Population Aware Diffusion for Time Series Generation

Yang Li[1], **Han Meng**[1], **Zhenyu Bi**[2], **Ingolv T. Urnes** [3], **Haipeng Chen** [1]

[1]William & Mary
[2]Virginia Tech
[3]Generated Health
[1] {yli102, hmeng, hchen23}@wm.edu, [2] zhenyub@vt.edu,[3] ingolv.urnes@generatedhealth.com

Abstract

Diffusion models have shown promising ability in generating high-quality time series (TS) data. Despite the initial success, existing works mostly focus on the authenticity of data at the *individual* level, but pay less attention to preserving the *population*-level properties on the entire dataset. Such population-level properties include value distributions for each dimension and distributions of certain functional dependencies (e.g., cross-correlation, CC) between different dimensions. For instance, when generating house energy consumption TS data, the value distributions of the outside temperature and the kitchen temperature should be preserved, as well as the distribution of CC between them. Preserving such TS population-level properties is critical in maintaining the statistical insights of the datasets, mitigating model bias, and augmenting downstream tasks like TS prediction. Yet, it is often overlooked by existing models. Hence, data generated by existing models often bear distribution shifts from the original data. We propose **P**opulation-**a**ware **D**iffusion for **T**ime **S**eries (**PaD-TS**), a new TS generation model that better preserves the population-level properties. The key novelties of PaD-TS include 1) a new training method explicitly incorporating TS population-level property preservation, and 2) a new dual-channel encoder model architecture that better captures the TS data structure. Empirical results in major benchmark datasets show that PaD-TS can improve the average CC distribution shift score between real and synthetic data by 5.9x while maintaining a performance comparable to state-of-the-art models on individual-level authenticity.

Code — https://github.com/wmd3i/PaD-TS

1 Introduction

Time series data exists in a broad spectrum of real-world domains, spanning healthcare (Kaushik et al. 2020; Morid, Sheng, and Dunbar 2023), energy (Priesmann et al. 2021; Deb et al. 2017), finance (Zeng et al. 2023; Masini, Medeiros, and Mendes 2023), and many more. TS models have been used in these domains for effective data analysis and prediction tasks. Developing such models requires rich and high-quality TS datasets, which unfortunately may not exist in many data-scarce domains like healthcare and energy. Various data augmentation techniques (e.g., jittering, scaling, permutation, time warping, and window slicing) have been developed to enhance the original datasets with synthetically generated TS data. Synthetic data can potentially create observations that do not exist but are close to the original dataset (Coletta et al. 2023; Esteban, Hyland, and Rätsch 2017). With a newly augmented dataset, we can further enhance TS models for data analysis, while protecting the privacy and confidentiality of the original data and potentially enhancing data sharing.

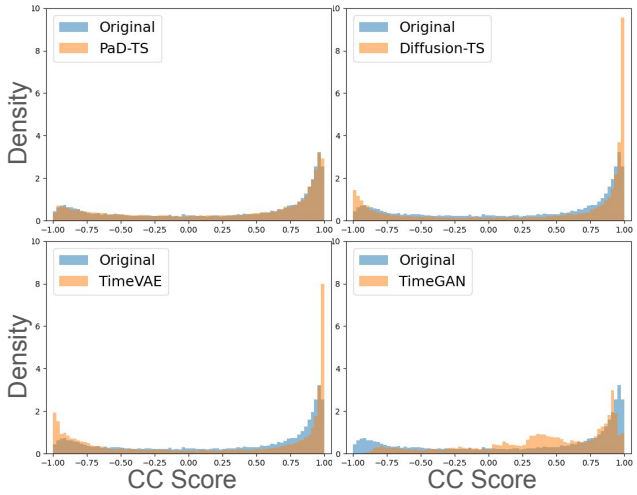

Figure 1: Histogram of CC distribution between the original and synthetic Energy datasets. The CC values are calculated between the outside temperature and kitchen temperature. PaD-TS (top left) best preserves such functional dependency distribution. Previous models tend to generate data points with a CC score close to 1 or -1, which leads to biases for downstream tasks.

How do we measure the quality of synthetic TS data? First, it should be authentic on the *individual* level. Given two samples – one from the original set and another from the generated set – we should not be able to identify which is fake or real. Second, it should preserve the TS *population*-level properties of the original data. Such population-level properties include distributions for each dimension of

the data and distributions of certain functional dependencies (e.g., CC) between different dimensions of the data. Taking house energy consumption TS as an example (Candanedo, Feldheim, and Deramaix 2017), the value distribution of the outside temperature and the kitchen temperature, as well as the CC distribution between them should be preserved. As shown in Figure 1, previous methods trust the model to estimate the CC distribution between them which fails to preserve the same distribution in generated TS. Preserving both the individual- and population-level properties is crucial in maintaining the standalone and statistical insights of the original data, hence reducing model bias and further augmenting downstream tasks such as prediction.

We revisit the TS generation problem with an emphasis on preserving the TS population-level properties. There have been extensive studies on TS generation using generative adversarial networks (GANs) (Yoon, Jarrett, and Van der Schaar 2019; Liao et al. 2020; Mogren 2016; Esteban, Hyland, and Rätsch 2017; Pei et al. 2021) and variational autoencoders (VAEs) (Desai et al. 2021; Naiman et al. 2023). Though they have achieved reasonable results, it is known that GANs suffer from unstable training because of the need to interactively and iteratively train both the generator and discriminator, while VAEs normally generate lower-quality samples due to optimizing an approximate objective via the evidence lower bound (ELBO). Moreover, both GANs and VAEs may struggle with the mode collapse issue. Diffusion models (DMs) emerge as another class of powerful generative models that are robust against mode collapse, and show state-of-the-art performances in domains such as image generation (Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021; Peebles and Xie 2023; Sohl-Dickstein et al. 2015), text-to-image generation (Ramesh et al. 2022; Nichol et al. 2021), and text-to-video generation (Brooks et al. 2024). In light of this, recent studies have developed DM-based TS generation models (Coletta et al. 2023; Yuan and Qiao 2024) that yield better results than GANs and VAEs for TS generation. Despite the initial success, most existing works pay less attention to the preservation of population-level properties and hence may suffer from the generation distribution shift.

We hypothesize that the distribution shift of existing TS generative models comes from two sources: 1) TS population-level property preservation is not explicitly incorporated into the training process, as they only try to capture TS population-level properties by minimizing the value distance between synthetic and original samples. InfoVAE (Zhao, Song, and Ermon 2017) tackles a similar issue in image generation by penalizing the distribution shift in the single encoded latent space with a regularization term in its training loss. However, this cannot be directly applied to DMs, as DMs follow an iterative generation framework usually with an extremely long series of latent spaces. 2) Model architectures of existing works cannot fully capture the multivariate TS data information. Cross-dimensional information has been shown to be critical (Liu et al. 2024) for TS prediction, as it can yield great performance using only dimension information. Previous methods either neglect cross-dimension features (Yuan and Qiao 2024; Yoon, Jarrett, and Van der Schaar 2019; Desai et al. 2021) or try to capture them in a shared block with temporal feature extractions (Coletta et al. 2023; Tashiro et al. 2021), which may not be sufficient to capture all cross-dimension features.

We propose PaD-TS, a new DM that addresses the above issues. PaD-TS comes with a new DM training objective that penalizes population-level distribution shifts. This is enabled by a new sampling strategy (during training) that enforces the comparison of two distributions to the same diffusion step in a mini-batch. In addition, we design a new transformer (Peebles and Xie 2023; Vaswani et al. 2017) encoder-based dual-channel architecture that can better capture the population-level properties.

