2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)

Execution Trace Reconstruction Using Diffusion-Based Generative Models

Madeline Janecek
Brock University
St. Catharines, ON, Canada
mj17th@brocku.ca

Naser Ezzati-Jivan
Brock University
St. Catharines, ON, Canada
nezzati@brocku.ca

Abdelwahab Hamou-Lhadj
Concordia University
Montreal, QC, Canada
wahab.hamou-lhadj@concordia.ca

***Abstract*—Execution tracing is essential for understanding system and software behaviour, yet lost trace events can significantly compromise data integrity and analysis. Existing solutions for trace reconstruction often fail to fully leverage available data, particularly in complex and high-dimensional contexts. Recent advancements in generative artificial intelligence, particularly diffusion models, have set new benchmarks in image, audio, and natural language generation. This study conducts the first comprehensive evaluation of diffusion models for reconstructing incomplete trace event sequences. Using nine distinct datasets generated from the Phoronix Test Suite, we rigorously test these models on sequences of varying lengths and missing data ratios. Our results indicate that the SSSD^{S4} model, in particular, achieves superior performance, in terms of accuracy, perfect rate, and ROUGE-L score across diverse imputation scenarios. These findings underscore the potential of diffusion-based models to accurately reconstruct missing events, thereby maintaining data integrity and enhancing system monitoring and analysis.**

***Index Terms*—Software Analysis, Generative Models, System Call Sequences, Execution Trace Reconstruction**

I. Introduction

Execution tracing is a powerful technique used by software engineers to analyze the behaviour of software systems. Tracing facilitates a wide range of software engineering tasks, including program comprehension [1], performance analysis [2], fault diagnosis [3], and anomaly detection [4], [5]. However, as software systems grow in complexity and scale, the volume of trace data expands, which increases the risk of data loss due to limitations in trace recording infrastructures, such as buffer overflows or misconfigurations.

Common tracing systems, such as Perf [6], LTTng [7], ftrace [8] in Linux, and Event Tracing for Windows (ETW) [9] in Windows, utilize circular buffers to temporarily store events before they are permanently recorded, effectively minimizing system disturbance. However, unexpected surges in events, such as those caused by high system load, error conditions, security breaches, or during intensive debugging and testing phases, can cause these buffers to overflow, resulting in events being overwritten or discarded. As shown in Figure 1, which demonstrates how tracers typically manage trace events [7], ring buffers are used to sequentially store the events as they occur. These buffers are divided into smaller sub-buffers: when a sub-buffer is available (Scenario 1), events are temporarily held there until the consumer daemon transfers them to the trace. When all sub-buffers are filled (Scenario 2), no space remains to record new events, leading to data loss or the discarding of entire sub-buffers.

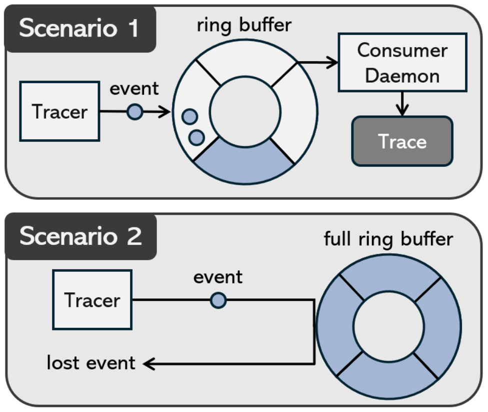


Fig. 1. A visual depiction of trace events being recorded. When there is an available sub-buffer (top), the event is stored until the consumer daemon writes the events to the trace. If every sub-buffer is full (bottom), the event is discarded.

To address this issue, existing tracers use three typical solutions: increasing the buffer size, blocking the traced application when the buffer is full, or allowing for some lost events. Increasing the buffer size is a common recommendation from tracing tools such as LTTng and EWT for users aiming to avoid missing events [10], [11]. However, this approach is limited by available system memory and does not entirely eliminate the risk of buffer overflow [12]. Some tracing tools, including LTTng, offer ways to block the application so that no events are ever lost [13]. However, blocking the traced application is undesirable as it alters the behaviour of the system that is being monitored, leading to an incorrect representation of the system execution [12]. Accepting lost events, which is the default configuration for most tracing tools [7]–[9], tends to compromise the integrity and reliability of the collected data, adversely affecting subsequent analyses [10].

In this paper, we propose a novel approach to reconstruct the content of traces by generating the missing events using generative models. Our approach aims to maintain the integrity of traces, without impacting the trace collection process. We achieve this by treating the problem of missing events as a data imputation task, which is defined as the process of replacing missing data with substituted values [14]. There is a substantial body of research on data imputation, ranging from traditional statistical methods to advanced machine learning models [14]. In recent years, data imputation methods that exploit the generative power of diffusion-based generative models have been shown to yield high accuracy [15], [16]. Diffusion models[1] are a class of deep generative models that have recently transformed the landscape of generative AI, particularly in the fields of image and audio synthesis [16]. Inspired by this success, we show how diffusion models can be employed to reconstruct execution trace content, specifically by predicting missing events using data from existing parts of the trace.

To demonstrate the effectiveness of our proposed method for trace reconstruction, we conduct experiments using four diffusion models and nine execution traces generated from benchmark applications across various domains. The results show that diffusion models are capable of reconstructing trace event sequences with an accuracy up to 81.71% and a ROUGE-L score of up to 90.90% across varying scenarios. We also show that diffusion models outperform traditional prediction-based models when applied to trace reconstruction.

The contributions of the paper are:

1) We introduce a novel application of diffusion-based generative models for reconstructing execution traces. To our knowledge, this is the first study that aims to reconstruct execution traces using data imputation methods.
2) We conduct a thorough evaluation of four diffusion-based generative models using nine execution traces, showing the adaptability and potential applicability of our method across different system behaviours and configurations.
3) We identify the key factors that influence the performance of these models, offering critical insights and highlighting promising directions for future research.

The remainder of this paper is structured as follows: In Section II, we provide an overview of diffusion models, their application for data imputation, and current literature examining execution trace reconstruction. In Section III, we present the study setup in which we describe the research questions, datasets, diffusion models used in this paper as well the evaluation metrics. In Section IV, we present the results of our study, and provide an in-depth analysis of the outcomes. Finally, we conclude our work with a summary of our findings and contributions in Section VII.

II. Background and Related Work

In this section, we present an overview of the key concepts used in this study and review the existing literature in the field.

A. Diffusion models

Diffusion models are a class of deep generative models that have revolutionized the landscape of generative AI, particularly in the realm of image and audio synthesis [16]. Despite the prevalence of diffusion models within the context of image generation, as evidenced by the popularity of models such as OpenAI's DALL·E 2 [17] and Google's Imagen [18], the versatility of diffusion models extends to a wide range of domains, including computational chemistry, natural language processing, temporal data modeling, and numerous other creative and technical fields [16].

