Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

SIMDIFF: MODELING AND GENERATION OF STOCHASTIC DISCRETE-EVENT SIMULATION INPUTS VIA DENOISING PROBABILISTIC DIFFUSION PROCESS

Fengwei Jia[1], Hongli Zhu[1], Fengyuan Jia[2], Xinyue Ren[1], Siqi Chen[1], Hongming Tan[1], and Wai Kin Victor Chan[1]

[1]Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, CHINA
[2]Anhui Province Key Laboratory of Special Heavy Load Robot, Maanshan, Anhui, CHINA

ABSTRACT

This paper introduces Simulation using Diffusion process (SimDiff), a novel framework for automated modeling and generation of stochastic discrete-event simulation (DES) input distributions, addressing the high entry barrier and challenges associated with obtaining accurate input data. Traditional DES models often rely on simplifying assumptions, such as Poisson and Exponential distributions, which may not fully capture the complexity of real-world systems. We propose SimDiff to overcome these limitations by utilizing the denoising probabilistic diffusion model, a generative neural network capable of learning complex statistical distributions and efficiently sampling from them. Additionally, we introduce SimDiff-ConvTrans, an extension that incorporates Transformer and Convolution components for simulating non-i.i.d inputs. Our experiments demonstrate the effectiveness of SimDiff in handling simple and hybrid data distributions, as well as empirical datasets. SimDiff represents a significant advancement in simplifying the stochastic simulation process, making it more accessible and efficient for users across various expertise levels.

1 INTRODUCTION

Random Discrete Event Simulation (DES) is a well-established technique that plays a pivotal role in enhancing the design and operation of complex engineering systems under uncertain conditions. The essence of DES lies in its ability to model and analyze systems that are characterized by discrete events, such as customer arrivals in a queuing system or machine breakdowns in a manufacturing process. Despite its significance, the entry barrier to DES remains high, primarily due to the complexity involved in defining accurate input distributions based on empirical data from existing systems.

Modern simulation software tools (AnyLogic, Vensim, FlexSim, and Arena etc.) have made significant strides in simplifying the task of specifying simulation model structures through graphical user interfaces. However, for non-experts, the challenge of modeling simulation inputs remains formidable. The crux of the issue lies in the difficulty of obtaining data that fits the assumed input distributions, which often leads to reliance on assumptions within the model. For instance, it is a common practice to assume that the input process adheres to a Poisson distribution and the service process to an exponential distribution (Palmer and Tian 2023; Mielczarek and Zabawa 2016). While these assumptions facilitate the calibration of DES models to match the performance metrics of real-world systems, they are not infallible. The complexity of real-world processes often transcends the simplistic representations offered by Poisson or exponential distributions, necessitating the exploration of more sophisticated distribution models for a more accurate simulation of actual systems.

The historical development of DES has seen the emergence of software tools like ExpertFit, which excels in generating input data that is independently and identically distributed (i.i.d.) with simple characteristics. However, for i.i.d. data with complex features and non-i.i.d. data, the support is sparse. Nelson et al. (2021) demonstrated that frequentist model averaging effectively creates input models that more accurately reflect the true, yet unknown, input distribution. This approach reduces the risk associated with modeling

and enhances the fidelity of simulation outcomes. Devroye (2010) investigated the intricacies of generating non-uniform random variables and devised an automated method for generating code applicable to various distributions. However, a common limitation in these methods is the requirement to initially define the distribution functions, which in turn constrains adaptability in modeling complex systems.

The challenge of non-i.i.d. input data is further compounded by the high temporal correlation in DES data, prompting researchers to employ time series forecasting methods. Traditional approaches, such as moving averages, exponential smoothing, and ARMA/ARIMA models (Shumway et al. 2000), are prevalent due to their mathematical rigor, swift prediction speeds, and low computational demands. Yet, these methods are not without their shortcomings. Their inability to accommodate linear models, manage complex time series structures, and limited predictive accuracy are significant drawbacks, especially when dealing with intricate time series data. The extensive preparatory work required for domain-specific time series predictions is both resource-intensive and laborious.

In the absence of adequate guidance or software support, users are left to navigate a labyrinth of potential autoregressive models, which may be non-stationary, including time series models. For example, ARIMA can analyze the dynamic structure of climate influenced by changes in precipitation and temperature (Dimri et al. 2020) and can also analyze the temporal trends of disease outbreaks (Majidnia et al. 2023). GARCH can be used for weather forecasting and wind power prediction (Hong et al. 2023). Based on SETAR, a novel and accurate tree algorithm (SETAR-Tree) has been proposed for global time series forecasting (Ghasempour et al. 2023). Additionally, time series forecasting can also be done through recurrent topologies (Godahewa et al. 2023). However, even once a suitable stochastic process model is chosen, the task of efficiently creating sample paths remains a significant challenge, as highlighted in Cen et al. (2020) work on the NIM model.

To overcome these limitations, we introduce Simulation using Diffusion process (SimDiff), a novel framework for the automated modeling and generation of stochastic discrete-event simulation input distributions. SimDiff harnesses the power of denoising probabilistic diffusion models, a type of generative neural network, which not only learns complex statistical distributions but also facilitates sampling from these distributions. This approach is designed to circumvent overfitting issues and ensure that the generated sample values remain within the original training range.

Our contributions to the field of stochastic simulation modeling are multifaceted:

1. We introduce the innovative SimDiff framework, which addresses the challenges associated with the costly and labor-intensive collection of stochastic discrete-event simulation inputs.
2. We extend the SimDiff framework to SimDiff-ConvTrans, incorporating Transformer and Convolution components to effectively simulate complex or non-i.i.d. inputs.
3. Our empirical evaluations demonstrate the efficacy of our approach across a spectrum of scenarios, including simple input distributions, hybrid data distributions, and real-world datasets.

Our approach, SimDiff, not only addresses the challenges posed by stochastic simulation modeling but also opens up new possibilities for traditional time series models. By leveraging the denoising probabilistic diffusion model, SimDiff can learn intricate statistical distributions and generate sample paths with remarkable efficiency. This innovation eliminates the need for extensive preparatory work, significantly reducing the manpower and resources required for authentic DES input data. In addition, SimDiff's integrated Transformer technology makes it unique, which is particularly suitable for automated input modeling to automate random simulations. This not only streamlines the stochastic simulation modeling process but also makes it more inclusive and accessible to users across various skill levels, and thereby democratizing the application of stochastic simulation in diverse domains.