Our main contributions are as follows. 1) We are the first to study DM-based TS generation that explicitly considers TS population-level property preservation, along with new metrics to evaluate it. 2) We propose PaD-TS, a novel DM that addresses the technical challenges of this problem. 3) We conduct extensive empirical evaluations of our model. The empirical results show that PaD-TS achieves state-of-the-art performance in population-level property preservation and comparable individual-level authenticity.

2 Related Work

GANs (Goodfellow et al. 2014) are generative models that usually consist of a generator and a discriminator. The generator generates plausible data to fool the discriminator, whereas the discriminator tries to distinguish the synthetic data from real data. Due to their ability to generate high-quality synthetic data (Mogren 2016; Yoon, Jarrett, and Van der Schaar 2019; Liao et al. 2020), GANs have been extensively studied for TS generation (Mogren 2016; Yoon, Jarrett, and Van der Schaar 2019; Liao et al. 2020) as well as functional dependency persevation tables (Chen et al. 2019). However, GANs suffer from inherent instability, resulting from the interactive and iterative training process on both the generator and the discriminator. This often leads to non-converging models that oscillate or vanishing gradient issues that prevent the generator from learning meaningful patterns.

VAEs (Kingma and Welling 2022) are another family of generative models based on the encoder-decoder architecture. The encoder in VAEs encodes the data in a latent space following a Gaussian distribution. The decoder samples from learned latent space as prior information to generate synthetic data. TimeVAE (Desai et al. 2021) utilizes convolution structures to capture temporal correlations. KoVAE (Naiman et al. 2023) uses a linear Koopman-based prior and a sequential posterior to further improve the performance. However, VAEs are known for their low generation quality and susceptibility to mode collapse, which limits their success in TS generation.

DMs (Ho, Jain, and Abbeel 2020; Sohl-Dickstein et al. 2015) emerge as a new generative framework that learns to generate data by gradually reversing a noising process applied to the training data. Being theoretically grounded with connections to score-based generative modeling (Song et al. 2020), and having robustness against mode collapse compared to other generative models like GANs and VAEs, they

are arguably the state-of-the-art methods in image generation (Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021; Peebles and Xie 2023; Sohl-Dickstein et al. 2015), text-to-image generation (Ramesh et al. 2022; Nichol et al. 2021), and text-to-video generation (Brooks et al. 2024). Recent studies have developed DM-based models for TS generation (Coletta et al. 2023; Yuan and Qiao 2024) that outperform GAN- and VAE-based models, further demonstrating the superior performance of DMs. Diffusion-TS (Yuan and Qiao 2024) uses trends, seasonal architectures, and Fourier-based objects, which outperform previous models by a significant margin. TimeLDM (Qian et al. 2024) combines VAE and the diffusion framework. In addition to TS generation, DMs have been widely used for TS prediction (Feng et al. 2024; Li et al. 2022; Rasul et al. 2021; Alcaraz and Strodthoff 2022; Yang et al. 2024) and imputation (Tashiro et al. 2021), extremely long TS generation (Barancikova, Huang, and Salvi 2024), and general pre-trained models for TS prediction (Yang et al. 2024). These works do not explicitly consider the preservation of population-level properties and hence may suffer from the generation shift, a key issue that we are going to address in this paper.

3 Problem Statement

Given a multivariate TS dataset $\mathcal{D}_{\text{orig}} = \{x_n\}_{n=1}^N$ with N samples. Each data sample $x^{1:L;1:F} \in \mathbb{R}^{L\times F}$ is a multivariate TS, where L is the sequence length and F is the number of features/dimensions (e.g., we can denote $\{x^{l;i}\}$ for all $l \in [1, L]$ as values in the l-th dimension). Our task is to generate synthetic dataset $\mathcal{D}_{\text{syn}} = \{\hat{x}_n\}$ such that the synthetic data is similar to the original data $\mathcal{D}_{\text{orig}}$ in individual level and follows original population-level property distributions. To evaluate the generation quality at the individual level, we have the following $metric$:

(1) *Discriminative Accuracy (DA)* (Yoon, Jarrett, and Van der Schaar 2019) is based on the post-hoc machine learning classifier (clf) trained with the training set from original and synthetic datasets (with label real=1; synthetic = 0). Then DA is the model performance on the test set with size S using the following equation:

$$\text{DA} = \Big| \frac{\sum_{n=1}^{S}(0=\text{clf}(\hat{x}_n)) + \sum_{n=1}^{S}(1=\text{clf}(x_n))}{2S} - 0.5 \Big| \quad (1)$$

where $\hat{x}_n$ and x_n are test samples. To evaluate the generation quality in terms of TS population-level property preservation, we propose two new *metrics*:

(2) *Value distribution shift (VDS)*:

$$\text{VDS} = \frac{1}{F}\sum_{i=1}^{F} D(P_V^i, Q_V^i) \quad (2)$$

where D stands for a certain distribution distance measure (e.g., KL divergence); P_V^i is the value distribution of i-th dimension over the original data ; and Q_V^i is the counterpart distribution for the synthetic dataset.

(3) *Functional dependency distribution shift (FDDS)*:

$$\text{FDDS} = \frac{1}{M}\sum_{m=1}^{M} D(P_{\text{FD}}^{i,j}, Q_{\text{FD}}^{i,j}) \quad (3)$$

where $P_{\text{FD}}^{i,j}$ is the distribution of the functional dependency scores between i-th and j-th dimension over the original data which can be calculated by any functional dependency function of interest $f : x^{1:L;i} \times x^{1:L;j} \to \mathbb{R}$ (e.g., cross-correlation CC, mutual information, etc.); Q_{FD} is the counterpart distribution for the synthetic dataset; and M represents the possible pairs of functional dependencies.

New metrics (VDS and FDDS) with reasonable distribution distance measure a more general similarity between real and synthetic population-level property distributions. They yield better performance in detecting potential biases and shifts in generated samples than aggregated statistics-based metrics (e.g., distance between property mean).

4 Approach

In this section, we introduce PaD-TS which addresses population-level preservation problems. Building on top of diffusion models, PaD-TS consists of two novel components: a new population-aware training process, and a new dual-channel encoder model architecture.

Preliminary: Diffusion Models

We briefly review the formulations of the denoising diffusion probabilistic models (DDPMs) (Ho, Jain, and Abbeel 2020; Nichol and Dhariwal 2021) which have iterative *forward* and *reverse* processes.

Given original data $x^0 \sim q(x^0)$, the *forward* process q is a Markov process which adds noise ϵ^t iteratively based on a fixed variance scheduler β^t and sampled diffusion step t. Normally, t is uniformly sampled from $[1, T-1]$ for each sample data.

$$\begin{aligned} q(x^{1:T}|x^0) &= \prod_{t=1}^{T} q(x^t|x^{t-1}) \\ q(x^t|x^{t-1}) &= \mathcal{N}(\sqrt{1-\beta^t}x^{t-1}, \beta^t\mathbf{I}) \end{aligned} \quad (4)$$

The *reverse* process (p_θ) gradually removes noises from $p(x^T) \sim \mathcal{N}(0, \mathbf{I})$ and tries to recover x^0 with $x^0(\theta)$.

$$\begin{aligned} p_\theta(x^{0:T}) &= p(x^T)\prod_{t=1}^{T} p_\theta(x^{t-1}|x^t) \\ p_\theta(x^{t-1}|x^t) &= \mathcal{N}(\mu_\theta, \Sigma_\theta) \end{aligned} \quad (5)$$

where mean μ_θ and variance Σ_θ are learnable parameters.