The training process for diffusion models generally begins by following a forward process, in which random noise is progressively added to the training data over a series of T timesteps. This is denoted using Equations 1-2, where x_t represents the latent variable at step t, and β_t denotes the amount of noise added at that step.

$$q(x_1, ..., x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

$$= \prod_{t=1}^{T} \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t \mathbb{1})[x_t] \tag{2}$$

The core of the training process then involves having the model learn to iteratively remove this noise through a backward process, represented by Equation 3.

$$p_\theta(x_1, ..., x_T|x_0) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3}$$

After training has been completed, new samples may then be generated by feeding randomly sampled noise through the trained denoising mechanism. Depending on the task at hand, the backward process may be conditioned using additional information to steer the model towards the desired output.

B. Generative models for time series data reconstruction

Over time, several deep learning approaches have been proposed for time series data imputation. For example, many successful approaches such as BRITS [19], GRU-D [20], and M-RNN [21] leverage different variants of recurrent neural networks (RNNs). However, these RNN-based methods often perform suboptimally in challenging scenarios, such as when working with high missing ratios or imputing long sequences of adjacent missing entries [22]–[24]. Transformer-based approaches [24], [25], as well as those that use generative models such as generative adversarial networks (GANs) [26]–[28] and variational autoencoders (VAEs) [29]–[31], have shown potential in addressing these challenges.

Across many different domains, diffusion models have replaced previous standard models as the new state of the art [15], [16], [32]. Recently, this trend has been extended to time series imputation and forecasting, as diffusion-based imputation models have demonstrated substantial improvements over previous architectures, particularly in handling complex and high-dimensional data imputation tasks [15], [16]. In this paper, we examine the most successful diffusion-based generative models, with a detailed discussion of their design and implementation provided in Section III-C.

[1]We use the terms diffusion-based generative models and diffusion models interchangeably.

C. *Reconstruction of execution traces*

Although this study is the first to examine the capabilities of diffusion-based generative models for trace event reconstruction, there exist studies that have explored the use of traditional predictive models to predict missing events in a trace. Sucholutsky et al. [33] propose an LSTM model to restore Automotive Controller Area Network (CAN) traces. The model predicts a missing event, and then uses its own output as input to continue predicting future events. Tobia and Narayan [34] present a similar RNN-based approach that uses the timing of events to improve the prediction accuracy. Martin and Dagenais [12] propose an approach that relies on finite state machines to determine what trace events are missing.

Even though these techniques are shown to be useful, they suffer from many limitations. First, they require a large amount of training data, which may not be possible to obtain, especially in situations where missing events occur in the beginning of a trace. Secondly, these methods completely disregard one of the main advantages of data reconstruction, which is the fact that we can use the events following (and not only preceding) the missing event segments. Finally, most of the techniques have been evaluated using overly simplistic scenarios. For instance, the work by Martin and Dagenais [12] examines a manually created trace consisting of only 63 events, focusing solely on cases where a single event is missing. Additionally, the LSTM-based approach proposed by Sucholutsky et al. [33] is evaluated exclusively with traces from a single execution scenario. In this paper, we address these limitations by proposing the use of diffusion generative models and apply them to various large execution traces. We also compare the diffusion models to an LSTM-based approach.

III. Study Setup

Our approach involves collecting trace event sequences, identifying segments with missing events, and using a diffusion-based imputation model to reconstruct the missing segments. Figure 2 provides a visual overview of this process. In this section, we present the study setup, starting with the study's goals and research questions. Then, we describe the datasets, the diffusion models selected, the experimental procedure, and finally, the evaluation metrics.

A. *Goal and Research Questions*

The goal of this study is to investigate whether diffusion models can be used to reconstruct the content of execution traces. To achieve this goal, we have formulated four research questions (RQs):

RQ1 *Are diffusion models effective at reconstructing trace events?* Existing diffusion models have demonstrated great success in recreating continuous time-series data, such as audio waveforms, ECGs, and electric usage [22]. However, their applicability in reconstructing sequences of discrete events, particularly trace event sequences, remains unexplored.

For the next questions, we used the best performing model.

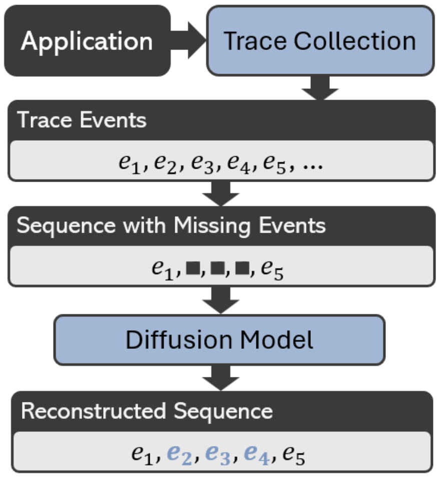


Fig. 2. Overview of our proposed method for trace reconstruction using diffusion models.

TABLE I
The PTS benchmarks used in this study, as described by [35]

Benchmark	Version	PTS Family	Description
compress-gzip	1.1.0	Processor	File compression
ffmpeg	2.5.0	Processor	Audio and video encoding
scimark2	1.2.0	Processor	Scientific computations
stream	1.3.0	Memory	Memory I/O performance
ramspeed	1.4.0	Memory	Memory I/O performance
phpbench	1.1.0	System	PHP interpreter performance
pybench	1.0.0	System	Python performance
iozone	1.8.0	Disk	Disk I/O performance
unpack-linux	1.0.0	Disk	Unpack the Linux kernel

RQ2 *How stable is the model's performance with varying imputation scenarios?* The length of missing sequences and the amount of intact data surrounding these missing blocks are not guaranteed. Therefore, a successful model must maintain high performance across a variety of scenarios.

RQ3 *How does a diffusion model compare to a traditional prediction-based approach?* This research question seeks to explore how diffusion models compare to traditional predictive-based approaches such as LSTM.

RQ4 *How well does a diffusion model predict different kinds of events?* Execution trace events can be grouped based on the kind of operations they represent. For example, an application will trigger a different sequence of events when it is performing file system operations than when it is performing network operations. This research question explores whether or not the effectiveness of diffusion models is impacted by the types of events its tasked with reconstructing, thereby assessing their generalizability and robustness across different event categories.

B. *Datasets*

Depending on the tracing objectives, various types of events can be collected, spanning from user space events generated by applications to kernel events that reveal the core activities of the operating system. For this study, we focus on system

TABLE II
THE SIZE AND COMPLEXITY OF EVENT SEQUENCES FROM DIFFERENT BENCHMARKS

Benchmark	Average # of Events per Run	# of Distinct Events
compress-gzip	4,619,719	108
ffmpeg	387,884	87
iozone	1,747,003	92
phpbench	9,763,841	96
pybench	129,618	91
ramspeed	249,348	104
scimark2	128,231	94
stream	133,628	99
unpack-linux	1,080,446	98

call sequences. System calls alone provide a detailed view of system execution, making them invaluable for various applications, including intrusion detection systems [36] and performance analysis [37].

To assess the models' capability in managing a diverse range of traces, we extracted the system call sequences from publicly available LTTng traces [35]. These traces were gathered from a desktop machine equipped with an Intel Xeon E3-1225 processor clocked at 3.20GHz, 32GB of RAM, a Gigabit Ethernet connection, and SSD storage. The system operated on Debian with the 4.4.0-1-amd64 kernel, and the traces were collected using version 2.8 of LTTng. All kernel events were enabled during tracing, however, we proceeded to filter out any events that were not system calls.