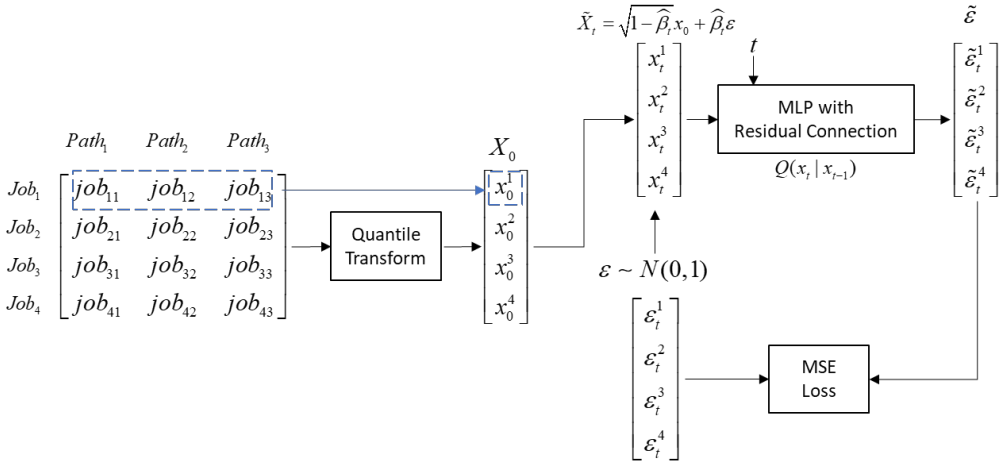


Figure 1: In the training phase, the SimDiff model was utilized to predict a denoising step, which was conditioned on a given timestep and noise level. The model was fed with an input consisting of three independent and identically distributed (i.i.d.) paths, each simulating a distinct job, totaling four jobs in the simulation.

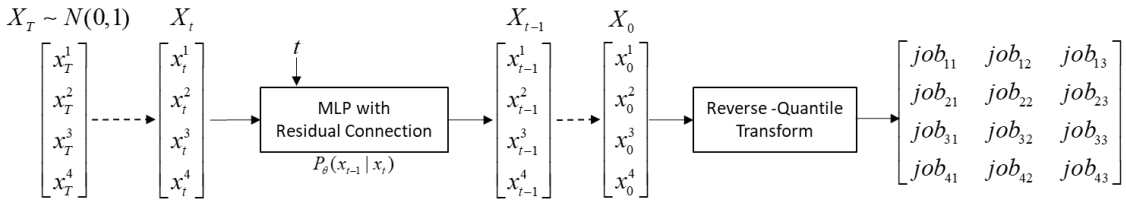


Figure 2: During the sampling phase, the SimDiff model denoise a sequence of vectors across multiple timesteps, effectively reconstructing the underlying signal and generating new, noise-free data points.

2 GAUSSIAN DIFFUSION AND SIMDIFF

2.1 Gaussian Diffusion Framework Overview

A Gaussian diffusion framework, also known as the Denoising Diffusion Probabilistic Model (Ho et al. 2020), is a specific type of generative neural network that simulates the process of data distribution diffusion and reconstruction to generate new data samples. The diffusion model assumes that a dataset is generated through the propagation of characteristic traits or a distributional pattern. The progress is governed by probabilistic rules that define how the states of nodes or individuals evolve over time.

In our initial prototype of the SimDiff model, we employ a direct implementation of the Gaussian diffusion process, as depicted in Figure 1 and 2.

The forward process begins at X_0 and sequentially generates latent variables $X_1, \ldots, X_T$ via a Markov Chain. This is accomplished by gradually incrementally introducing perturbations, transitioning towards pure Gaussian noise $X_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Each Markov transition is characterized by the equation:$q(X_t|X_{t-1}) = \mathcal{N}(X_t; \sqrt{1-\beta_t}X_{t-1}, \beta_t\mathbf{I})$, where β_t represents the noise level introduced at timestep t. The closed-form sampling of X_t from the initial state X_0 for an arbitrary timestep t is given by $q(X_t|X_0) = \mathcal{N}(X_t; \sqrt{1-\hat{\beta}_t}X_0, \hat{\beta}_t\mathbf{I})$, where $\hat{\beta}_t := 1 - \prod_{i=0}^{t}(1-\beta_i)$.

Conversely, the reverse process is designed to incrementally denoise the latent variables X_t to retrieve the original data X_0. To approximate this reverse process, we train a neural network with parameters θ. Each denoising step is parameterized as $p_\theta(X_{t-1}|X_t) = \mathcal{N}(X_{t-1}; \mu_\theta(X_t,t), \Sigma_\theta(X_t,t))$, where μ_θ and Σ_θ denote

the estimated mean and covariance of $q(X_t|X_{t-1})$, respectively. Given that Σ_θ is diagonal, the computation of μ_θ is detailed by $\mu_\theta(X_t,t) = \frac{1}{\sqrt{\alpha_t}}(X_t - \frac{\beta_t}{\sqrt{1-\hat{\alpha}_t}}\varepsilon_\theta(X_t,t))$, where $\alpha_t := 1-\beta_t$, $\hat{\alpha}_t := \prod_{i=0}^{t}\alpha_i$, and $\varepsilon_\theta(X_t,t)$ represents the predicted noise component at timestep t.

This process provides a structured approach to model the data generation process as a diffusion phenomenon, with the forward process capturing the gradual introduction of noise and the reverse process learning to recover the underlying data structure. The neural network, trained to approximate the reverse process, serves as the backbone of our model's generative capabilities.

2.2 Network Architecture

Our SimDiff model leverages a Multilayer Perceptron (MLP) architecture, a type of feedforward artificial neural network that draws inspiration from biological neural systems and is parameterized by θ. The SimDiff model's training loss function is the Mean Squared Error (MSE).

Within the SimDiff framework, diffusion timestep embeddings and encoded samples are preprocessed through a linear layer with SiLU activation before aggregation. This aggregated representation is subsequently input into the MLP structure, which is designed to predict the noise component at the current timestep.