To reduce complexity, DDPMs set $\Sigma_\theta = \sigma_t^2\mathbf{I}$ where $\sigma_t^2 = \beta^t$ follows the same variance scheduler as the forward process. With fixed Σ_θ, one can effectively approximate the reverse process by training a model that predicts μ_θ with:

$$\tilde{\mu}^t(x^t, x^0) = \frac{\sqrt{\bar{\alpha}^{t-1}}\beta^t}{1-\bar{\alpha}^t}x^0 + \frac{\sqrt{\alpha^t}(1-\bar{\alpha}^{t-1})}{1-\bar{\alpha}^t}x^t \quad (6)$$

where $\alpha^t = 1-\beta^t$ and $\bar{\alpha}^t = \prod_{s=1}^{t}\alpha_s$ are both constants.

Note that the only unknown in $\tilde{\mu}^t(x^t, x^0)$ is x^0. A neural network can directly model towards x^0 or ϵ^t when applying reparametrization trick $x^0 = \frac{1}{\sqrt{\bar{\alpha}^t}}(x^t - \sqrt{1-\bar{\alpha}^t}\epsilon^t)$. With target x^0, DDPMs have the following training objective:

$$L_0(\theta) = \mathbb{E}_{t,x^0}\Big[\|x^0 - x^0(\theta)\|^2\Big] \quad (7)$$

Algorithm 1: PaD-TS training procedure

Input: Original TS data, FD function f, epochs E, and total diffusion steps T

Output: Trained PaD-TS model θ

1: **for** $i = 1$ to E **do**
2: Sample a mini-batch of x^0 with b samples
3: Sample $t_1 \in [1, T-1]$ ▷▷▷ SSS
4: Let $t = [t_1, ..., t_1]$
5: Get $\hat{x}^0$ using PaD-TS model ▷▷▷ PAT objective
6: Find all FD distributions for $\hat{x}^0$ and x^0
7: Calculate L_0 and L_{pop}
8: Update θ with gradient $\nabla_\theta(L_0 + L_{pop})$
9: **end for**
10: **return** Model θ

PaD-TS Training

As mentioned above, existing DMs show promising performance in individual-level authenticity but exhibit suboptimal performance in preserving the TS distribution of population-level properties. One hypothesis is that the original DDPM training process focuses on the value distance between model input and output, and overlooks TS population-level properties preservation. A naive resolution to this is to penalize distribution shifts by regularizing the loss function. However, this turns out to be not directly feasible because of the iterative DM generation process. To address this technical challenge, we propose a new DM training process that enables applying any regularization of interest for DMs which consists of two components: population aware training (PAT) objectives and same diffusion step samplings (SSS).

Algorithm 1 shows our new DM training procedure for PaD-TS: (1) We first sample a mini-batch of data from the original TS dataset in Line 2. (2) In Lines 3-4, we use the SSS for the mini-batch diffusion step t. (3) In Lines 5-8, we use the PAT objective to update the model parameter θ. The details of the training procedure are as follows.

Population Aware Training Objective This objective considers preserving the TS population-level property distribution rather than simple statistical measures (e.g., mean, variance, etc). Thus it is crucial to use the right distribution distance. The distribution of property at the population level may come in arbitrary parametric forms, thus distribution shift measures such as Kullback-Leibler divergence (Zhao, Song, and Ermon 2017; Liu and Wang 2016) and the Wasserstein distance (Arjovsky, Chintala, and Bottou 2017) are inappropriate, as in practice they usually either have assumptions toward the underlying distributions and/or is computationally expensive, especially with high dimensional data. Inspired by InfoVAE (Zhao, Song, and Ermon 2017), we use the Maximum Mean Discrepancy (MMD) (Gretton et al. 2012) as the distribution distance. It is commonly used in deep learning tasks such as image generation (Zhao, Song, and Ermon 2017; Li et al. 2017) and domain adaptation (Yan et al. 2017; Cao, Long, and Wang 2018).

With a pair of arbitrary distributions (P and Q), MMD compares all their moments using the selected kernels. Using a Radial Basis Function (RBF) kernel on multiple window sizes as an example, the MMD distance can be efficiently estimated as follows:

$$\begin{aligned}\text{MMD}_W(Q||P) = &\sum_{w_i \in W} \mathbb{E}_{Q,Q}\big[\text{RBF}_{w_i}(Q,Q)\big] + \\ &\sum_{w_i \in W} \mathbb{E}_{Q,P}\big[\text{RBF}_{w_i}(Q,P)\big] + \sum_{w_i \in W} \mathbb{E}_{P,P}\big[\text{RBF}_{w_i}(P,P)\big]\end{aligned} \tag{8}$$

where RBF_{w_i} stands for a RBF kernel with window w_i.

We use cross-correlation (CC, see definition in Appendix A) as an example of TS functional dependency, as CC is often a critical property in TS data. For each TS data sample $x^{1:L;1:F}$, we can calculate $M = \frac{F(F-1)}{2}$ unique CC values. We use $P_{\text{CC}}^{i,j}$ to represent the distribution of CC between i-th and j-th dimension in the original data, and use $Q_{\text{CC}}^{i,j}$ to represent its counterpart for the synthetic data. By considering all possible pairs of CC distributions, the regularization loss term can be defined as:

$$L_{pop} = \frac{1}{M} \sum_{m=1}^{M} \text{MMD}_W(P_{\text{CC}}^{i,j}, Q_{\text{CC}}^{i,j}) \tag{9}$$

Hence, the PAT objective can be formally defined as follows:

$$L_{total} = L_0 + \alpha * L_{pop} \tag{10}$$

where α is a hyperparameter that controls the weight of the population aware loss.

Same Diffusion Step Sampling How do we empirically compute a meaningful L_{pop} in DMs? As mentioned, the DM framework is an iterative generation process (i.e., gradually removing noise with a variance scheduler β^t) that behaves differently at each diffusion step t. A common uniform diffusion step sampling strategy (Yuan and Qiao 2024; Ho, Jain, and Abbeel 2020; Coletta et al. 2023; Peebles and Xie 2023) and importance sampling (Nichol and Dhariwal 2021) will produce mixed diffusion step sampling within a mini-batch. If the resulting diffusion steps of one of the two strategies are applied, DM will yield generations from mixed diffusion steps. Although it works for value distribution preservation, it will be problematic to compare functional dependency distributions because of different behaviors at different diffusion steps. To ensure a reasonable functional dependency distribution comparison in DM, we introduce the SSS strategy.

Given a mini-batch training procedure with b samples. We have diffusion step vector $t = [t_1, t_2, ..., t_b]$, where each $t_i \in [0, T-1]$. SSS first samples $t_1 \in [1, T-1]$ and duplicates the diffusion step t_1 to fill the vector t. Thus we have the SSS-based diffusion step vector $t = [t_1, t_1, ..., t_1]$. For a mini-batch sample, SSS ensures the distribution comparison is on the same diffusion step. Compared to uniform sampling, SSS has one obvious limitation: less coverage of diffusion steps. By increasing the number of training epochs, each diffusion step in $[0, T-1]$ will eventually be sampled.