To evaluate the generalizability of the diffusion models, we constructed nine distinct datasets, all of which have been made publicly available[2]. Each dataset is derived from traces monitoring 32 runs of different benchmark applications from the Phoronix Test Suite (PTS) version 6.2.2[3]. An overview of the specific benchmarks analyzed in this study is presented in Table I. These benchmarks triggered a varying number of events, indicating different levels of system activity. The smallest traces were collected from the scimark2 benchmark, whereas the largest traces were collected from the phpbench benchmark. An overview of each benchmark's average number of system calls per run, as well as the number of distinct events in the traces, is given in Table II. The inclusion of varying benchmarks in our analysis allows us to assess the models' robustness across different execution scenarios, resource requirements, and general execution behaviours.

C. Diffusion model algorithms

We use four diffusion-based time series imputation models, DiffWave [38], SSSD^{S4} [22], SSSD^{SA} [22], and a variant of CSDI [39]. These models were chosen as they represent both the pioneering and most recent advancements in diffusion models that are specifically designed for time-series data imputation. Furthermore, they have demonstrated superior accuracy compared to conventional architectures in challenging imputation scenarios [22]. During training, these approaches add random noise conditioned on the intact data, and then learn to iteratively refine the noise using a reverse process p_θ into plausible time series (see Figure 3). This reverse process is then used to fill in damaged portions of the data. Despite sharing this common approach, the models have subtle architectural differences, which we discuss in what follows.

1) DiffWave: The DiffWave model [38] is a versatile diffusion probabilistic model initially proposed for audio synthesis. DiffWave's success has significantly influenced the use of diffusion models for time-series modeling [15], so much so that each of the other models we evaluate can be thought of as DiffWave variants. In our study, we use an adaptation of DiffWave's original architecture, which has demonstrated promising results in time-series reconstruction [22].

2) SSSD^{S4}: The SSSD^{S4} model [22] is one of the few diffusion-based models that was specifically designed for time series imputation. Unlike models that rely on dilated convolutions or transformers, SSSD^{S4} leverages structured state space (S4) models [40]. The S4 architecture stacks State Space Models (SSM), based on a linear state space transition equations given in Equation 4, where $u(t)$ is the input sequence, $y(t)$ is the output sequence, $x(t)$ is a hidden state, and A, B, C, and D are transition matrices.

$$x'(t) = Ax(t) + Bu(t) \text{ and } y(t) = Cx(t) + Du(t) \tag{4}$$

The S4 architecture excels in modeling long-term dependencies, making it a powerful and efficient tool for sequence-related tasks. Combining S4 models with the generative capabilities of diffusion models, the SSSD^{S4} architecture has been shown to outperform other state-of-the-art approaches, especially in diverse and challenging imputation scenarios.

3) SSSD^{SA}: ShaShiMi [41] is a generative sequence model that structures S4 layers in a U-Net-inspired configuration. While it was proposed as a standalone waveform modeling architecture, its authors also demonstrated how combining it with a diffusion model leads to improved generation performance. SSSD^{SA} is a DiffWave variant proposed alongside SSSD^{S4}, using a SaShiMi model instead of S4 components.

4) CSDI^{S4}: The Conditional Score-based Diffusion model for Imputation (CSDI) [39] is another architecture inspired by DiffWave, and the pioneering work on diffusion-based time eries imputation [42]. A variant of CSDI, known as CSDI^{S4} [22], replaces the transformer used to encode the conditional input with an S4 model. This substitution has been shown to significantly improve performance in reconstructing sequences of missing data points [22].

D. Procedure

1) Data Preprocessing: Given that the models examined in this study were originally designed for imputing continuous time series values, slight modifications are required to adapt the system call sequences. Specifically, for each dataset we extract the distinct system calls, sort them based on their frequency in the training dataset, and then assign each event a unique integer identifier based on this ordering. This approach allows the model to more effectively focus on the

[2]https://github.com/janecekm/TraceReconstruction

[3]https://www.phoronix-test-suite.com/

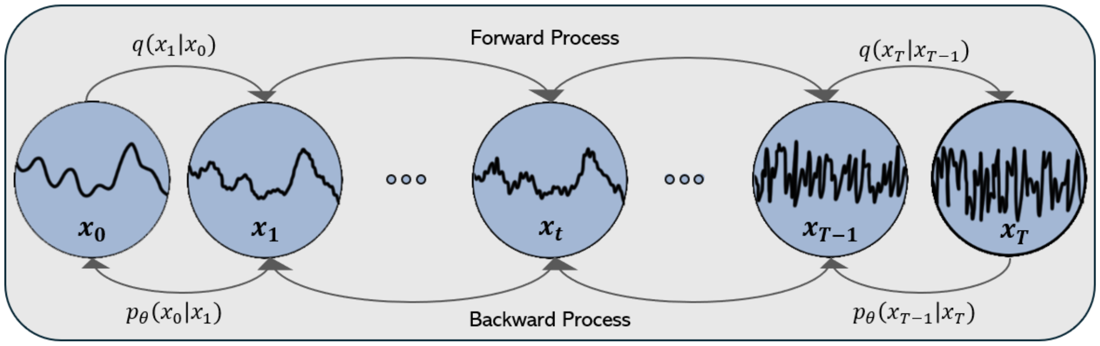


Fig. 3. A visualization of the general diffusion training process for the diffusion models

most common events during training while still capturing rare occurrences. The model's output is interpreted as the event corresponding to the integer nearest to its predicted value.

When working with the imputation models, examining the trace in its entirety is infeasible due to its size. Therefore, we provide the model with a smaller subsequence focused on the missing events, meaning we include some of the preceding events, the missing segment, and some subsequent events. For each benchmark, we split its corresponding dataset into 10,000 sequences of events that we use for training, and 500 sequences for testing. We experimented with various sequence lengths up to 200 events. This upper limit was determined based on computational constraints.

With each sequence, we artificially remove a continuous block of events for reconstruction. We define the number of removed events as the **blackout size**, which refers to the segment length of missing events in the sequence. In this study, we examine blackout sizes of 5, 10, 20, 30, and 40. In the worst case of a blackout size of 40, this means that longest sequences of 200 have a 20% missingness ratio and the shortest sequences of 50 have a 80% missingness ratio, allowing us to examine the models under extremely adverse conditions.

Unless otherwise specified, the missing segment is taken from the center of the sequence. This choice was not due to any limitation of the models, which are capable of imputing data anywhere within a sequence. Rather, it standardizes the evaluation process by enabling direct comparison of results across different models and scenarios. For example, reconstructing events at the end of a sequence is akin to time-series forecasting, which is a related yet distinctly different challenge [22]. Accordingly, such edge-case scenarios should not be equated with those where the model benefits from leveraging events occurring after the missing segment. The effects of reconstructing segments at the end of the sequence, rather than in the center, are exampled in Section IV-B3.