The embedding layer, a crucial component in deep learning models, represents categorical or discrete data as continuous vectors, known as embeddings. These embeddings capture the relationships and similarities among different categories or classes. Timestep embeddings are computed as $t_{\text{embedding}}(t,d) = \left[\sin\left(\frac{t}{1000\times 10^{2i/d}}\right), \cos\left(\frac{t}{1000\times 10^{2i/d}}\right)\right]$, where t represents the timestep, d denotes the dimension of the time embedding, and i is the index of the time embedding vector. The time embedding utilizes sine and cosine functions to encode diffusion step features through oscillations at various frequencies.

The output of the MLP predicts the mean noise from the current timestep t observation X_t through the sequence of operations: $X_t^{\text{embed}} = \mathit{LinearLayer}(X_t) + \mathit{LinearLayer}(t_{\text{embedding}})$,$\mathbf{h}_t = \text{SiLU}(\mathit{LinearLayer}(X_t^{\text{embed}}))$, and $\mu_\theta^t = \mathit{LinearLayer}(\mathbf{h}_t)$, where $\mathbf{h}_t$ represents the hidden intermediate vector, and the SiLU activation function ensures a non-linear relationship between the input and output. The predicted mean noise at timestep t, denoted as μ_θ^t, is the output of the final linear layer.

The denoising process at timestep t utilizes this predicted mean noise along with a constant variance σ_t^2, which is derived from the diffusion process as $\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. The predicted noise at timestep t for the observation X_t is $\varepsilon_\theta(X_t,t) = \mu_\theta^t + \sigma_t\mathcal{N}(0,\mathbf{I})$.

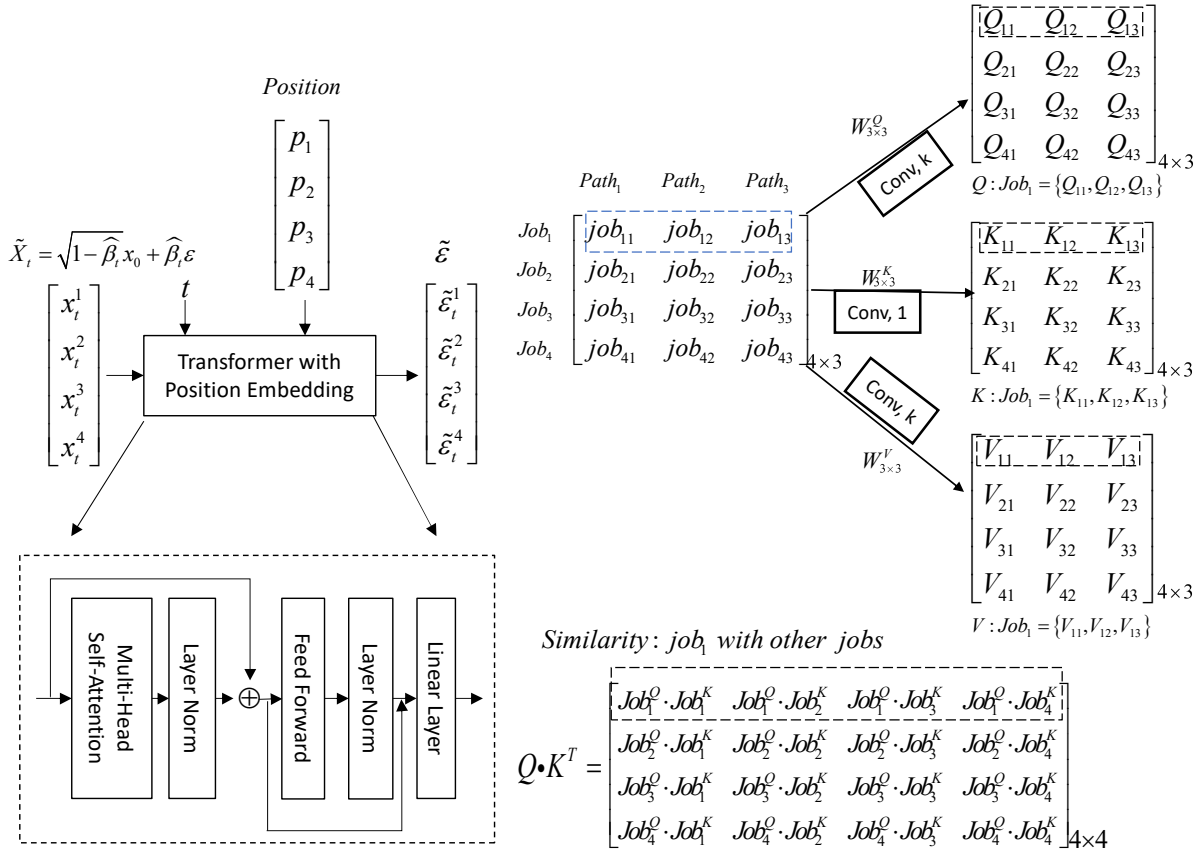
3 TRANSFORMER COMPONENTS AND SIMDIFF-CONVTRANS

3.1 SimDiff-ConvTrans Overview

SimDiff-ConvTrans is specifically designed to capture the dependencies between jobs that occur at long distances $x = \{x^1, x^2, \cdots, x^p\}$ in the jobs' timeline where $p > 1$. This is achieved by leveraging the self-attention mechanism within the Transformer component, which enables the model to consider all positions in the timeline simultaneously, rather than processing them sequentially as in a traditional MLP. This property of the Transformer is particularly useful when modeling complex stochastic processes, where dependencies between distant events can be crucial for accurate predictions.

In the SimDiff-ConvTrans architecture, the input embedding is modified to incorporate a convolutional transformation $\mathit{Conv}(X_1, \mathit{Position}_1)$. This transformation serves to extract relevant features from the input data before it is fed into the Transformer. The resulting feature vectors are then passed to the encoder and decoder components of the Transformer, which are responsible for understanding the temporal dependencies and generating the corresponding output sequences, respectively.

During the training process, the SimDiff-ConvTrans model learns to associate specific convolutional patterns with desired output sequences. By doing so, it gains the ability to generate realistic and coherent sequences that capture the underlying structure and dynamics of the stochastic process being modeled.



(a) convolutional transformer (b) self-attention mechanism with convolution

Figure 3: The convolutional transformer (left) and self-attention mechanism (right) in SimDiff-ConvTrans.