Model Architecture

Our model in Figure 2 is based on transformer encoders including vanilla transformer (Vaswani et al. 2017) encoders

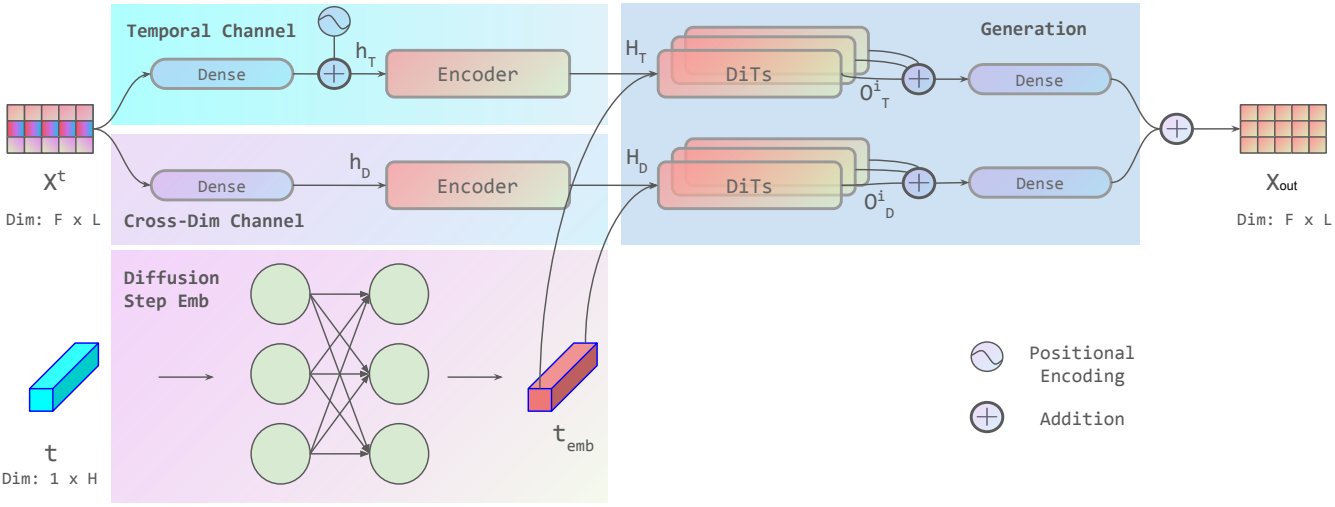

Figure 2: PaD-TS model architecture

and diffusion transformer (DiT) blocks (Peebles and Xie 2023). To fully capture temporal and cross-dimensional information, we propose a dual-channel architecture where each part of the information is processed separately. Each channel (temporal and cross-dimensional) passes a dense layer to encode channel representation, a vanilla transformer encoder, a few residual connected DiT blocks, and a dense layer revert to its original shape.

Temporal and cross-dimensional representation can be learned via a linear dense layer (Liu et al. 2024). Given a mini-batch TS $x \in \mathbb{R}^{b \times L \times F}$ and its corresponding diffusion steps t, where b stands for the number of samples in a mini-batch, L stands for sequence length, and F stands for the number of features. By permuting L and F separately, we obtain temporal first input $x_{\mathrm{T}} \in \mathbb{R}^{b \times L \times F}$ and feature first inputs $x_{\mathrm{D}} \in \mathbb{R}^{b \times F \times L}$. For the temporal first input, we have an additional learned positional embedding pos(x_T). This process can be formulated as follows:

$$h_T = (W_T x_T + b_T) + \mathrm{pos}(x_T) \tag{11}$$

$$h_D = W_D x_D + b_D \tag{12}$$

where W_T and W_D represent dense layer parameters; b_T and b_D represent bias terms; and has outputs $h_T \in \mathbb{R}^{b \times L \times H}$ and $h_D \in \mathbb{R}^{b \times F \times H}$.

Vanilla transformer encoder (Enc) is used to analyze TS at each diffusion step. The transformer encoder block is based on multi-head attention which is commonly used for pattern recognition and feature extraction (Peebles and Xie 2023; Liu et al. 2024; Yuan and Qiao 2024; Coletta et al. 2023; Tashiro et al. 2021). We use one transformer block for each channel to extract relative information:

$$H_T = \mathrm{Enc}(h_T) \tag{13}$$

$$H_D = \mathrm{Enc}(h_D) \tag{14}$$

DiT blocks with residual connections are the final layers in the model. Compared to the vanilla encoder blocks, Peebles and Xie (2023) design DiTs which perform well in the diffusion framework with two advantages: high throughput and the way conditional information is introduced. Transformers are naturally with high-throughput due to the multi-head attention mechanism that can be processed in parallel. Unlike many methods that add conditional information before each block, DiTs introduce partial conditional embedding at each layer. More details can be found in Appendix B. In our study, conditional embedding is the diffusion step embedding t_{emb} which is learned via dense layers. We use DiT blocks for the generation process which can be formally described as:

$$\begin{aligned} O^i = &\mathbf{1}_{i=0}\mathrm{DiT}(H, t_{emb}) + \mathbf{1}_{i=1}\mathrm{DiT}(O^0 + H, t_{emb}) \\ &+ \mathbf{1}_{i>0}\mathrm{DiT}(O^{i-1} + O^{i-2}, t_{emb}) \end{aligned} \tag{15}$$

where O^i is the i-th DiT block output, H is the encoded information from the previous section, t is the diffusion conditional embedding (i.e., diffusion step), and $\mathbf{1_c}$ is an indicator function with condition c. For the different channels, we simply replace H with H_D or H_T to obtain O_D^i or O_T^i.

The final output can be obtained by adding all DiT blocks output from cross-dimension and temporal modules with a dense layer that converts to its original shape:

$$x_{out} = (W_D^2 \sum_{i=0}^{N} O_D^i + b_D^2) + (W_T^2 \sum_{i=0}^{N} O_T^i + b_T^2) \tag{16}$$

5 Experiments

In this section, we describe our experiment settings and evaluate the TS generation quality of PaD-TS across different domains and sequence lengths. The experiment results consist of quantitive and qualitative results in terms of *individual* authenticity and *population*-level property preservation. We also perform an ablation study to demonstrate the effectiveness of each proposed component and the effect of the hyperparameter α.

Metrics	Dataset	PaD-TS	Diffusion-TS	TimeGAN	TimeVAE
VDS	Sines	**0.0005**	0.0007	0.0034	0.0177
	Stocks	**0.0029**	0.0369	0.0257	0.0038
	Energy	**0.0019**	0.0060	0.0427	0.0882
FDDS	Sines	**0.0003**	0.0031	0.0167	0.0135
	Stocks	**0.0588**	0.1841	0.1117	0.2161
	Energy	**0.0442**	0.1837	0.2777	0.4413
DA	Sines	0.013± 0.004	**0.005± 0.000**	0.037± 0.004	0.072± 0.061
	Stocks	**0.055± 0.087**	0.082± 0.025	0.143± 0.073	0.133± 0.115
	Energy	**0.078± 0.011**	0.127± 0.016	0.469± 0.017	0.498± 0.004
Predictive score	Sines	**0.093± 0.000**	**0.093± 0.000**	0.095± 0.000	0.229± 0.001
	Stocks	**0.037± 0.000**	**0.037± 0.000**	0.039± 0.000	0.038± 0.000
	Energy	0.251± 0.011	**0.250± 0.000**	0.338± 0.010	0.277± 0.001

Table 1: TS generation results with generation length 24 for Sines, Stocks, and Energy datasets. PaD-TS shows state-of-the-art performance in most cases. **Bold** font (lower score) indicates the best performance. Hyperparameters in Appendix C.

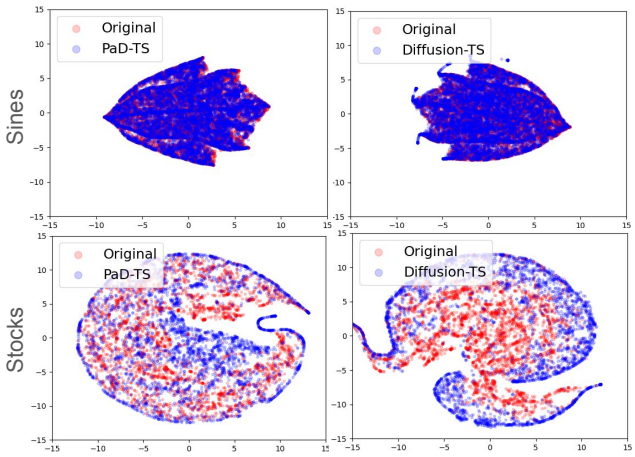

Figure 3: t-SNE plots on the cross-correlation values between original data (red dots) and synthetic data (blue dots) on the Sines and Stocks dataset.

