TimeDP: Learning to Generate Multi-Domain Time Series with Domain Prompts

Yu-Hao Huang[1], **Chang Xu**[2*], **Yueying Wu**[3], **Wu-Jun Li**[1], **Jiang Bian**[2]
[1]National Key Laboratory for Novel Software Technology, School of Computer Science, Nanjing University
[2]Microsoft Research Asia
[3]Peking University
huangyh@smail.nju.edu.cn, {chanx, jiang.bian}@microsoft.com
wuyueying@stu.pku.edu.cn, liwujun@nju.edu.cn

Abstract

Time series generation models are crucial for applications like data augmentation and privacy preservation. Most existing time series generation models are typically designed to generate data from one specified domain. While leveraging data from other domain for better generalization is proved to work in other application areas, this approach remains challenging for time series modeling due to the large divergence in patterns among different real world time series categories. In this paper, we propose a multi-domain time series diffusion model with domain prompts, named **TimeDP**. In TimeDP, we utilize a time series semantic prototype module which defines time series prototypes to represent time series basis, each prototype vector serving as "word" representing some elementary time series feature. A prototype assignment module is applied to extract the extract domain specific prototype weights, for learning domain prompts as generation condition. During sampling, we extract "domain prompt" with few-shot samples from the target domain and use the domain prompts as condition to generate time series samples. Experiments demonstrate that our method outperforms baselines to provide the state-of-the-art in-domain generation quality and strong unseen domain generation capability.

Extended version — https://arxiv.org/abs/2501.05403

Introduction

In the landscape of large models and advanced machine learning techniques, time series foundation models (Das et al. 2024; Gao et al. 2024) have garnered increasing attention. These models, typically trained on extensive datasets spanning various domains (Woo et al. 2024), have predominantly emphasized forecasting tasks rather than the generation of new data. However, the accurate and meaningful generation of time series is critical for applications such as medical record synthetic (Li, Yu, and Príncipe 2023) and financial scenario simulations (Coletta et al. 2021; Huang et al. 2024), as well as for augmenting datasets where historical records are limited or incomplete (Kollovieh et al. 2023a).

Although some research has been conducted on time series generation, most efforts have been confined to the development of generation model for single-domain data. In contrast, cross-domain time series generation presents a significantly more complex challenge, as it requires the creation of new data across various domains without relying on existing historical records. This stands out as a gap, underscoring a substantial opportunity for further advancements in multi-domain time series generation.

One straightforward approach to multi-domain time series generation involves the use of predefined domain labels during the training process (Lee, Malacarne, and Aune 2023). This method relies on the availability of domain labels to formulate the conditional generation process. However, this approach may struggle generalizing to large number of domains or unseen domains. Moreover, the challenge intensifies when domain labels are not explicitly available.

An alternative approach frames cross-domain time series generation as a conditional generation task by describing the domain using natural language (Jin et al. 2024; Jiang et al. 2024). However, the use of natural language descriptions introduces significant challenges. Domain-specific nuances are often difficult to articulate precisely, leading to noisy, incomplete, or ambiguous prompts. Moreover, for entirely new or evolving domains, crafting these domain descriptors can be impractical. This has underscored a critical need for a more systematic and robust way to represent and utilize domain-specific information in time series generation.

To address these challenges, we propose a label-free, text-free method that learns time series prototypes as basic elements to construct domain prompts for generating time series with a diffusion model, named **TimeDP**. Through training, the prototypes learn to represent time series basis, serving as "word" with time series semantics. A prototype assignment module is applied for each training samples to construct the specific "prompt" for generating this sample. During sampling, we extract "prompt" with few-shot samples from the target domain to construct the population of domain prompts and use the domain prompts as condition to generate time series samples.

To summarize, the main contributions of this paper are listed as follows:

- We propose **TimeDP**, a multi-domain time series generation model by learning a set of time series prototypes and prototype assignment module to construct domain prompts, where the domain prompts serve as condition for a time series diffusion model.

*Corresponding author.

- We are the first to propose a multi-domain time series generation model using label-free, text-free conditioning mechanism.
- Experiments demonstrate that our method outperforms baselines with the state-of-the-art in-domain generation quality, and strong unseen domain generation capability.