Additionally, the model's flexibility allows it to handle a wide range of input data formats and dimensions, making it a versatile tool for various applications in the field of stochastic process modeling.

Thus, the Convolutional Transformer, as shown in Figure 3a, enhances the relational coherence among job tasks by leveraging the transformative capabilities of the conventional Transformer model. The Transformer model has garnered significant acclaim in the domain of natural language processing due to its superior performance in sequence modeling. Central to the Transformer's efficacy is the self-attention mechanism, which facilitates global context awareness during the processing of input sequences. The architecture of the Transformer is primarily composed of self-attention layers, multi-head attention blocks, positional encodings, and feed-forward network layers.

The effective principles: the ability to discern local dependencies through convolutions and to maintain global interdependencies through the self-attention mechanism. The integration of convolutional operations within the Convolutional Transformer serves to compactly capture temporal correlations intrinsic to individual sample paths. This addition is complemented by the utilization of a Positional Encoding module, which is essential for preserving the sequential order information within the data.

3.2 Self-Attention Mechanism and Multi-Head Self-Attention

The self-attention mechanism in the Transformer model, shown in Figure 3b, helps analyze how parts of an input sequence relate to each other. For a sequence $X = \{x^1, x^2, \ldots, x^p\}$, self-attention measures how each element interacts with others. It uses query (Q), key (K), and value (V) vectors, calculated as $Q = W^Q X, K = W^K X, V = W^V X$, where W^Q, W^K, and W^V are the weight matrices for Q, K, and V. The

attention scores, Z, are found by normalizing the QK^T dot product with the key dimension's square root, $Z = \text{softmax}(QK^T/\sqrt{d_k})V$. The softmax function turns these scores into a probability distribution that shows the importance of each value vector V for each query Q. Weight matrices start with random values, which are optimized during training.

We've adapted the Transformer by adding convolutional operations with larger kernel sizes to better understand the connections between different parts of the input matrix (Peebles and Xie 2023).

The effective principles: the Multi-Head Self-Attention layer can capture various relationships within a sequence. It uses multiple attention functions in parallel, each with its own Q, K, and V matrices. The outputs from each head are combined into a single output by concatenating them and applying a final linear transformation $Z = Concatenate(Z_1, Z_2, \ldots)W_0$, where Z_i are the individual head outputs and W_0 combines them.

3.3 Timestep Embedding and Position Embedding

To address the lack of positional sensitivity in the self-attention mechanism, the Transformer architecture incorporates Positional Embedding. This component assigns a unique representation to each position in the sequence, enabling the model to comprehend the order of elements within the sequence. A common approach to positional encoding involves the use of sine and cosine functions, similar to those used in timestep embeddings: $P_{\text{embedding}}(p,d) = \left[\sin\left(\frac{p}{1000\times 10^{2i/d}}\right), \cos\left(\frac{p}{1000\times 10^{2i/d}}\right)\right]$.

In diffusion models, it is imperative for the model to discern the current diffusion timestep t to effectively reverse the diffusion process and reconstruct data from noise. Timestep embeddings provide a mechanism for the model to represent and leverage temporal information, enabling it to learn how to handle data x_t at different timesteps and generate more accurate and coherent outputs x_{t-1}.

Within the Transformer model, capturing the sequential information of elements at the current timestep t is crucial. Positional embeddings generate a unique vector representation for each position in the input sequence (e.g. the order of people in a queue), encapsulating relative or absolute positional information of that position within the sequence.

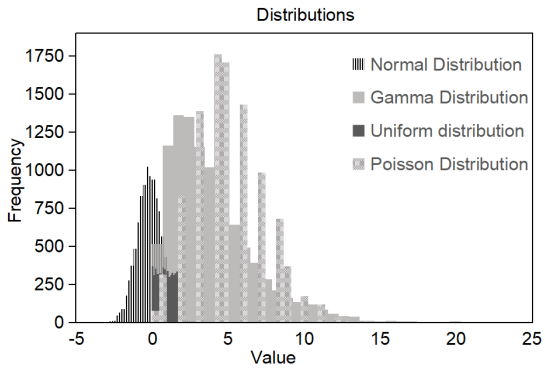
Hence, at a given timestep t within the diffusion process, the input X_t to the Transformer model is a composite of the embeddings representing the current timestep t and the positions of the elements within that timestep. The timestep embedding $t_{\text{embedding}}(t,d)$ captures the temporal dynamics, while the positional embeddings $P_{\text{embedding}}(p,d)$ encapsulate the spatial arrangement of the elements. The aggregated representation is formalized as $X_t = \{x_t^1, x_t^2, \ldots, x_t^p\} + t_{\text{embedding}}(t,d) + \sum_{p=1}^{P} P_{\text{embedding}}(p,d)$, where x_t^p represents the p-th element's embedding at timestep t, and P denotes the total number of elements in the sequence.

The effective principles: Timestep Embedding and Position Embedding can provide extra positional information in the simulation input sequence. The order and position between elements encode the position information into a vector and adds it to the input sequence, so that the model can distinguish the elements in different positions and make full use of this information to optimize the prediction results.

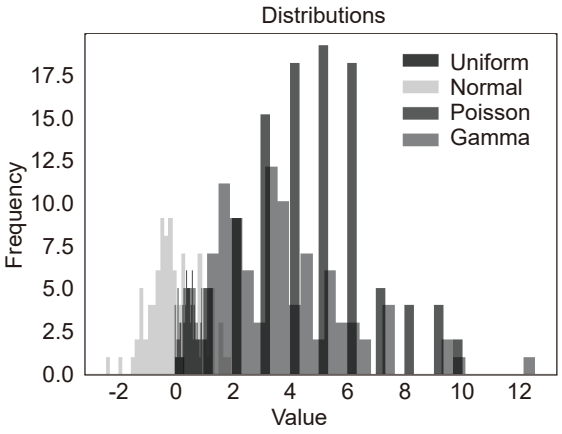
3.4 Feed-Forward Network

The Feed-Forward Network (FFN) layer follows each attention sub-layer in the Transformer. This layer typically consists of two linear transformations with a ReLU activation function in between. Given an input x, the FFN computes the output as $\text{FFN}(Z_{\text{multiple}}) = \text{LinearLayer}(\text{ReLU}(\text{LinearLayer}(Z_{\text{multiple}})))$, $\quad where\ \text{ReLU} = Max(0,x)$. The ReLU activation function introduces a simple form of non-linearity by returning the input value when it is greater than or equal to zero and outputting zero otherwise. This straightforward mechanism facilitates optimization during training and helps mitigate the vanishing gradient problem.