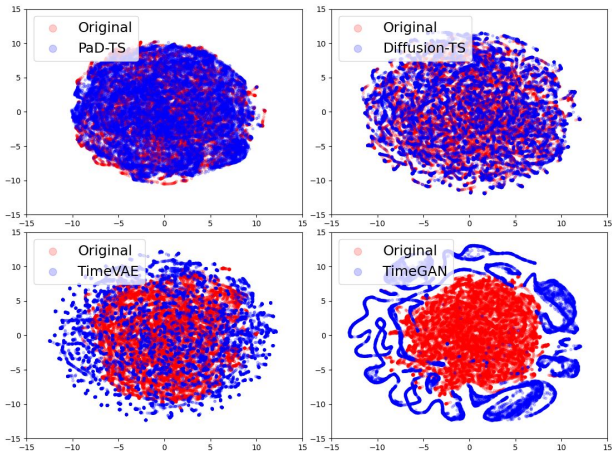

Figure 4: t-SNE plots on the cross-correlation values between original data (red dots) and synthetic data (blue dots) on the Energy dataset.

Experiment Settings

We briefly discuss the datasets, baseline models, and evaluation methods. All experiments are run on a Rocky Linux server with AMD EPYC 7313 CPU, 128 GB of memory, and 2 Nvidia A40 GPUs. Additional model hyperparameters are provided in Appendix C.

Datasets: We use three major benchmark datasets, spanning domains such as physics, finance, and synthetic time series. (1) *Sines* (Yoon, Jarrett, and Van der Schaar 2019): Synthetic sine wave time series data that can be sampled based on parameters. (2) *Stocks* (Yoon, Jarrett, and Van der Schaar 2019): Google stocks history time series data includes 5 features such as Open, Close, Volume, etc. (3) *Energy* (Candanedo, Feldheim, and Deramaix 2017): Home appliances' energy consumption time series data includes 28 features such as energy consumption, room temperatures, room humidity levels, and more. Additional Mujoco (Tunyasuvunakool et al. 2020) and fMRI (Smith et al. 2011) dataset results are available in Appendix E.

Baselines: We carefully select three previous models that perform well and cover all three generative frameworks: (1) Diffusion-TS (Yuan and Qiao 2024) is a DM with trends and Fourier-based layers. (2) TimeGAN (Yoon, Jarrett, and Van der Schaar 2019) is a GAN-based model with RNN layers. (3) TimeVAE (Desai et al. 2021) is a VAE-based model with convolution layers, trends blocks, and seasonal blocks.

Evaluation metrics: We use the following metrics to evaluate the TS generation quality: (1) VDS score, (2) FDDS score, (3) DA score, and (4) predictive score. The first three metrics are introduced in Section 3, where VDS and FDDS scores measure the population-level distribution shift of generated TS in terms of value and functional dependency. We use CC as an example of functional dependency. DA score (Yoon, Jarrett, and Van der Schaar 2019) measures the individual-level authenticity. In addition, we are interested in evaluating how the generated data can be used in down-

Metrics	Length	Pad-TS	Diffusion-TS	TimeGAN	TimeVAE
VDS Score	64	**0.0009**	0.0043	0.1688	0.0658
	128	**0.0005**	0.0046	0.1565	0.0544
	256	**0.0008**	0.0044	0.2725	0.0416
FDDS score	64	**0.0087**	0.0476	0.8540	0.2656
	128	**0.0009**	0.0112	0.9767	0.1120
	256	**0.0010**	0.0038	3.0019	0.0424
DA	64	$\mathbf{0.023 \pm 0.009}$	0.094 ± 0.009	0.437 ± 0.062	0.499 ± 0.000
	128	$\mathbf{0.050 \pm 0.080}$	0.165 ± 0.067	0.399 ± 0.268	0.499 ± 0.000
	256	$\mathbf{0.138 \pm 0.174}$	0.393 ± 0.009	0.499 ± 0.000	0.492 ± 0.001
Predictive Score	64	$\mathbf{0.248 \pm 0.000}$	0.249 ± 0.000	0.301 ± 0.007	0.290 ± 0.001
	128	$\mathbf{0.247 \pm 0.003}$	0.248 ± 0.001	0.316 ± 0.008	0.290 ± 0.000
	256	$\mathbf{0.244 \pm 0.001}$	0.250 ± 0.002	0.285 ± 0.006	0.266 ± 0.001

Table 2: Long TS Generation Results on Energy dataset. **Bold** font (lower score) indicates the best performance.

Metrics	PaD-TS	Diffusion-TS
MDD	0.609	**0.573**
ACD	**0.061**	0.200
SD	0.027	**0.025**
KD	**0.032**	0.049
ED	**0.645**	0.658
DTW	**1.674**	1.718

Table 3: Feature and distance-based measures comparison between Diffusion-TS and PaD-Ts on Sines dataset. Bold font (lower score) indicates the best performance.

Metrics	PaD-TS	Diffusion-TS
MDD	**0.379**	0.440
ACD	0.111	**0.028**
SD	**0.375**	0.471
KD	4.290	**2.207**
ED	1.135	**1.093**
DTW	2.937	**2.829**

Table 4: Feature and distance-based measures comparison between Diffusion-TS and PaD-Ts on Stocks dataset. Bold font (lower score) indicates the best performance.

stream tasks such as TS prediction. Hence, our last metric is the predictive score (Yoon, Jarrett, and Van der Schaar 2019), which is the mean absolute error score of the TS prediction result where the post-hoc RNN model is trained using synthetic TS data and evaluated on real TS data. Due to the unstable nature of the DA score and predictive score, we repeat the evaluation for 5 iterations and report the mean and standard deviation for robust results. We additionally include feature and distance-based metrics summarized in TSGBench (Ang et al. 2023): Marginal Distribution Difference (MDD), AutoCorrelation Difference (ACD), Skewness Difference (SD), Kurtosis Difference (KD), Euclidean Distance (ED), and Dynamic Time Warping (DTW).

Comparison with Baselines

In Table 1, we present the result for a benchmark setting that is commonly used in state-of-the-art TS generation models (Yuan and Qiao 2024; Yoon, Jarrett, and Van der Schaar 2019): TS generation with sequence length 24. The result shows that PaD-TS consistently outperforms previous methods in terms of population-level property preservation (i.e., VDS and FDDS). Averaging across all three datasets, PaD-TS improves the FDDS score by 5.9x and the VDS score by 5.7x compared to the previous state-of-the-art model, Diffusion-TS, while maintaining comparable performance in individual-level authenticity. In Table 3, Table 4 and Table 5, we present the feature and distance-based metrics results. PaD-TS achieved comparable or better performance across Sines, Stocks, and Energy datasets.

To better understand the performance of population-level preservation in different models, we visualize the t-distributed stochastic neighbor embedding (t-SNE) (Van der Maaten and Hinton 2008) on the functional dependency cross-correlation values obtained from the Sines, Stocks, and Energy datasets and their corresponding synthetic data in a lower-dimensional space. As we can see in Figure 3 and 4, the t-SNE plot produced by PaD-TS shows the best alignment with that of the original data (red dots). This is consistent with Table 1, where the FDDS score of PaD-TS is significantly lower than the baselines. In addition to t-SNE, we include more visualizations of the global value distribution in the Appendix D. The results indicate that PaD-TS is also the most effective in preserving value distributions.

Long sequence generation. We further challenge PaD-TS in TS generation with longer sequence lengths (64, 128, and 256) on the high-dimensional Energy dataset. The results in Table 2 show that PaD-TS has a dominating performance compared to baselines in all 4 metrics. PaD-TS not only improved the FDDS score by 6.1x on average but also made a significant improvement in the DA score by 3.4x on average.