2) Training: For every combination of model and dataset, a new model was trained using a blackout size of 10. After training, the model could be used to reconstruct event sequences of varying sizes. In each instance, the model version with the lowest mean squared error (MSE) was selected. The MSE, given a set of n events $(x_1, \ldots, x_n)$ and their corresponding predictions $(y_1, \ldots, y_n)$, is calculated using Equation 5.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \tag{5}$$

For the DiffWave, SSSD^{S4}, and SSSD^{SA} models, initial experimentation lead us to choose best model after 10,000 iterations using a learning rate of 2e^{-4}. For the models using the CSDI^{S4} architecture, the model with the lowest MSE after 200 epochs using a learning rate of 1e^{-3} was chosen.

E. Evaluation metrics

To assess the efficacy of each model, we define the original sequence of trace events as the ground truth. The aim of this study was to reconstruct the exact sequence, with the events in the exact order. Consequently, if the order is not reconstructed properly, we do not consider the results to be a successful reconstruction. Furthermore, due to the categorical nature of the data, any output that deviates from the ground truth, regardless of its numerical proximity, is considered incorrect. Based on these factors, we chose to use the following performance metrics.

The **accuracy** of the model is defined as the proportion of events that are correctly positioned in the reconstructed sequence. Specifically, for a segment consisting of n events $s = (x_1, ..., x_n)$, the accuracy of the reconstructed sequence $r = (y_1, ..., y_n)$ is calculated using Equation 6.

$$a(s, r) = \frac{\sum_{i=1}^{n} \mathbb{1}_{x_i = y_i}}{n} \tag{6}$$

The overall accuracy of the model, given a set of original sequences $S = (s_1, ..., s_m)$ and their reconstructions $R = (r_1, ..., r_m)$, is then computed using Equation 7.

$$acc(S, R) = 100 * \frac{\sum_{j=1}^{m} a(s_j, r_j)}{m} \tag{7}$$

We quantify the model's ability to perfectly reconstruct entire sequences by calculating its **perfect rate**, defined as Equation 8.

$$p(S,R) = 100 * \frac{\sum_{j=1}^{m} \mathbb{1}_{a(s_j,r_j)=1}}{m} \tag{8}$$

It is important to note that the previous metrics can unduly penalize minor positional errors that do not significantly impact the overall sequence quality. Therefore, we also use the **ROUGE-L** metric [43], which is calculated using the the Longest Common Subsequence (LCS) [44] between the original and reconstructed sequences, as shown in Equation 9.

$$rouge\_l(s,r) = 100 * \frac{LCS(s,r)}{n} \tag{9}$$

We then define the model's overall ROUGE-L score as the average score across each test sequence and corresponding reconstruction. This provides a more nuanced evaluation of the model's performance, acknowledging that slight deviations in event positions may still result in a sequence that is functionally accurate.

IV. Results

In this section, we present the findings of our experiments evaluating the models under various conditions.

A. Are diffusion models effective at reconstructing trace events? (RQ1)

To address this question, we initially evaluated the models' accuracy when a single event was removed, which serves as a baseline for comparison before moving to more challenging imputation scenarios. As shown in Table III, the average accuracy ranges from 49.18% to 88.04%, with the SSSD^{S4} and SSSD^{SA} models consistently outperforming the others.

Notably, the CSDI^{SA} model exhibited significantly lower accuracy, averaging at 49.18%, indicating a less effective architectural design for trace event sequence reconstruction. This suboptimal performance may be partly due to the model's training objective, which focuses solely on denoising the missing segment. The SSSD models, which target the entire time series in their diffusion module, appear to more effectively capture the long-term dependencies within the data, leading to higher imputation accuracy.

We then tested the models' ability to reconstruct a sequence of 10 consecutive events. The performance seen with even larger blackout sizes is discussed in Section IV-B1. The accuracy, perfect rate, and ROUGE-L score that were obtained across the different datasets are shown in Table IV. Similarly to the single imputation, the models' performance remained stable across each dataset. Similar to the previous results, we found that CSDI^{S4} performs the worst among all the algorithms when applied to a blackout size of 10 events, with an accuracy, perfect rate, and ROUGE-L of 52.24%, 1.36% and 60.26%. This further supports the conclusions drawn from the single imputation results.

The SSSD^{SA} and SSSD^{S4} achieve the best accuracy, perfect rate, and ROUGE-L score, followed by DiffWave. Notably, the difference between the DiffWave model's accuracy and that of the SSSD variants grew with the imputation of multiple events. While the DiffWave model's perfect rate was notably low, averaging only 52.78%, its ROUGE-L score was comparable to that of the SSSD models, averaging 87.88%. These results highlight the critical contributions of the S4 components in the SSSD^{S4} and SSSD^{SA} architectures. The ROUGE-L score suggests that DiffWave can select events similar to the original sequence. However, lacking the S4 components found in the SSSD variants, the DiffWave model is less capable of consistently choosing the most probable events in the context of surrounding events, leading to at least one error in roughly half of the missing segments.

Given the better average accuracy seen with SSSD^{S4} models and over SSSD^{SA} models when imputing a single event, and comparable performance when working with a blackout size of 10, all subsequent experiments were conducted exclusively with the SSSD^{S4} model.

Finding RQ1: We found that the SSSD^{S4} model is particularly well-suited for trace data reconstruction, with an average 81.62% accuracy, 74.27% perfect rate, and 90.84% ROUGE-L score when reconstructing 10 events.

B. How stable is the model's performance with varying imputation scenarios? (RQ2)

When working with missing data, there is no guarantee when it comes to how much contextualizing data remains intact. To address this, we evaluated the robustness of the SSSD^{S4} model across various reconstruction scenarios. Specifically, we analyzed three primary alterations in the data: changes in the blackout size, changes in the sequence length, and changes in the missing segment's location.

1) Blackout Size: Table V presents the performance impact when changing the blackout size when working with sequences of 200 events. Following the procedure described in Section III-D1, we first removed 5 events from the sequences, then increased it to our previous baseline of 10. Further increments of 10 were then examined, culminating at a maximum blackout size of 40 events. This progression allows u to assess the model's performance up to a 20% missingness ratio, a condition that is indicative of substantial data loss.

We observed that as the blackout size increases, there is a corresponding decrease in model accuracy. However, even under the most challenging conditions, the model can still perfectly position missing events 72.66-79.44% of the time. The impact on performance is notably less pronounced when taking the increased difficulty into account. For instance, transitioning from a blackout size of ten to forty, a 400% increase in missing data, the model's average accuracy only decreases by 4.51%, demonstrating the resilience of the model to more challenging scenarios.