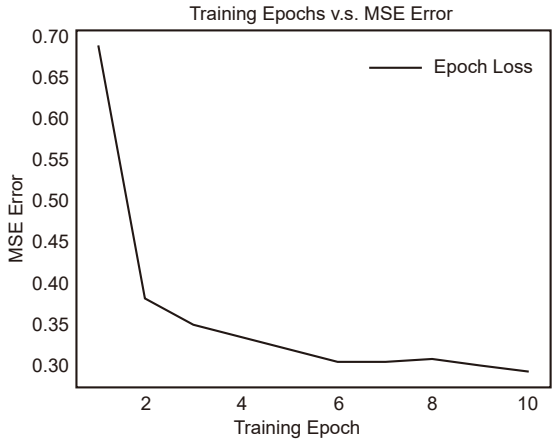
As depicted in Figure 3b, the Transformer architecture effectively models sequence data through the synergistic application of the multi-head attention layer, positional encoding, and feed-forward network



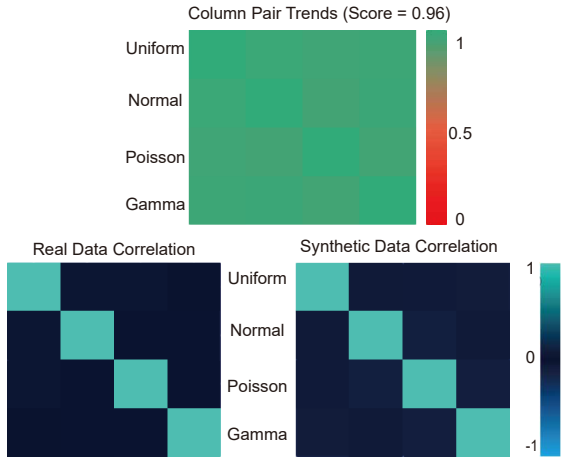
(a) Raw data distribution



(b) Generated data distribution



(c) Training loss



(d) Correlation coefficient differences

Figure 4: The SimDiff performance of simple simulation input distribution.

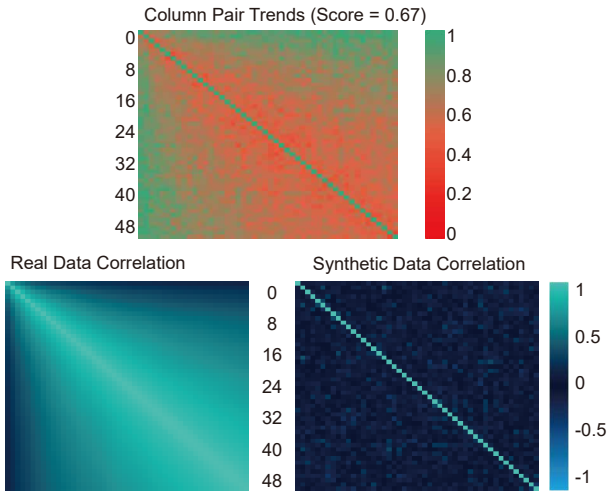
layer. The model is adept at predicting the estimated noise at the current diffusion step, denoted as $\hat{\varepsilon}(x_t, t)$. The loss function is also the Mean Squared Error (MSE).

4 EXPERIMENTS

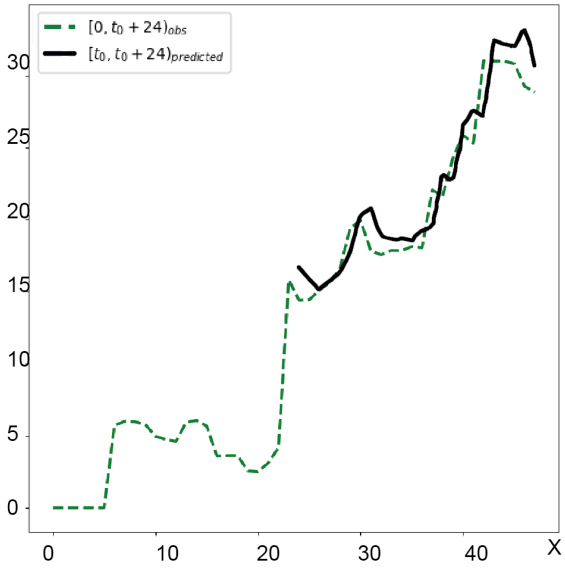
To ensure that the synthetic data generated maintains statistical consistency with the empirical data, this section employs both visual distribution comparison and quantitative assessment metrics. Specifically, we utilize the "KSComplement Metric" (also known as job fidelity/fidelity column) for single-column data and the "Column Pair Trends" analysis (also called as sample path fidelity/fidelity row) for multi-column data (Patki et al. 2016). Off-diagonal elements depict the degree of linear correlation between different elements within the same row of the original and synthetic data. These elements reveal the interplay between variables, uncovering underlying synthetic data structures and patterns.

4.1 Simple Simulation Input Distribution

In our initial study, we test SimDiff with common numerical distributions—Uniform[0,1], Gaussian[0,1], Poisson[$\lambda = 5$], and Gamma[2,2]—to understand its performance. We create many random samples from each distribution and process them with SimDiff. Then, we check how close SimDiff results are to the



(a) Training data and synthetic data coefficient differences.



(b) The first $t_0 = 24$ jobs are used for training data, and the last 24 jobs test for fitting trends.

Figure 5: The SimDiff-ConvTrans performance of queuing delays in a GI/G/1 system.

expected values. We make a table with 10,000 rows and 4 columns, each column for a different distribution. This helps SimDiff learn from four distributions at once, improving its ability to apply to new situations.