Time Complexity Comparison. PaD-TS requires slightly longer training time compared to existing DMs but remains reasonable. Two primary factors contribute to the

Metrics	PaD-TS	Diffusion-TS
MDD	0.221	**0.200**
ACD	**0.055**	0.141
SD	**0.124**	0.174
KD	**1.037**	1.387
ED	**1.030**	1.032
DTW	**6.395**	6.439

Table 5: Feature and distance-based measures comparison between Diffusion-TS and PaD-Ts on Energy dataset. Bold font (lower score) indicates the best performance.

Dataset	PaD-TS	Diffusion-TS
Sines	77min	17min
Stocks	75min	15min
Energy	117min	60min

Table 6: Training time comparison between PaD-TS and Diffusion-TS.

extended training time: 1) the additional loss term, L_{pop}, which computes the pairwise distribution distance, and 2) the SSS, which necessitates additional iterations relative to the standard sampling strategy. In table 6, we present the training time required between Diffusion-TS and PaD-TS for selected datasets.

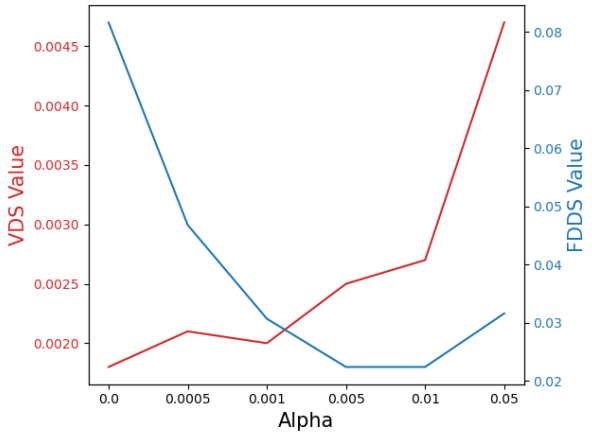

Figure 5: Ablation study on α and Energy dataset. The blue and red curves resp. depict the FDDS and VDS scores.

Ablation Study

To further understand our model, we conduct two ablation studies to evaluate: (1) the effectiveness of each model component in terms of population-level property (i.e., CC) preservation, and (2) the effect of the population aware training objective hyperparameter α.

(1) In the first ablation study, we train PaD-TS variants by taking out one of the four components of the full PaD-TS as follows: (1) without (w/o) temporal channel, (2) w/o dimension channel, (3) w/o PAT objective, and (4) w/o SSS strategy. In Table 7, results indicate that the SSS strategy and temporal channel are the most useful components, while the PAT training objective and the dimension channel are less effective but still crucial. By combining all four components, the full PaD-TS variant shows the best performance.

Metrics	Model	Sines	Stocks	Energy
FDDS	PaD-TS	**0.0003**	**0.0588**	**0.0442**
	w/o Temporal	0.0005	1.8838	0.5254
	w/o Dimension	0.0087	0.0868	0.0533
	w/o PAT	0.0007	0.2459	0.0816
	w/o SSS	0.0286	0.0965	0.3626

Table 7: Ablation study for the effectiveness of PaD-TS components. **Bold** font indicates the best performance.

(2) In the second ablation study, we train different PaD-TS models with different α values ranging from 0 to 0.05. Intuitively, a larger α indicates more weight to the CC distribution loss L_{pop} and less weight to the original loss L_0. In Figure 5, results show that when α increases, there is a general trend of increasing (worse) VDS score and decreasing (better) FDDS score. Once α goes too large ($\alpha = 0.05$), the entire training collapses with large VDS and FDDS scores.

6 Conclusion

We study the TS generation problem with a focus on the preservation of TS population-level property. Towards this goal, our core contribution is PaD-TS, a novel DM that is equipped with a new population-aware training process, and a new dual-channel encoder model architecture. Our extensive experimental results show that PaD-TS achieves state-of-the-art performance both qualitatively and quantitatively in all three benchmark datasets over the two population-level authenticity metrics. Our ablation study also shows the effectiveness of each new component in PaD-TS. In the future, we would like to further enhance PaD-TS with the ability to do conditional generation (e.g., constrained by certain trend information), and apply PaD-TS to downstream TS-related tasks in low-resource domains, especially where generation bias could lead to critical issues.

Acknowledgements

This work was supported in part by Generated Health and The William & Mary Applied Research & Innovation Initiative Exploratory Award.

References

Alcaraz, J. M. L.; and Strodthoff, N. 2022. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*.

Ang, Y.; Huang, Q.; Bao, Y.; Tung, A. K.; and Huang, Z. 2023. Tsgbench: Time series generation benchmark. *arXiv preprint arXiv:2309.03755*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, 214–223. PMLR.

Barancikova, B.; Huang, Z.; and Salvi, C. 2024. SigDiffusions: Score-Based Diffusion Models for Long Time Series via Log-Signature Embeddings. *arXiv preprint arXiv:2406.10354*.

Brooks, T.; Peebles, B.; Holmes, C.; DePue, W.; Guo, Y.; Jing, L.; Schnurr, D.; Taylor, J.; Luhman, T.; Luhman, E.; Ng, C.; Wang, R.; and Ramesh, A. 2024. Video generation models as world simulators.

Candanedo, L. M.; Feldheim, V.; and Deramaix, D. 2017. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and buildings*, 140: 81–97.

Cao, Y.; Long, M.; and Wang, J. 2018. Unsupervised domain adaptation with distribution matching machines. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Chen, H.; Jajodia, S.; Liu, J.; Park, N.; Sokolov, V.; and Subrahmanian, V. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *IJCAI*, 2074–2080.

Coletta, A.; Gopalakrishnan, S.; Borrajo, D.; and Vyetrenko, S. 2023. On the constrained time-series generation problem. *Advances in Neural Information Processing Systems*, 36.

Deb, C.; Zhang, F.; Yang, J.; Lee, S. E.; and Shah, K. W. 2017. A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74: 902–924.

Desai, A.; Freeman, C.; Wang, Z.; and Beaver, I. 2021. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*.

Esteban, C.; Hyland, S. L.; and Rätsch, G. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*.

Feng, S.; Miao, C.; Zhang, Z.; and Zhao, P. 2024. Latent diffusion transformer for probabilistic time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 11979–11987.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1): 723–773.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Kaushik, S.; Choudhury, A.; Sheron, P. K.; Dasgupta, N.; Natarajan, S.; Pickett, L. A.; and Dutt, V. 2020. AI in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3: 4.

Kingma, D. P.; and Welling, M. 2022. Auto-Encoding Variational Bayes. arXiv:1312.6114.

Li, C.-L.; Chang, W.-C.; Cheng, Y.; Yang, Y.; and Póczos, B. 2017. Mmd gan: Towards deeper understanding of moment matching network. *Advances in neural information processing systems*, 30.

Li, Y.; Lu, X.; Wang, Y.; and Dou, D. 2022. Generative time series forecasting with diffusion, denoise, and disentanglement. *Advances in Neural Information Processing Systems*, 35: 23009–23022.

Liao, S.; Ni, H.; Szpruch, L.; Wiese, M.; Sabate-Vidales, M.; and Xiao, B. 2020. Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421*.

Liu, Q.; and Wang, D. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29.

Liu, Y.; Hu, T.; Zhang, H.; Wu, H.; Wang, S.; Ma, L.; and Long, M. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*.

Loshchilov, I.; and Hutter, F. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Masini, R. P.; Medeiros, M. C.; and Mendes, E. F. 2023. Machine learning advances for time series forecasting. *Journal of economic surveys*, 37(1): 76–111.