Additionally, the average ROUGE-L score of SSSD^{S4} models for each dataset when looking at different blackout sizes is shown in Table VI. We see that the models maintain their

TABLE III
ACCURACY WHEN IMPUTING A SINGLE EVENT

Model	Dataset									Average
	compress-gzip	ffmpeg	iozone	phpbench	pybench	ramspeed	scimark2	stream	unpack-linux	
DiffWave	86.0	85.4	86.2	79.4	82.8	83.8	85.8	84.4	81.8	83.96
SSSD^{S4}	**89.4**	**88.4**	89.8	84.0	**86.0**	**89.8**	86.0	**88.4**	90.6	**88.04**
SSSD^{SA}	89.2	87.4	**90.8**	**86.4**	84.4	**89.8**	**88.0**	83.3	**91.4**	87.86
CSDI^{S4}	32.0	46.8	71.0	63.0	37.6	41.6	33.8	64.2	52.6	49.18

TABLE IV
PERFORMANCE WITH A BLACKOUT SIZE OF 10

Model		Dataset									Average
		compress-gzip	ffmpeg	iozone	phpbench	pybench	ramspeed	scimark2	stream	unpack-linux	
DiffWave	Accuracy	77.06	76.70	77.42	71.40	70.92	76.90	77.94	75.56	75.74	75.52
	Perfect Rate	54.6	65.0	56.8	46.6	47.0	52.2	63.6	43.6	45.6	52.78
	ROUGE-L	88.67	88.50	88.81	85.85	85.61	88.59	89.09	87.85	87.99	87.88
SSSD^{S4}	Accuracy	**83.06**	81.46	**83.46**	77.44	**78.14**	82.58	82.44	82.80	83.16	81.62
	Perfect Rate	**76.2**	**74.0**	**76.8**	68.4	69.0	76.4	**75.2**	76.0	**76.4**	74.27
	ROUGE-L	**91.59**	90.74	**91.76**	88.78	**89.07**	91.35	91.22	91.44	91.58	90.84
SSSD^{SA}	Accuracy	82.72	**81.48**	83.18	**78.02**	78.02	**82.84**	**82.66**	**83.3**	**83.2**	**81.71**
	Perfect Rate	76.4	73.4	76.4	**69.4**	**69.4**	**76.6**	75.0	**76.4**	**76.4**	**74.38**
	ROUGE-L	91.43	**90.82**	91.61	**89.14**	89.02	**91.48**	**91.35**	**91.66**	**91.61**	**90.90**
CSDI^{S4}	Accuracy	53.84	55.08	59.92	63.54	32.42	44.72	47.06	62.14	51.44	52.24
	Perfect Rate	0.6	0.8	1.6	3.6	0.4	0.8	0.6	3.2	0.6	1.36
	ROUGE-L	56.88	60.18	62.82	68.62	52.14	57.78	57.40	66.92	59.58	60.26

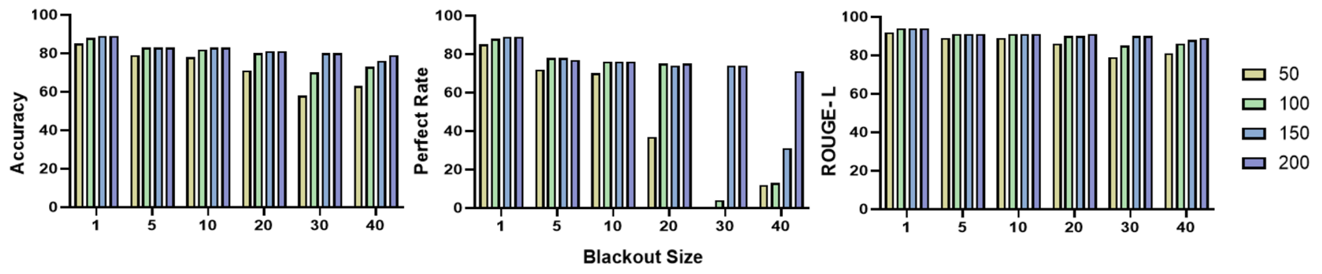


Fig. 4. The accuracy, perfect rate, and ROUGE-L score of SSSD^{S4} models working with sequence lengths of 50, 100, 150, and 200. All results presented here were obtained using the compress-gzip dataset.

TABLE V
ACCURACY OF SSSD^{S4} MODELS USING DIFFERENT BLACKOUT SIZES

Dataset	Blackout Size				
	5	10	20	30	40
Average	82.70	81.62	80.19	78.82	77.11
compress-gzip	83.88	83.06	81.98	80.573	79.39
ffmpeg	82.92	81.46	79.91	78.12	76.01
iozone	84.44	83.46	81.91	80.56	78.91
phpbench	78.72	77.44	76.44	74.48	72.66
pybench	79.88	78.14	76.50	75.04	73.63
ramspeed	83.76	82.58	81.27	80.11	78.26
scimark2	83.00	82.44	80.54	79.35	77.24
stream	83.80	82.80	81.39	80.55	79.44
unpack-linux	83.88	83.16	81.77	80.57	78.46

TABLE VI
AVERAGE ROUGE-L SCORE OF SSSD^{S4} MODELS

Dataset	Blackout Size				
	5	10	20	30	40
Average	91.35	90.84	90.29	89.52	88.70
compress-gzip	91.94	91.59	91.11	90.34	89.80
ffmpeg	91.46	90.74	90.09	89.24	88.22
iozone	92.22	91.76	91.10	90.44	89.58
phpbench	89.36	88.78	88.43	87.33	86.51
pybench	89.94	89.07	88.41	87.61	86.95
ramspeed	91.88	91.35	90.91	90.16	89.32
scimark2	91.50	91.22	90.55	89.83	88.79
stream	91.90	91.44	90.92	90.37	89.83
unpack-linux	91.94	91.58	91.05	90.35	89.33

ability to produce sequence that are similar, although not exact matches of the original segment, across varying blackout sizes. Specifically, the difference in average ROUGE-L score moving from a blackout size of ten to forty is only 2.14, indicating the models' capacity to maintain sequence coherence across varying degrees of data loss.

2) Sequence Length: We also examined the effect of reducing the number of events in the overall sequence. The accuracy, perfect rate, and ROUGE-L score using different blackout sizes and sequences of length 50, 100, 150, and 200 is shown in Figure 4. For sequences of lengths 100 and 150, there was no significant performance impact until the blackout size exceeded 20% of the overall sequence. In the most challenging

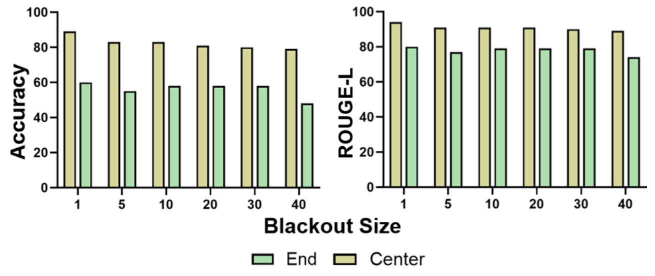


Fig. 5. The accuracy (left) and ROUGE-L score (right) of SSSD^{S4} when reconstructing segments in the center of the sequence compared to reconstructing segments at the end of the sequence. All results presented here were obtained using the compress-gzip dataset.

configuration, sequences of length 50, we observed a notable decrease in accuracy and the perfect rate. Nonetheless, even with a blackout size of 40 and sequence length of 50, meaning the model had only 5 preceding and 5 following events, it achieved an impressive 63.24% accuracy and an 81.8 ROUGE-L score. This indicates that while the model's performance is affected by data availability, it still produces reasonable reconstructions under unfavorable conditions.

Generally, the ROUGE-L score remained stable across variations, ranging from 79.34% in the worst case to 94.8% at its best. This suggests that decreasing the context and increasing the reconstruction length impacts the model's ability to exactly reconstruct the trace, but it consistently retains the ability to reconstruct a plausible trace sequence.