Related Work and Backgrounds

Time Series Generation

Existing time series generation models has based on various foundational type of generative models. GAN-based methods has been introduced to encourage the network to consider temporal dynamic by jointly optimize both supervised and adversarial objectives for a learned embedding space (Yoon, Jarrett, and van der Schaar 2019). VAE-based methods have designed specific decoder structure for temporal data considering trend and seasonal decomposition (Desai et al. 2021), and first introduces vector quantization technique together with bidirectional transformers to better capture temporal consistency (Lee, Malacarne, and Aune 2023). Another category is considered as mixed-type methods, combining GANs, flows and ODEs (Jeon et al. 2022). Different from these methods, we utilize denoising diffusion probabilistic models (DDPM) as our generation backbone.

Existing diffusion-based time series generation methods leverage both unconditional and conditional diffusion models for generating time series data with various denoising network backbones (Kollovieh et al. 2023b; Yang et al. 2024). Researchers have also considered combining diffusion models with the constrained generation problem (Coletta et al. 2023) and the extraction of time series intrinsic such as seasonal-trend decomposition techniques (Yuan and Qiao 2024). Compared with these single-domain methods, we first propose to utilize label-free, text-free domain prompts as condition for generating time series.

Cross-Domain Time Series Model

There have been several recent work consider utilizing multiple-domain time series data for training time series foundation models. These works can be divided into two branches. The first branch builds two-stage models. Kraus et al. (2024) pretrains a representation learning model on 75 datasets for the first stage and finetuning to task specific models at the second stage. Gao et al. (2024) conducts masked reconstruction pretraining on 38 multi-domain datasets for the first stage and a multi-task supervised learning for downstream tasks. The second branch pretrains end-to-end transformer models with patch tokenizers for time series forecasting. Woo et al. (2024) pretrains on a dataset with over 2B observations and Das et al. (2024) pretrains on a dataset with 100B time-points, both using patching and instance normalizing to unify across different data scale, frequencies and lengths. These methods employ instance normalization to generate forecasts based on historical data without explicitly addressing domain differences. Compared with these approaches, we propose to use time series prototypes, constructing domain prompts to explicitly distinguish domains as well as bridge them.

Denoising Diffusion Probabilistic Models (DDPMs)

A diffusion probabilistic model (Sohl-Dickstein et al. 2015) learns to reverse the transitions of a Markov chain which is known as the diffusion process that gradually adds noise to data, ultimately destroying the signal.

Let $\mathbf{x}_0 \in \mathbb{R}^d \sim q(\mathbf{x}_0)$ be real data of dimension d from space $\mathcal{X}$. The diffusion process generates $\mathbf{x}_1, ..., \mathbf{x}_N$ from the same space with the same shape as $\mathbf{x}_0$, using a Markov chain that adds Gaussian noise over N time steps: $q(\mathbf{x}_1, ..., \mathbf{x}_N|\mathbf{x}_0) := \prod_{n=1}^{N} q(\mathbf{x}_n|\mathbf{x}_{n-1})$. The transition kernel is commonly defined as:

$$q(\mathbf{x}_n|\mathbf{x}_{n-1}) := \mathcal{N}(\mathbf{x}_n; \sqrt{1-\beta_n}\mathbf{x}_n, \beta_n\mathbf{I}), \quad (1)$$

where $\{\beta_n \in (0,1)\}_{n=1,...,N}$ defines the variance schedule. Note that $\mathbf{x}_n$ at any arbitrary time step n can be derived in a closed form $q(\mathbf{x}_n|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_n; \sqrt{\bar{\alpha}_n}\mathbf{x}_0, (1-\bar{\alpha}_n)\mathbf{I})$, where $\alpha_n := 1 - \beta_n$ and $\bar{\alpha}_n := \prod_{s=1}^{n} \alpha_s$. For the reverse process, the diffusion model, parameterized by θ, yields:

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N) := p(\mathbf{x}_N) \prod_{n=1}^{N} p_\theta(\mathbf{x_{n-1}}|\mathbf{x_n}), \quad (2)$$

where $p_\theta(\mathbf{x}_{n-1}|\mathbf{x}_n) := \mathcal{N}(\mathbf{x}_{n-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_n, n), \boldsymbol{\Sigma}_\theta(\mathbf{x}_n, n))$ and the transitions start at $p(\mathbf{x}_N) = \mathcal{N}(\mathbf{x}_n; \mathbf{0}, \mathbf{I})$. The optimization objective is derived into the maximizing of an approximation of the evidence lower bound (ELBO) of the log-likelihood $\log p_\theta(\mathbf{x}_0)$. With a widely adopted parameterization:

$$\boldsymbol{\mu}_\theta(\mathbf{x}_n, n) = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}_n - \frac{\beta_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n)), \quad (3)$$

the training is performed to predict the noise term added in the forward process which simplifies the objective to:

$$L_{simple} := \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\bar{\alpha}_n}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}, n)\|]. \quad (4)$$

On sampling, $\mathbf{x}_{n-1} = \frac{1}{\sqrt{\alpha_n}}(\mathbf{x}_n - \frac{1-\alpha_n}{\sqrt{1-\bar{\alpha}_n}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_n, n)) + \sigma_n\mathbf{z}$, where $\sigma_n = \sqrt{\beta_n}$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ (Ho, Jain, and Abbeel 2020).