Mogren, O. 2016. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. *arXiv preprint arXiv:1611.09904*.

Morid, M. A.; Sheng, O. R. L.; and Dunbar, J. 2023. Time series prediction using deep learning methods in healthcare. *ACM Transactions on Management Information Systems*, 14(1): 1–29.

Naiman, I.; Erichson, N. B.; Ren, P.; Mahoney, M. W.; and Azencot, O. 2023. Generative modeling of regular and irregular time series data via koopman VAEs. *arXiv preprint arXiv:2310.02619*.

Nichol, A.; Dhariwal, P.; Ramesh, A.; Shyam, P.; Mishkin, P.; McGrew, B.; Sutskever, I.; and Chen, M. 2021. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

Nichol, A. Q.; and Dhariwal, P. 2021. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, 8162–8171. PMLR.

Peebles, W.; and Xie, S. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4195–4205.

Pei, H.; Ren, K.; Yang, Y.; Liu, C.; Qin, T.; and Li, D. 2021. Towards generating real-world time series data. In *2021 IEEE International Conference on Data Mining (ICDM)*, 469–478. IEEE.

Priesmann, J.; Nolting, L.; Kockel, C.; and Praktiknjo, A. 2021. Time series of useful energy consumption patterns for energy system modeling. *Scientific Data*, 8(1): 148.

Qian, J.; Sun, M.; Zhou, S.; Wan, B.; Li, M.; and Chiang, P. 2024. TimeLDM: Latent Diffusion Model for Unconditional Time Series Generation. arXiv:2407.04211.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2): 3.

Rasul, K.; Seward, C.; Schuster, I.; and Vollgraf, R. 2021. Autoregressive denoising diffusion models for multivariate

probabilistic time series forecasting. In *International Conference on Machine Learning*, 8857–8868. PMLR.

Smith, S. M.; Miller, K. L.; Salimi-Khorshidi, G.; Webster, M.; Beckmann, C. F.; Nichols, T. E.; Ramsey, J. D.; and Woolrich, M. W. 2011. Network modelling methods for FMRI. *Neuroimage*, 54(2): 875–891.

Sohl-Dickstein, J.; Weiss, E.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, 2256–2265. PMLR.

Song, Y.; Sohl-Dickstein, J.; Kingma, D. P.; Kumar, A.; Ermon, S.; and Poole, B. 2020. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Tashiro, Y.; Song, J.; Song, Y.; and Ermon, S. 2021. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816.

Tunyasuvunakool, S.; Muldal, A.; Doron, Y.; Liu, S.; Bohez, S.; Merel, J.; Erez, T.; Lillicrap, T.; Heess, N.; and Tassa, Y. 2020. dm_control: Software and tasks for continuous control. *Software Impacts*, 6: 100022.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yan, H.; Ding, Y.; Li, P.; Wang, Q.; Xu, Y.; and Zuo, W. 2017. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2272–2281.

Yang, J.; Dai, T.; Li, N.; Wu, J.; Liu, P.; Li, J.; Bao, J.; Zhang, H.; and Xia, S. 2024. Generative Pre-Trained Diffusion Paradigm for Zero-Shot Time Series Forecasting. *arXiv preprint arXiv:2406.02212*.

Yoon, J.; Jarrett, D.; and Van der Schaar, M. 2019. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32.

Yuan, X.; and Qiao, Y. 2024. Diffusion-TS: Interpretable Diffusion for General Time Series Generation. In *The Twelfth International Conference on Learning Representations*.

Zeng, Z.; Kaur, R.; Siddagangappa, S.; Rahimi, S.; Balch, T.; and Veloso, M. 2023. Financial time series forecasting using CNN and transformer. *arXiv preprint arXiv:2304.04912*.

Zhao, S.; Song, J.; and Ermon, S. 2017. Infovae: Information maximizing variational autoencoders. *arXiv preprint arXiv:1706.02262*.

A Cross-Correlation

In this section, we briefly review the definition of CC. CC and its normalized form Pearson correlation are popular similarity measurements in TS and signal processing tasks (Liao et al. 2020; Yuan and Qiao 2024). Given two univariate TS X_t and Y_t where t stands for time, the general CC function $R_{X_tY_t}$ can be formulated as:

$$R_{X_tY_t}(\tau) = \mathbb{E}[X_{t-\tau}Y_t] \tag{17}$$

where τ stands for optional lags. Throughout the experiment, we set $\tau = 0$ and apply it for later derivations.

By further normalizing X_t and Y_t with their means μ_X and μ_Y, we will obtain cross-covariance K_{XY} between them:

$$K_{X_tY_t} = \mathbb{E}[(X_t - \mu_X)(Y_t - \mu_Y)] \tag{18}$$

Finally, we can normalize K_{XY} with variance measures σ_X^2 and σ_Y^2 to obtain the Pearson correlation coefficient ρ, which is more interpretable and ranges from [-1,1] with the following:

$$\begin{aligned} \rho_{X_tY_t} &= \frac{\mathbb{E}[(X_t - \mu_X)(Y_t - \mu_Y)]}{\sqrt{\sigma_X^2}\sqrt{\sigma_Y^2}} \\ &= \frac{\mathbb{E}[X_tY_t] - \mathbb{E}[X_t]\mathbb{E}[Y_t]}{\sqrt{\mathbb{E}[X_t^2] - (\mathbb{E}[X_t])^2}\sqrt{\mathbb{E}[Y_t^2] - (\mathbb{E}[Y_t])^2}} \end{aligned} \tag{19}$$

B Diffusion Transformer (DiT) Blocks

In this section, we briefly review the architecture of DiT blocks (Peebles and Xie 2023). DiT blocks contain more parameters and offer high throughput during training. The experiments demonstrate promising generation quality and strong scalability. As illustrated in Figure 6, the DiT block architecture closely resembles that of a standard transformer encoder. For the conditional input c, DiT divides the hidden state into 6 chunks and gradually introduces them using an adaLN-Zero design. Condition injection layers 1 and 3 incorporate two chunks of the conditional hidden features each, while layers 2 and 4 incorporate one chunk each. Unlike vanilla transformer encoder-based models, DiT can generate samples based on conditional information.