3) Missing Segment Positioning: One of the key advantages of data reconstruction over prediction is the ability to leverage the intact data that follows a missing segment. To assess the significance of this additional information, position the missing segment at the end of the sequence, thereby removing any subsequent data that the model could use for context. The results, which are visualized in Figure 5, indicate a significant decline in accuracy when reconstructing without the benefit of following events. This finding underscores the the importance of contextual data, and highlights the value of models specifically designed for data imputation, which can fully leverage this context.

Despite this decrease in performance, the SSSD^{S4} model still demonstrates a remarkable ability to reconstruct events at the end of the sequence. When working with a blackout size of 10, it achieved a 58.0% accuracy and a 79.05% ROUGE-L score, even in the absence of following data.

Finding RQ2: The performance of the SSSD^{S4} model remains stable across different sequence lengths and blackout sizes, provided the missingness ratio is below 20%. Adversely challenging scenarios lead to decreased performance, but the model still achieves acceptable results, particularly in terms of ROUGE-L score.

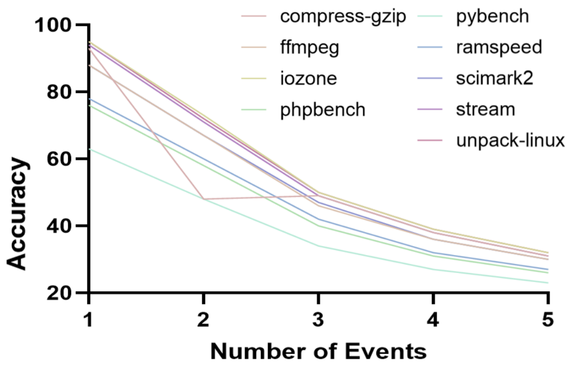


Fig. 6. The accuracy of LSTM models when tasked with repeatedly predicting the next event in the missing segment.

C. *How does a diffusion model compare to a traditional prediction-based approach? (RQ3)*

To compare the performance of the SSSD^{S4} model against more established prediction-based methods, we trained LSTM models to perform next event prediction for each of the nine datasets. To make these results comparable to the results obtained by having the diffusion models impute a single event in a sequence of 200, the LSTM models were given the preceding 199 events to make a prediction. Upon achieving sufficiently high accuracy in this task, we evaluated the models' ability to reconstruct entire segments of consecutive events. This was accomplished by iteratively feeding a model's prediction back into itself to predict subsequent events until the desired segment length was reached. The results of this testing when reconstructing up to five events is visualized in Figure 6.

When just predicting a single event, an LSTM model appear like a competitive alternative to a SSSD^{S4} model, with an average 86.10% accuracy across datasets. However, the performance declines significantly when tasked with predicting subsequent events. When predicting just the second event, the LSTM models had a markedly lower accuracy range of 33.73-50.44%, bringing down the average accuracy of reconstructing a segment of two consecutive events to 65.73%.

For the reconstruction of sequences of five events, errors made early on in the segment are propagated and expanded upon, leading the LSTM models to have an average accuracy of 29.61%. This is a stark contrast to the SSSD^{S4} models, which attained an average accuracy of 88.04%. Even when focusing on reconstructing the last five events in a sequence, where the comparison is arguably more equitable, the SSSD^{S4} models nearly double the performance of the LSTM models, achieving an average accuracy of 55.69%.

Finding RQ3: Models that are not specifically designed for data imputation perform considerably worse than the SSSD^{S4} model, with the LSTM model achieving a 29.61% accuracy when reconstructing 5 events.

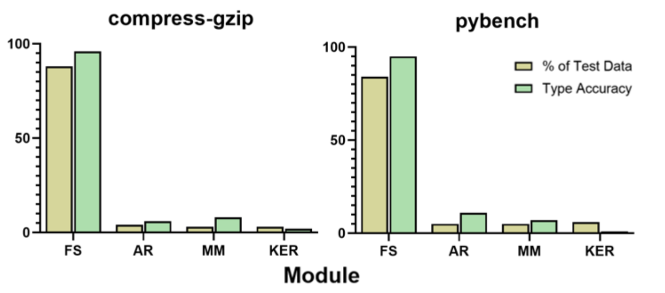


Fig. 7. The type accuracy of SSSD^{S4} models broken down by module and compared to the proportion of events in the test set that belong in that module. All results presented here were obtained using a sequence length of 200 and blackout size of 10.

D. How well does a diffusion model predict different kinds of events? (RQ4)

To evaluate how well a SSSD^{S4} model handles different types of trace events, we grouped the system calls by their corresponding modules in the Linux source code. We observed that across the different datasets, the most common events were associated with the file system (FS), architecture (AR), memory management (MM), and kernel (KER) modules. To assess the a SSSD^{S4} model's ability to correctly impute events of the correct module, we defined the **type accuracy** metric as the proportion of times the model positioned an event of the correct type in the reconstruction sequence.

Using a sequence length of 200 and a blackout size of 10, the SSSD^{S4} models' average type accuracy across all nine datasets was 84.74%, a slight improvement over its average accuracy of 81.62%.

We further analyzed these results by breaking down a model's type accuracy by module and comparing it to the proportion of test events in that module. The results for the compress-gzip and pybench datasets are illustrated in Figure 7. The results indicate that SSSD^{S4} models do not necessarily struggle with a specific type of event, but rather the performance is impacted by the rarity of a given event. Events within the AR, MM, and KER modules are far less frequent than those in the FS module, and consequently a model is less likely to accurately position an event of these types.

Finding RQ4: We found that the type of event does not have an impact on performance, however the rarity of an event does.

E. Discussion

The key findings from our research, along with potential avenues for future investigation are as follows:

1) Diffusion Model Architectures: There are some key properties of trace event sequences that make them particularly well-suited for the SSSD^{S4} model. Firstly, trace events have a limited alphabet (in our case, the alphabet consists of the Linux system calls), unlike natural language processing tasks that deal with extensive vocabularies or datasets comprised of a continuous range of numbers, as discussed in the SSSD^{S4} model's foundational paper [22]. In this study, the datasets have a much more manageable and distinct set of elements.

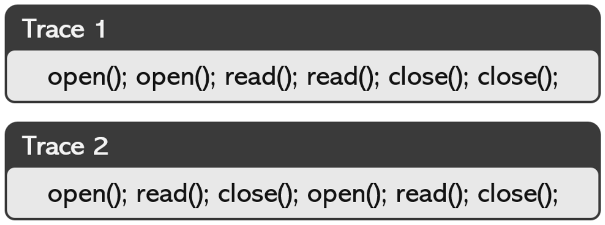
Secondly, there is a strong causality between trace events. For instance, the existence of a `open` system call necessitates a corresponding `close` system call later on in the sequence. As such, there are strong patterns that the model, particularly the S4 layers, can leverage when reconstructing missing segments. The importance of the S4 layers is particularly evident when looking at the DiffWave model, which while effective, lacks the specialized handling of long-term dependencies that the S4 layers provides in the SSSD^{S4} and SSSD^{SA} architectures. The superior performance of these models suggests that this architectural enhancement is crucial for effectively processing patterns in execution trace event sequences.