Typical time series diffusion models for forecasting task (Rasul et al. 2021; Shen and Kwok 2023; Fan et al. 2024) encodes history context into c as condition and make use of the conditional form of DDPMs (Ho and Salimans 2021) for generating future time series:

$$p_\theta(\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_N|c) := p(\mathbf{x}_N) \prod_{n=1}^{N} p_\theta(\mathbf{x_{n-1}}|\mathbf{x_n}, c), \quad (5)$$

In the problem setting of time series generation, the generation process does not rely on time series history. We explore the use of term c to provide domain semantics for time series generation model in this work.

Problem Formulation

Let $D_i^T = \{\boldsymbol{x} \in \mathbb{R}^T\}^{N_i}, \boldsymbol{x} = (x_1, x_2, ..., x_T)$ denote a time series dataset domain i with N_i time series samples, where each sample contains T sequential values. A straightforward single-domain time series generation model fits the

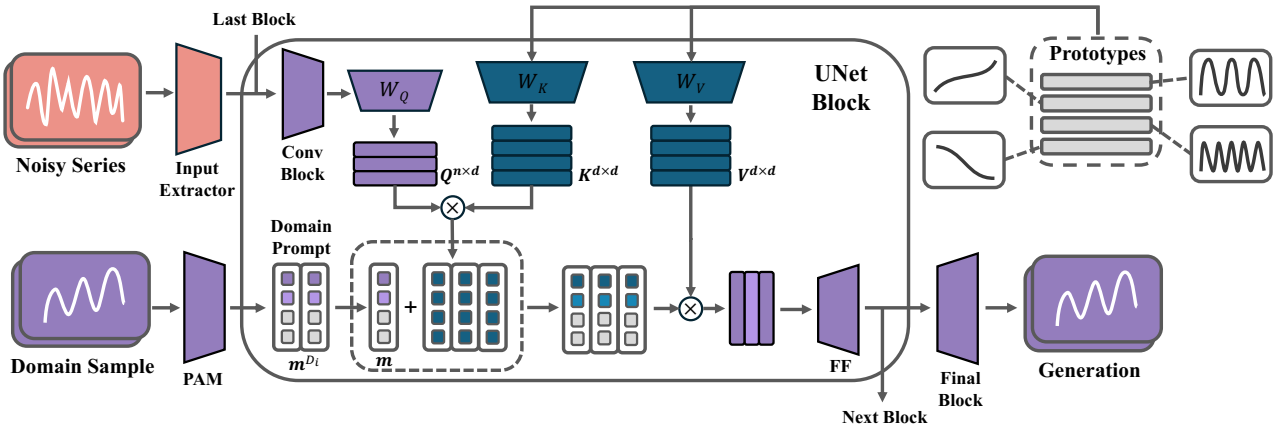

Figure 1: Overview of TimeDP model.

joint distribution over time steps $p(x_1, x_2, ..., x_T)$ of each dataset with a separate model parameterized by θ_i, namely $p_{\theta_i}(x_1, x_2, ..., x_T)$ for all $\boldsymbol{x}$ in D_i^T.

In this paper, we explore a domain-unified setting where the mixture of M domain datasets with sequence length T is denoted by $D^T = \bigcup_{i=1}^{M} D_i$ and we aim to build one model for the mixed dataset parameterized by θ, namely $p_\theta(x_1, x_2, ..., x_T|i)$ for all $\boldsymbol{x}$ in D^T.

Adhering to the channel-independent setting (Nie et al. 2023) that is widely accepted by recent researches, we formulate the problem studied in this paper in a uni-variate time series generation manner to handle the heterogeneity of time series in terms of dimension (Woo et al. 2024).