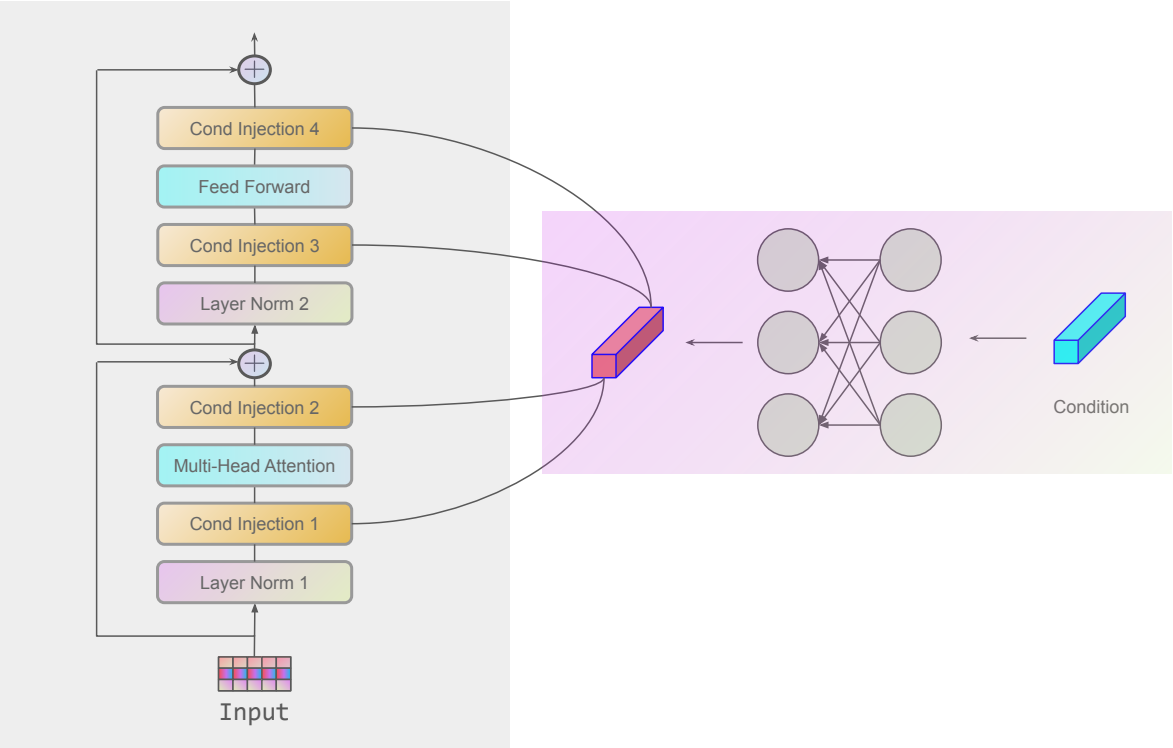

Figure 6: DiT block architecture.

C Additional Experiment Details

In this section, we discuss the code implementation details and the hyperparameters we explored throughout the experiment.

Our code is based on the improved DDPM (Nichol and Dhariwal 2021) as it is more formally implemented with additional features such as different sampling strategies, models towards different targets (x^0, x^t and ϵ^t), etc. For the model architecture, we adopted some implementation from diffusion transformer (Peebles and Xie 2023) and inverted transformer (Liu et al. 2024). Finally, we modify code from TimeGAN (Yoon, Jarrett, and Van der Schaar 2019) and Diffusion-TS (Yuan and Qiao 2024) for data pre-processing and evaluation.

The hyperparameters of a DM usually come from two sources: (1) the general DM pipeline and (2) the model architecture.

(1) For the general DM pipeline, we use a cosine noise scheduler and a model toward input without noise x^0 throughout our experiment. We additionally tuned the following hyperparameters: diffusion steps from 100 to 700, batch sizes from 32 to 128, and normalization strategies (min-max or -1 to 1).

(2) For the model architecture, we use an AdamW optimizer (Loshchilov and Hutter 2017) (with learning rate $= 0.0001$), the proposed PaD-TS architecture, and 4 attention heads in all transformer-related blocks. We additionally tuned our model hyperparameters: α from 0 to 0.05, hidden dimension from 32 to 256, number of encoders from 1 to 2, and number of DiT blocks from 2 to 4. We found the following set of best-working hyperparameters listed in Table 8:

Parameter	Sines	Stocks	Energy
Target	x^0	x^0	x^0
Noise Scheduler	Cosine	Cosine	Cosine
Diffusion Step	250	250	500
Batch Size	64	64	64
Normalization	-1 to 1	-1 to 1	-1 to 1
Optimizer	AdamW	AdamW	AdamW
Num of Heads	4	4	4
α	0.0005	0.0008	0.0005
Hidden Dim	128	128	256
Num of Enc	1	1	1
Num of DiTs	3	3	3

Table 8: List of DM and model-related parameters with generation length 24 for Sines, Stocks, and Energy dataset. DM parameters are listed in the first half, and model-related parameters are listed in the second half of the Table.

D Additional Figures

This section provides additional figures that compare the value distributions between the original and synthetic data. Figure 4 displays a t-SNE plot of the CC values for both the original and synthetic data on the Energy dataset, highlighting the performance in preserving the CC distribution. Following Diffusion-TS (Yuan and Qiao 2024), we use t-SNE and data distribution plots to compare how well different methods maintain the value distribution. As shown in Figure 7, the results indicate that PaD-TS is more closely aligned with the original data set (red dots).

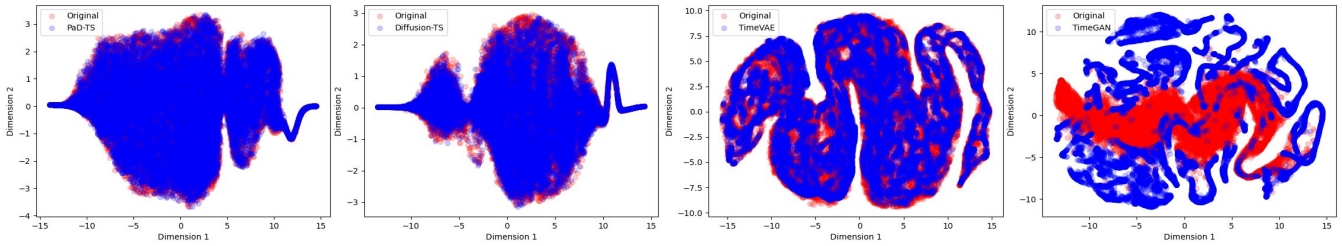

Figure 7: t-SNE plots on the average values for each dimension between original data (red dots) and synthetic data (blue dots) on the Energy dataset.

For the value distribution plot Figure 8, we also see the PaD-TS best aligns with the original dataset (red line), which is consistent with results in Figure 7 and Table 1.

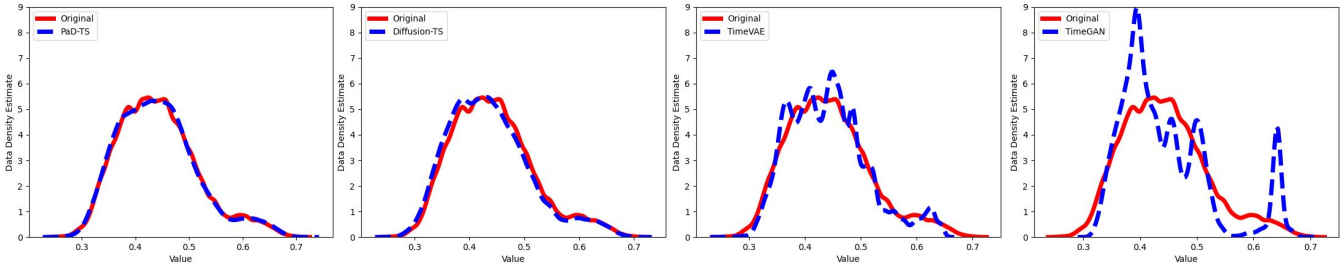

Figure 8: Value distribution plots on the average values for each dimension between original data (red line) and synthetic data (blue line) on the Energy dataset.

E Additional Experiment

This section compares the PaD-TS performance with Diffusion-TS on the Mujoco and the fMRI datasets. In Table 9, PaD-TS achieves performance on par with Diff-TS in terms of discriminative accuracy and predictive score, along with improvements in FDDS and VDS.

Metrics	**Dataset**	**PaD-TS**	**Diffusion-TS**
VDS	fMRI	**0.0008**	0.0010
	Mujoco	**0.0009**	0.0014
FDDS	fMRI	**0.0034**	0.0046
	Mujoco	**0.0092**	0.0164
DA	fMRI	**0.153**$\pm$ **0.032**	0.164$\pm$ 0.015
	Mujoco	**0.016** $\pm$ **0.005**	0.018$\pm$ 0.009
Predictive score	fMRI	**0.100**$\pm$ **0.000**	**0.100**$\pm$ **0.000**
	Mujoco	**0.008**$\pm$ **0.002**	**0.007**$\pm$ **0.001**

Table 9: TS generation results with generation length 24 for Mujoco and fMRI datasets. **Bold** font (lower score) indicates the best performance. Hyperparameters are listed in the code repo.