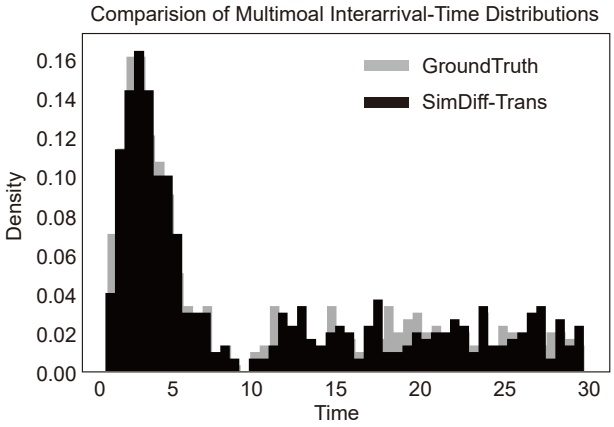
The results are shown in Figure 4. SimDiff handles these basic distributions well (Figures 4a and 4b show the approximate distributions, and Figure 4d compares the differences directly). It performs well in accuracy and consistency, as seen in Figure 4c. These results prove SimDiff's potential for solving more complex problems. However, these tests don't show how SimDiff will perform in real-world situations. So, in the next sections, we will test SimDiff with more varied and complex data to fully understand and use this powerful tool.

4.2 Queuing Delays in a GI/G/1 System

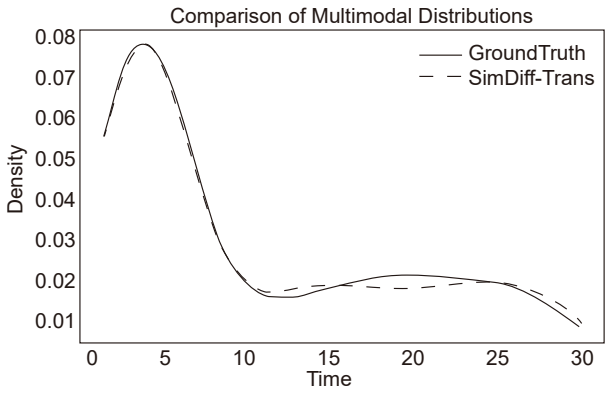
We are conducting an in-depth analysis of the queuing delay sequences within a GI/G/1 system, a model that can be accurately represented using either the Lindley recursion or through direct queue simulation. We choose to model interarrival times with a Gamma[0.25, 4] and service times with Gamma[0.5,1].

To train the SimDiff-ConvTrans model, we generate a dataset comprising 4,500 sample sequences, each detailing the delay progression for the first 48 service requests. The model's performance is then rigorously evaluated against a validation set consisting of 100 genuine sample sequences. The KSComplement score, a measure of distribution similarity, is found to be 0.92, indicating a high degree of congruence between the synthetic and real data distributions.

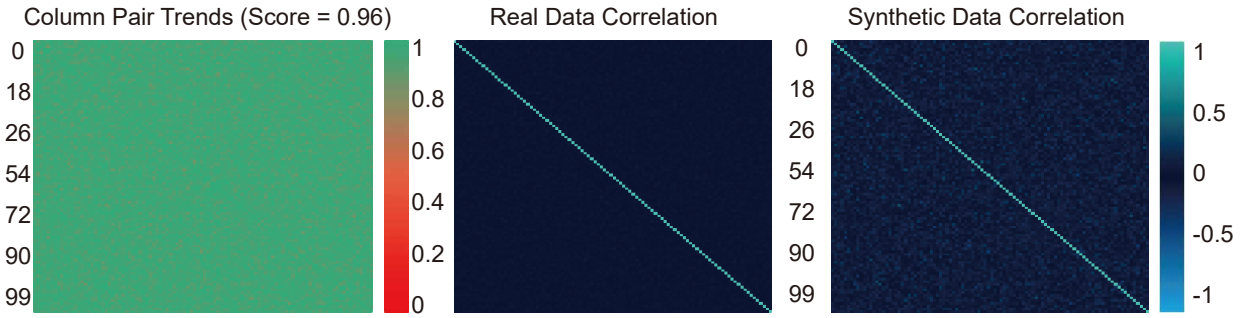
As illustrated in Figure 5a, the empirical correlation discrepancies are highlighted, with the main diagonal of the "real vs synthetic similarity" matrix showing a majority of 100% similarity scores, which further substantiates the high level of alignment between the model's outputs and the actual data. The distribution of waiting times across different jobs exhibits significant variation, attributable to the combined effect of interarrival times and service times, resulting in a hybrid data distributional state. Figure 5b demonstrates the data generation process for different jobs, where our method closely approximates the ground truth distribution, underscoring the efficacy of our approach in capturing the nuanced characteristics of the queuing system under study.



(a) Histogram comparison



(b) Densities curve



(c) Data Quality Column Pair Trends(Average Score=0.96)

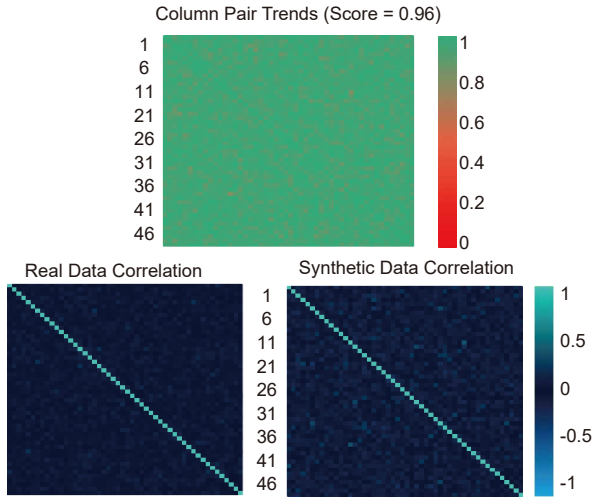
Figure 6: Performance of multimodal distributions in SimDiff-ConvTrans.

4.3 Multimodal Distributions

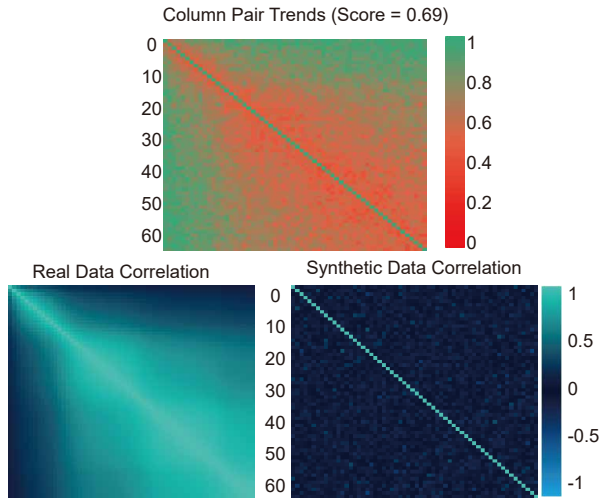
In our investigation of multimodal distributions within an i.i.d. context, we demonstrate the capability of SimDiff-ConvTrans to accurately capture a multimodal interarrival-time distribution through a gradual adjustment of its parameters. We examine a mixture distribution consisting of two components: a Gamma[2.875, 0.5] and a Uniform[10,20] under mixing weights 0.6 and 0.4.