Methodology

With sequences from all data domains mixed together during training, all time series features within latent representation are entangled without a explicit way for distinguishing specific time series data domain. Although utilizing domain labels as class labels for training the time series generation model can provide instruction for identifying specific domain, this approach implies an assumption that all domains are independent from each other, neglecting the different similarity levels among domain pairs. Therefore, it is challenging to equip the model with the ability of generating time series in selected domain while considering the inter-relationship among domains. To overcome this challenge, the key is to build a triggering mechanism for cross-domain time series model that can control the model for generating time series data from specific domain. Motivated by the recent advancements in controllable content generation with prompting technique, we propose to construct domain prompts for controlling a cross-domain model.

In the rest of this section, we first describe the model architecture design. Then, we describe the optimization objective and training algorithm of the proposed model. Finally, we discuss the procedure for in-domain sampling and unseen domain generation using the proposed model.

Domain Prompts

Different from text and image modality where the generation target can be expressed by natural language or categorized into discrete classes, it is difficult to obtain explicit representation of time series with words or class labels. Inspired by the widely adopted technique to extract "basis" which are the elementary features of time series (Ni et al. 2023), these basis can be utilized as the shared "dictionary" among different domains, each of which encodes different semantic feature for time series.

Semantic Prototype Module Each basis represents certain elementary time series feature like trend and seasonality, that may exist in time series data samples. Different individual time series samples are assumed to share the same collection of basis but reflect distinct subset of the collection. As a result, each time series gets unique realization of these underlying features, similar to variable weighted allocations to all the basis. Based on this assumption, a set of latent arrays is introduced as time series prototypes $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{N_p \times d}$ for representing cross-domain time series common knowledge, where each prototype vector $\boldsymbol{p} \in \mathbb{R}^{1\times d}$ serves as the representation of a time series basis. In practice, the time series prototypes $\boldsymbol{P}$ are initialized with random orthogonal vectors and are frozen afterwards.

Prototype Assignment Module (PAM) Given the assumption that each time series sample corresponds to a distinct allocation of all the basis, the mapping from time series samples to the allocations needs to be established for explicitly identifying important prototypes for each time series instance as well as distinguishing among domains. We propose to extract a prototype assignment for each time series instance as the importance weights of each time series on each prototype, and the prototype assignments then serve as conditions for the generation model.

Specifically, each input sequence $\boldsymbol{x}$ is mapped into a weight vector whose dimension equals to the number of prototypes using weight extractor ϕ, which is a neural network. The vector $\phi(\boldsymbol{x})$ represents the weight of each vector inside $\boldsymbol{P}$, and the weights are utilized to modify the atten-

tion weight within the cross-attention mechanism so that the predicted noises are only conditioned on the assigned prototypes. Therefore, sequences from different domains are represented by different $\boldsymbol{m}$ weighted combinations of the shared same set of time series prototypes. For ensuring sparsity on prototype assignments, all negative weights are discarded when conducting prototype assignment. Formally, the prototype assignments $\boldsymbol{m}$ is extracted with the following formula:

$$\boldsymbol{m} = \phi(\boldsymbol{x}_0) - \mathbf{I}_{\phi(\boldsymbol{x}_0)<=0} \cdot \infty, \tag{6}$$

where $\mathbf{I}_{\cdot<0}$ is the indicator function of negative elements.

Domain-Unified Training

Instead of training individual model for each specific dataset, we train one model with data from multiple datasets at the same time for generating different domain data. Here, we treat each dataset as a separate domain. While data from each domain only represent limited fraction of possible data distribution, leveraging data from other domain can help model capture a more diverse time series data distribution.

Other than taking the unconditional denoising diffusion objective stated in eq. (4), we employ the conditional denoising objective using c from eq. (5) as condition to make use of the encoded semantic context for denoising process, where the conditions are incorporated into the intermediate layers of noise prediction network by spatial attention.

$$\boldsymbol{Q}^{(i)} = \boldsymbol{z}^{(i-1)} \cdot \boldsymbol{W}_Q^{(i)}, \tag{7}$$

$$\boldsymbol{K}^{(i)} = \boldsymbol{P} \cdot \boldsymbol{W}_K^{(i)}, \quad \boldsymbol{V}^{(i)} = \boldsymbol{P} \cdot \boldsymbol{W}_V^{(i)}, \tag{8}$$

$$\boldsymbol{z}^{(i)} = \mathrm{FF}(\mathrm{softmax}(\frac{\boldsymbol{Q}^{(i)}\boldsymbol{K}^{(i)T}}{\sqrt{d}} + \boldsymbol{m}) \cdot \boldsymbol{V}^{(i)}), \tag{9}$$

where $\boldsymbol{z}^{(i)} \in \mathbb{R}^{N \times d}$ denotes the output of the i^{th} last U-Net block. $\boldsymbol{W}_Q^{(i)} \in \mathbb{R}^{d \times d}$, $\boldsymbol{W}_K^{(i)} \in \mathbb{R}^{d \times d}$ and $\boldsymbol{W}_V^{(i)} \in \mathbb{R}^{d \times d}$ are learnable projection matrices applied on the sequence dimension. FF denotes feed forward layer. The attention output $\boldsymbol{z}^{\text{final}}$ is followed by another feed forward network to produce final block output $\hat{\epsilon} = \mathrm{FF}(\boldsymbol{z}^{\text{final}})$.