Lastly, the loss of events in traces often occurs in segments rather than as isolated, random events. The original authors of the CSDI^{S4} and SSSD^{S4} models observed that while CSDI^{S4} is adept at handling scenarios where items are randomly missing, the SSSD^{S4} excels in reconstructing contiguous blocks of events. This capability is crucial for modeling dependencies and accurately restoring the integrity of the sequence.

2) Model Enhancement: While several of the characteristics of trace sequences make them well-suited for an SSSD^{S4} model, certain attributes render accurate reconstruction a challenging task. A notable difficulty arises from trace events that are nearly identical except for minor differences, such as system calls like `chmod` and `fchmod`, `fork` and `vfork`, or `stat`, `fstat`, and `lstat`. A robust model must be capable of discerning these nuanced differences and accurately determine which event was more likely to have taken place. Furthermore, trace event sequence often exhibit domain-specific patterns. For example, it is not uncommon to see several `read` system calls in a row. This typically occurs when a program reads data in chunks to manage memory efficiently, particularly when working with large files. One common error we observed in the SSSD^{S4} model is the placement of a `read` system call where a `close` system call should be, indicating difficulties in correctly identifying the frequency and sequence of file access events.

To improve the model's generalizability and its ability to recognize these nuanced scenarios, we propose two primary approaches. Firstly, the ordering of events is a small part of the information captured in trace data. Integrating additional contextualizing data, such as the event timings, arguments, and durations, could enhance the model's capacity to accurately reconstruct the missing data. Secondly, incorporating methods that augment the model's semantic understanding of the trace, potentially through the use of knowledge bases or expert-defined patterns, should be investigated.

3) Model Stability: As it stands, a model's ability to position reconstructed events so that they exactly line up exactly with the original sequence is significantly influenced to the availability of intact data. As the sequence length decreases and the blackout size increases, the overall performance of the model also declines. In particular, we observed that the


Trace 1
Trace 2

```
open(); open(); read(); read(); close(); close();
```

```
open(); read(); close(); open(); read(); close();
```

Fig. 8. Two functionally equivalent but syntactically different sequences of system calls.

accuracy and perfect rate of a model can sharply decrease when the blackout rate is greater 20% of the sequence length. This indicates that shorter traces are particularly vulnerable to the adverse effects of larger blackout sizes.

Addressing this challenge requires the development of imputation models that are resistant to varying levels of data integrity. Future research should explore architectural innovations that mitigate the effects of high data loss. For instance, while the S4 model is a critical component in many of the studied architectures due to its efficiency in handling long sequences, other SSM-based architectures have demonstrated superior performance in certain sequence learning benchmarks [45]. Investigating the potential benefits of replacing S4 layers with these alternative architectures could enhance the accuracy of trace event sequence reconstruction. Furthermore, enhancing the size and diversity of the training data, through methods such as increasing sample counts, incorporating a variety of trace types, or employing artificial augmentation techniques, could further bolster model robustness.

4) Partial Reconstruction: In this study, the overarching aim was to assess the capability of diffusion models in generating sequences that exactly match the original data. To this end, metrics like accuracy, perfect rate, and ROUGE-L are well suited for quantifying the models' fidelity. However, there are scenarios where generating sequences that are not exact replicas, but are functionally similar to the original segments is more desirable. This is particularly relevant in applications like dataset augmentation, where the model would generate plausible training sequences that were not initially captured during tracing, thus enabling the training of more robust classification or prediction models. Another example is program optimization, where the model would suggest functionally equivalent but optimized sequences of events.

For the development of models that partially reconstruct sequences, the metrics we use would be inadequate. Consider the example shown in Figure 8. Both system call sequences involve opening, reading, and closing two files. The second sequence, however, minimizes the number of files that are open simultaneously. If a model tasked with reconstructing the first sequence were to produce the second, the functional equivalency would be overlooked, and the model would be penalized with a low ROUGE-L score of 66.67% and an even lower accuracy of 33.33%. We believe that development of partial reconstruction models would therefore require alternative evaluation methods, such as expert assessment or automated evaluation through large language models (LLMs).

V. Threats to Validity

A. Internal Threats

The performance of diffusion models can be significantly influenced by their parameter settings. To mitigate this, we conducting initial testing informed by the findings of the models' publishers [22]. However, this is still always a possibility that different settings would produce better results.

Additionally, events were removed from complete sequences in a controlled manner to create artificial test scenarios. While this approach may not fully replicate the complexity and variability of naturally lost events, it was necessary to ensure comparability between the models' output and the original sequences. Testing the model's ability to reconstruct authentically lost events would require methods that assess the plausibility of events, like those mentioned in Section IV-E4.

B. External Validity

To support generalizability with different traces, we used trace data from nine different benchmarking applications. Although future research should aim to fully generalize our results, we believe that this threat has been minimized by the number of datasets we used.

C. Construction Threats

The accuracy, perfect rate, and ROUGE-L score how close syntatically the reconstructed sequence is to the original sequence, however, they do not evaluate functional correctness or plausibility. In practice, functionally equivalent but syntactically different sequences might be acceptable or even preferable. Once again, this calls for the development of the evaluation metrics mentioned in Section IV-E4.

VI. Replication Package

All datasets and results are available on GitHub: https://github.com/janecekm/TraceReconstruction

VII. Conclusion

In this study, we investigated methods for reconstructing execution trace events, focusing on the application of diffusion models. While previous work has shown the effectiveness of these models on time-series data, our study extends these findings to sequential trace data. Through comprehensive experiments involving multiple model architectures and datasets, we aimed to address several key research questions.

Our findings demonstrate that diffusion-based models, particularly the SSSD^{S4} model, are highly effective at reconstructing missing events in execution traces. These models performed exceptionally well in terms of accuracy, perfect rate, and ROUGE-L scores across various imputation scenarios. We confirmed that these models maintain stability under different conditions, such as varying lengths of missing sequences and different types of events, reflecting their robustness and adaptability. Furthermore, when comparing diffusion-based models with traditional prediction-based approaches like LSTM, we found that the former offered superior accuracy and robustness.

This study demonstrates the potential of diffusion-based models for reconstructing sequence data across various domains, thereby preserving data integrity and enhancing analytical capabilities. Overall, this study provides critical insights into the factors influencing the success and limitations of these models. We identified key elements that impact performance and offered directions for future research. These findings underscore the broader applicability of diffusion-based generative models in reconstructing sequential data, benefiting various fields that rely on accurate sequence data reconstruction.

Future research should focus on further refining the SSSD^{S4} model's architecture to enhance its accuracy and robustness. Exploring the integration of additional contextual data, such as event timings and arguments, could provide a more comprehensive understanding of trace events, leading to even more precise reconstructions. Other models beyond those explored in this study, such as the use of transformer-based imputation models or language models, should also be explored. Additionally, further research should investigate the efficacy of models trained on varying kinds of traces to ensure they are not application-specific.