The empirical results indicate a KSComplement score of 0.91 and a Column Pair Trends score of 0.96, both of which affirm the model's proficiency in closely mirroring the complex characteristics of the multimodal distribution. As depicted in Figure 6, the empirical densities for the Gamma-Uniform mixture are well-captured by SimDiff-ConvTrans, particularly for the initial set of 0 to 10 jobs, which are predominantly governed by the Gamma distribution. Following this, as the model has internalized the established distribution, it undergoes a gradual transition towards the Uniform distribution state. Consequently, there is a discernible divergence around the 20th job, indicative of the model's adaptive learning process. By the 25th to 30th jobs, the model has fully assimilated the new distributional traits, resulting in a near-perfect fit.

This transition period exemplifies the model's capacity for dynamic adaptation and its ability to refine its output in response to shifts in the underlying data distribution. The successful simulation of the multimodal interarrival-time distribution by the SimDiff-ConvTrans model underscores its potential as a versatile tool for modeling complex, real-world scenarios where data distributions are subject to change.



(a) Data Quality Column Pair Trends(Average Score=0.95)



(b) Data Quality Column Pair Trends(Average Score=0.69)

Figure 7: Performance of emergency call center (left) and queueing simulation (right) in SimDiff-ConvTrans.

4.4 A Queueing Simulation

We inspect the overall impact of using data from the SimDiff-ConvTrans model to predict the waiting time, denoted as W_{60}, for the 60th task in a queue that follows a NHPP/Gamma/1 FIFO setup. The times between task arrivals are based on a non-uniform Poisson process (NHPP), with a rate given by the function $\lambda(t) = \frac{1}{2}\sin(\frac{\pi}{8}t) + \frac{3}{2}$. Concurrently, the task service times are i.i.d., following Gamma [1.2,0.4].

For the purpose of model training, we utilize datasets comprising 1,000 sample paths, each containing 50 observations for both interarrival times and service times. As depicted in Figure 7, we evaluate the W_{60} waiting time distribution across 4,000 simulation replications, juxtaposing the results obtained from the ground-truth inputs against those generated using the SimDiff-ConvTrans model. The results exhibit a high degree of congruence between the two simulation sets. Notably, while the training data sample paths encompass only 50 transactions, our simulations extend up to the 60th job, thereby illustrating the model's capacity to predict outcomes beyond the training data scope, provided that the system's nonstationarity is not excessive. This predictive capability surpasses the limitations of traditional trace-driven simulation approaches.

Further, Figure 7b present the KSComplement score of 0.91 and the Column Pair Trends score of 0.69 for the simulation. The primary values of 1 along the main diagonal of the "similarity" coefficient differences matrix confirm the efficacy of the SimDiff-ConvTrans method. The red regions in the figures, indicating scores less than 1 for other column pairs, reflect the complexity of the data distribution, which arises from the mixture of two distributions in the training data and the varying relationships between different columns, as also evidenced by the Pearson and Spearman rank correlation coefficients.

4.5 Emergency Call Center (Real World Dataset)

We extend the application of the SimDiff-ConvTrans model to a real-world dataset, focusing on the daily interarrival times of emergency calls for the San Francisco Fire Department (SFFD) from January 1, 2023, to March 21, 2024. This dataset presents a unique challenge due to the inherent complexity and variability of real-world data.

We treat each day as a separate sample path, totaling 446 distinct observations. The dataset is characterized by a minimum of 755 emergency calls (jobs) per day, with the interarrival times of the first 755 calls being utilized as the learning data for the model.

Table 1: Performance of compare methods on emergency call center (real world dataset).

	Our Model	NIM-VL	Gaussian Copula	CTGAN	TVAE
KSComplement Score	0.9347	0.4877	0.4959	0.6687	0.5051
Data Quality Column Pair Trends	0.9648	0.9769	0.9733	0.9709	0.9653
Overall Score	0.9498	0.7323	0.7346	0.8199	0.7352

Compared with methods NIM-VL(Cen et al. 2020) , Gaussian Copula(Patki et al. 2016), CTGAN(Zhao et al. 2021), TVAE(Liu et al. 2022) in Table 1, our model is exemplary, as evidenced by the KSComplement score of 0.93 (far better than others) and the Column Pair Trends score of 0.96 (similar to others). The overall scores 0.95 not only reflect a strong concordance between the model's synthetic predictions and the actual observed interarrival times but also underscore the model's ability to accurately capture the underlying patterns and trends despite the data's inherent complexity and uncertainty.

The high performance metrics achieved by SimDiff-ConvTrans are particularly significant given the dataset's challenging characteristics. The interarrival times for emergency calls are marked by considerable variability and non-stationarity, which complicates the modeling process. The model's capacity to learn and adapt to these patterns is a testament to its robustness and adaptability.

5 CONCLUSION

In this work, we present the SimDiff framework, which innovatively integrates denoising probabilistic diffusion with convolutional transformers to address the critical challenge of generating simulation input data. SimDiff is designed to effectively manage complex hybrid distributions, offering a robust solution for stochastic simulation. Its ability to capture long-range dependencies within data is crucial for modeling intricate processes, significantly enhancing input data generation. Although powerful, SimDiff has limitations, particularly in tail estimation and data extrapolation, necessitating careful application and validation. The SimDiff framework represents a significant advancement, making stochastic simulation more accessible and efficient, and we anticipate its broad application in various domains to aid simulation-based modeling and decision-making.