With the conditional denoising mechanism described above, the denoising objective using ϵ-parameterization can be written and simplified into:

$$\begin{aligned} L_{cond} &= \mathbb{E}[\|\boldsymbol{\epsilon} - \hat{\boldsymbol{\epsilon}}\|_1] \\ &= \mathbb{E}_{\boldsymbol{x}_0 \in D^T, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), n}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\theta, \boldsymbol{P}}(x_n, n, \boldsymbol{m})\|_1]. \end{aligned} \tag{10}$$

Due to the imbalance number of training samples across domain, we adopt a re-weight sampling method for making the probability equal for training on samples from each domain. Let N_i denote the number of sample sequences in dataset i, we set the weight for sampling each sample of this dataset as $w_i = \frac{1}{N_i * |D|}$, such that the probability for sampling sequence from each dataset is balanced. The pseudo code for training algorithm is shown at Algorithm 1.

Generation with Domain Prompt

To generate time series samples of selected domain after the domain-unified training on multiple datasets, we first extract domain-specific prototype assignments of a small random subset of training samples for the selected domain and group them into a distribution of domain prompt representing the selected domain. Let K denotes the number of selected samples from dataset i, the domain prompt is denoted by $\boldsymbol{m}^{D_i} = \{\boldsymbol{m}_1^i, ..., \boldsymbol{m}_K^i\}$. By constructing the conditioning input, the model generates samples adhere to the selected domain while is not constrained by the general temporal patterns exhibited in the selected samples. When the number of expected generated samples is larger than K, we use a strategy of repeatedly generate with each assignment in K samples until the number of expected samples is satisfied.

The sampling algorithm is described as Algorithm 2.

Algorithm 1: Training algorithm

Require: Sequence sample $\boldsymbol{x}$
Ensure: Network parameters ϕ and θ, prototypes $\boldsymbol{P}$
1: Initialize prototypes $\boldsymbol{P}$
2: **repeat**
3: Sample $\boldsymbol{x}_0$ from D^T
4: Extract prototype assignments $\boldsymbol{m}$ according to eq. (6)
5: Randomly set $\boldsymbol{P}$ as unconditional identifier $\boldsymbol{p}_u$
6: Randomly sample time step $n \sim \mathcal{U}(1, N)$
7: Randomly sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
8: Corrupt data $\boldsymbol{x}_n = \sqrt{\bar{\alpha}_n}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_n}\boldsymbol{\epsilon}$
9: Predict step noise with $\tilde{\boldsymbol{\epsilon}} = \tilde{\boldsymbol{\epsilon}}_{\theta, \boldsymbol{P}}(\boldsymbol{x}_n, n, \boldsymbol{m})$
10: Compute loss with Equation (10) and take gradient step.
11: **until** maximum training step

Algorithm 2: Sampling with domain prompts

Require: K time series prompts $\boldsymbol{x}$, prototypes $\boldsymbol{P}$
Ensure: Generated time series samples $\hat{\boldsymbol{x}}$
1: Extract prototype prompts $\boldsymbol{m}$ with $\boldsymbol{x}$ according to eq. (6)
2: Randomly sample noise $\hat{\boldsymbol{x}}_N \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
3: **for** n from N to 1 **do**
4: Predict step noise with $\tilde{\boldsymbol{\epsilon}}_n = \tilde{\boldsymbol{\epsilon}}_{\theta, \boldsymbol{P}}(\hat{\boldsymbol{x}}_n, n, \boldsymbol{m})$
5: Denoise $\hat{\boldsymbol{x}}_{n-1} = \frac{\hat{\boldsymbol{x}}_n - \sqrt{1-\bar{\alpha}_n}\tilde{\boldsymbol{\epsilon}}_n}{\sqrt{\bar{\alpha}_n}}$
6: **end for**
7: $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}_0$

Unseen Domain Generation

Since the prototypes provide representations for time series basis, their representation ability is not restricted to the domains in training sets. Therefore, they can be utilized to represent unseen domain or datasets. For any unseen dataset or domain D_j with respect to the training set, we can use extract "few-shot" samples $\boldsymbol{x}_1^j, ..., \boldsymbol{x}_K^j$ from the dataset to construct domain prompt $\boldsymbol{m}^{D_j} = \{\boldsymbol{m}_1^j, ..., \boldsymbol{m}_K^j\}$, and then feed them into the model as condition to generate new samples of the required dataset.

