outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*; ACM: New York, NY, USA, 2000; pp. 93–104. [CrossRef]
18. Schölkopf, B.; Platt, J.C.; Shawe-Taylor, J.; Smola, A.J.; Williamson, R.C. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.* **2001**, *13*, 1443–1471. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.