ACKNOWLEDGMENTS

This research was funded by the Science and Technology Innovation Committee of Shenzhen-Platform and Carrier (International Science and Technology Information Center), the Open Project of Anhui Province Key Laboratory of Special Heavy Load Robot (TZJQR006-2023), the Natural Science Research Key Project of Education Department of Anhui Provincial Government (2023AH051092), the Science and Technology Innovation Commission of Shenzhen (JCYJ20210324135011030, WDZC20200818121348001, JCYJ20210324115604012), Guangdong Pearl River Plan (2019QN01X890), High-end Foreign Expert Talent Introduction Plan (G2021032022L).

REFERENCES

Cen, W., E. A. Herbert, and P. J. Haas. 2020. "NIM: Modeling and Generation of Simulation Inputs via Generative Neural Networks". In *2020 Winter Simulation Conference (WSC)*, 584–595 https://doi.org/10.1109/wsc48552.2020.9383966.

Devroye, L. 2010. "Complexity Questions in Non-Uniform Random Variate Generation". In *Proceedings of COMPSTAT'2010*, edited by Y. Lechevallier and G. Saporta. Heidelberg: Physica-Verlag HD.

Dimri, T., S. Ahmad, and M. Sharif. 2020. "Time Series Analysis of Climate Variables Using Seasonal ARIMA Approach". *Journal of Earth System Science* 129(1):149.

Ghasempour, R., K. Roushangar, and F. Alizadeh. 2023. "Hybrid Models for Drought Forecasting: Integration of Multi Pre-processing-Data Driven Approaches and Non-linear GARCH Time Series Model". *Arabian Journal of Geosciences* 16(6):361.

Godahewa, R., G. I. Webb, D. Schmidt, and C. Bergmeir. 2023. "SETAR-Tree: A Novel and Accurate Tree Algorithm for Global Time Series Forecasting". *Machine Learning* 112(7):2555–2591.

Ho, J., A. Jain, and P. Abbeel. 2020. "Denoising Diffusion Probabilistic Models". In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Volume 33. Vancouver, BC, Canada: Curran Associates Inc.

Hong, J., Y. Yan, E. E. Kuruoglu, and W. K. Chan. 2023. "Multivariate Time Series Forecasting with GARCH Models on Graphs". *IEEE Transactions on Signal and Information Processing over Networks* 9:557–568.

Liu, T., Z. Qian, J. Berrevoets, and M. van der Schaar. 2022. "GOGGLE: Generative Modelling for Tabular Data by Learning Relational Structure". In *The Eleventh International Conference on Learning Representations*. May 1th-5th, Kigali, Rwanda.

Majidnia, M., Z. Ahmadabadi, P. Zolfaghari, and A. Khosravi. 2023. "Time Series Analysis of Cutaneous Leishmaniasis Incidence in Shahroud Based on ARIMA Model". *BMC Public Health* 23(1):1190.

Mielczarek, B. and J. Zabawa. 2016. "Modeling Healthcare Demand Using a Hybrid Simulation Approach". In *2016 Winter Simulation Conference (WSC)*, 1535–1546 https://doi.org/10.5555/3042094.3042290.

Nelson, B. L., A. T. Wan, G. Zou, and X. Zhang. 2021. "Reducing Simulation Input-Model Risk via Input Model Averaging". *INFORMS Journal on Computing* 33(2):672–684.

Palmer, G. I. and Y. Tian. 2023. "Implementing Hybrid Simulations that Integrate DES+SD in Python". *Journal of Simulation* 17(3):240–256.

Patki, N., R. Wedge, and K. Veeramachaneni. 2016. "The Synthetic Data Vault". In *2016 IEEE International Conference on Data Science and Advanced Analytics*. October 17th-19th, Montreal, QC, Canada, 399-410.

Peebles, W. and S. Xie. 2023. "Scalable Diffusion Models with Transformers". In *2023 IEEE/CVF International Conference on Computer Vision*. October 1th-6th, Paris, France, 4172-4182.

Shumway, R. H., D. S. Stoffer, and D. S. Stoffer. 2000. *Time Series Analysis and Its Applications*. New York: Springer.

Zhao, Z., A. Kunar, R. Birke, and L. Y. Chen. 2021. "CTab-GAN: Effective Table Data Synthesizing". In *Proceedings of The 13th Asian Conference on Machine Learning*. November 17th-19th, Istanbul, Turkey, 97–112.

AUTHOR BIOGRAPHIES

FENGWEI JIA is a Postdoctoral Fellow in Tsinghua Shenzhen International Graduate School at Tsinghua University, and Tsinghua-Berkely Shenzhen Institute, with research interests in Artificial Intelligent. His email address is fengweijia@sz.tsinghua.edu.cn.

HONGLI ZHU is a Postdoctoral Fellow in Tsinghua Shenzhen International Graduate School at Tsinghua University, and Tsinghua-Berkely Shenzhen Institute, with research interests in System Simulation. Her email address is honglizhu@sz.tsinghua.edu.cn.

FENGYUAN JIA is a lecturer at the School of Mechanical Engineering at Anhui University of Technology, Anhui Province Key Laboratory of Special Heavy Load Robot, and Anhui Province Engineering Laboratory of Intelligent Demolition Equipment, with research interests in Unmanned Aerial Vehicle and Machine Learning. His email address is jiafengyuan@ahut.edu.cn.

XINYUE REN is a doctoral student in Tsinghua Shenzhen International Graduate School at Tsinghua University, and Tsinghua-Berkely Shenzhen Institute, with research interests in Time Series Analysis and System Simulation. Her email address is renxy21@sz.tsinghua.edu.cn.

HONGMING TAN is a doctoral student in Tsinghua Shenzhen International Graduate School at Tsinghua University, and Department of Network Intelligence at Peng Chen Laboratory, with research interests in Time Series Analysis and System Simulation. His email address is thm22@sz.tsinghua.edu.cn.

SIQI CHEN is a doctoral student in Tsinghua Shenzhen International Graduate School at Tsinghua University, with research interests in Time Series Analysis and System Simulation. Her email address is chensq22@sz.tsinghua.edu.cn.

WAI KIN VICTOR CHAN is an Professor in Tsinghua Shenzhen International Graduate School at Tsinghua University, Tsinghua-Berkely Shenzhen Institute, and International Science and Technology Information Center, with research interests in Industrial Engineering, Operations Research, Statistics, Computer Science and Electrical Engineering. His email address is chanw@sz.tsinghua.edu.cn.