References

[1] R. Alanazi, G. Gharibi, and Y. Lee, "Facilitating program comprehension with call graph multilevel hierarchical abstractions," *Journal of Systems and Software*, vol. 176, p. 110945, 2021.

[2] M. Noferesti and N. Ezzati-Jivan, "Enhancing empirical software performance engineering research with kernel-level events: A comprehensive system tracing approach," in *Journal of Systems and Software*, vol. 216: 112117, 2024.

[3] H. Chen, K. Wei, A. Li, T. Wang, and W. Zhang, "Trace-based intelligent fault diagnosis for microservices with deep learning," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 884–893.

[4] M. Janeck, N. Ezzati-Jivan, and A. Hamou-Lhadj, "Performance anomaly detection through sequence alignment of system-level traces," in *Proc. of the 30th IEEE/ACM International Conference on Program Comprehension (ICPC)*, 2022, pp. 264–274.

[5] M. Panahandeh, A. Hamou-Lhadj, M. Hamdaqa, and J. Miller, "Serviceanomaly: An anomaly detection approach in microservices using distributed traces and profiling metrics," in *Journal of Systems and Software*, vol. 209: 111917, 2024.

[6] "perf: Linux profiling with performance counters," accessed: date-of-access. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page

[7] M. Desnoyers and M. Dagenais, "LTTng: Tracing across execution layers, from the hypervisor to user-space," in *Linux Symposium*, 2008, p. 101.

[8] "ftrace - function tracer." [Online]. Available: https://www.kernel.org/doc/html/v5.0/trace/ftrace.html

[9] "Event tracing for windows." [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/etw/event-tracing-portal

[10] F. Giraldeau, "Recovering system metrics from kernel trace," 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:209505649

[11] "Avoiding lost buffers." [Online]. Available: https://learn.microsoft.com/en-us/previous-versions/windows/desktop/xperf/avoiding-lost-buffers?redirectedfrom=MSDN

[12] M. Martin and M. Dagenais, "Analyse detaillee de trace en depit d'evenements manquants," Master's thesis, École Polytechnique de Montréal.

[13] "The LTTng documentation." [Online]. Available: https://lttng.org/docs/v2.13/

[14] M. S. Osman, A. M. Abu-Mahfouz, and P. R. Page, "A survey on data imputation techniques: Water distribution system as a use case," *IEEE Access*, vol. 6, pp. 63 279–63 291, 2018.

[15] C. Meijer and L. Y. Chen, "The rise of diffusion models in time-series forecasting," 2024.

[16] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Comput. Surv.*, vol. 56, no. 4, nov 2023. [Online]. Available: https://doi.org/10.1145/3626235

[17] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. [Online]. Available: https://arxiv.org/abs/2204.06125

[18] C. Saharia, W. Chan, S. Saxena, L. Lit, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. Gontijo-Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22. Red Hook, NY, USA: Curran Associates Inc., 2024.

[19] W. Cao, D. Wang, J. Li, H. Zhou, L. Li, and Y. Li, "BRITS: Bidirectional recurrent imputation for time series," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[20] Z. Che, S. Purushotham, K. Cho, D. A. Sontag, and Y. Liu, "Recurrent neural networks for multivariate time series with missing values," *Scientific Reports*, vol. 8, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:4900015

[21] J. Yoon, W. R. Zame, and M. van der Schaar, "Estimating missing data in temporal data streams using multi-directional recurrent neural networks," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 5, pp. 1477–1490, 2018.

[22] J. L. Alcaraz and N. Strodthoff, "Diffusion-based time series imputation and forecasting with structured state space models," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=hHiIbk7ApW

[23] I. M. Andrea Cini and C. Alippi, "Filling the g_ap_s: Multivariate time series imputation by graph neural networks," *International Conference on Learning Representations*, 2022.

[24] W. Du, D. Côté, and Y. Liu, "SAITS: Self-attention-based imputation for time series," *Expert Systems with Applications*, vol. 219, p. 119619, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423001203

[25] A. Lotfipoor, S. Patidar, and D. P. Jenkins, "Transformer network for data imputation in electricity demand data," *Energy and Buildings*, vol. 300, p. 113675, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778823009052

[26] Y. Luo, X. Cai, Y. ZHANG, J. Xu, and Y. xiaojie, "Multivariate time series imputation with generative adversarial networks," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[27] Y. Luo, Y. Zhang, X. Cai, and X. Yuan, "E2gan: end-to-end generative adversarial network for multivariate time series imputation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 3094–3100.

[28] X. Miao, Y. Wu, J. Wang, Y. Gao, X. Mao, and J. Yin, "Generative semi-supervised learning for multivariate time series imputation," in *AAAI Conference on Artificial Intelligence*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235349067

[29] V. Fortuin, D. Baranchuk, G. Raetsch, and S. Mandt, "Gp-vae: Deep probabilistic time series imputation," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 1651–1661. [Online]. Available: https://proceedings.mlr.press/v108/fortuin20a.html

[30] A. W. Mulyadi, E. Jun, and H.-I. Suk, "Uncertainty-aware variational-recurrent imputation network for clinical time series," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9684–9694, 2022.

[31] S. Kim, H. Kim, E. Yun, H. Lee, J. Lee, and J. Lee, "Probabilistic imputation for time-series classification with missing data," in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML'23. JMLR.org, 2023.

[32] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[33] I. Sucholutsky, A. Narayan, M. Schonlau, and S. Fischmeister, "Deep learning for system trace restoration," Budapest, Hungary, 2019.

[34] J. P. Tobia and A. Narayan, "Is timing critical to trace reconstruction?" in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2021, pp. 2700–2705.

[35] A. Martin and V. Marangozova-Martin, "Automatic benchmark profiling through advanced workflow-based trace analysis," *Software: Practice and Experience*, vol. 48, 02 2018.

[36] M. Liu, Z. Xue, X. Xu, C. Zhong, and J. Chen, "Host-based intrusion detection system with system calls: Review and future trends," *ACM Comput. Surv.*, vol. 51, no. 5, nov 2018. [Online]. Available: https://doi.org/10.1145/3214304

[37] I. Kohyarnejadfard, M. Shakeri, and D. Aloise, "System performance anomaly detection using tracing data analysis," in *Proceedings of the 2019 5th International Conference on Computer and Technology Applications*, ser. ICCTA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 169–173. [Online]. Available: https://doi.org/10.1145/3323933.3324085

[38] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," 09 2020.

[39] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "Csdi: conditional score-based diffusion models for probabilistic time series imputation," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2024.

[40] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *The International Conference on Learning Representations (ICLR)*, 2022.

[41] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," 02 2022.

[42] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, "Diffusion models for time-series applications: a survey," *Frontiers of Information Technology & Electronic Engineering*, vol. 25, no. 1, pp. 19–41, 2024.

[43] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[44] D. S. Hirschberg, "Algorithms for the longest common subsequence problem," *J. ACM*, vol. 24, no. 4, p. 664–675, oct 1977. [Online]. Available: https://doi.org/10.1145/322033.322044

[45] C. A. Alonso, J. Sieber, and M. N. Zeilinger, "State space models as foundation models: A control theoretic overview," *ArXiv*, vol. abs/2403.16899, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268681121