Experiments

In this section, we provide empirical experiment results for our method using multiple real-world datasets. The experiment goal is to investigate on the following research questions: (a) How good is the quality of prompted generation on trained domain? (b) Can the learned prototype help the model generalize to unseen domains?

		TimeDP	TimeGAN	GT-GAN	TimeVAE	TimeVQVAE	TimeVQVAE-C
Maximum Mean Discrepancy	Electricity	**0.001**$_{\pm 0.001}$	0.367$_{\pm 0.255}$	0.254$_{\pm 0.166}$	0.577$_{\pm 0.006}$	0.152$_{\pm 0.024}$	0.002$_{\pm 0.001}$
	Solar	**0.041**$_{\pm 0.011}$	0.628$_{\pm 0.053}$	0.578$_{\pm 0.039}$	0.353$_{\pm 0.014}$	0.437$_{\pm 0.020}$	0.058$_{\pm 0.005}$
	Wind	0.025$_{\pm 0.017}$	0.213$_{\pm 0.017}$	0.170$_{\pm 0.040}$	0.170$_{\pm 0.004}$	0.131$_{\pm 0.014}$	**0.018**$_{\pm 0.007}$
	Traffic	**0.083**$_{\pm 0.034}$	0.567$_{\pm 0.057}$	0.538$_{\pm 0.078}$	0.218$_{\pm 0.007}$	0.213$_{\pm 0.016}$	0.089$_{\pm 0.001}$
	Taxi	**0.095**$_{\pm 0.023}$	0.275$_{\pm 0.054}$	0.319$_{\pm 0.032}$	0.139$_{\pm 0.007}$	0.128$_{\pm 0.004}$	0.109$_{\pm 0.014}$
	Pedestrian	**0.044**$_{\pm 0.020}$	0.090$_{\pm 0.030}$	0.112$_{\pm 0.019}$	0.065$_{\pm 0.002}$	0.067$_{\pm 0.007}$	0.058$_{\pm 0.002}$
	Air	**0.011**$_{\pm 0.003}$	0.120$_{\pm 0.045}$	0.211$_{\pm 0.041}$	0.089$_{\pm 0.016}$	0.028$_{\pm 0.002}$	0.041$_{\pm 0.008}$
	Temperature	**0.219**$_{\pm 0.022}$	0.926$_{\pm 0.042}$	0.809$_{\pm 0.081}$	1.002$_{\pm 0.014}$	0.323$_{\pm 0.008}$	0.259$_{\pm 0.043}$
	Rain	**0.057**$_{\pm 0.039}$	0.329$_{\pm 0.285}$	0.111$_{\pm 0.109}$	0.292$_{\pm 0.019}$	0.074$_{\pm 0.007}$	0.080$_{\pm 0.004}$
	NN5	**0.164**$_{\pm 0.010}$	0.874$_{\pm 0.088}$	0.632$_{\pm 0.074}$	0.821$_{\pm 0.061}$	0.327$_{\pm 0.012}$	0.243$_{\pm 0.041}$
	Fred-MD	**0.002**$_{\pm 0.001}$	0.043$_{\pm 0.021}$	0.133$_{\pm 0.102}$	0.059$_{\pm 0.008}$	0.008$_{\pm 0.002}$	0.005$_{\pm 0.002}$
	Exchange	**0.151**$_{\pm 0.024}$	0.530$_{\pm 0.154}$	0.475$_{\pm 0.116}$	0.543$_{\pm 0.149}$	0.342$_{\pm 0.050}$	0.233$_{\pm 0.107}$
K-L	Electricity	**0.012**$_{\pm 0.016}$	0.488$_{\pm 0.175}$	0.407$_{\pm 0.079}$	0.734$_{\pm 0.023}$	0.280$_{\pm 0.051}$	0.027$_{\pm 0.015}$
	Solar	**0.016**$_{\pm 0.005}$	0.612$_{\pm 0.447}$	0.120$_{\pm 0.041}$	0.260$_{\pm 0.016}$	0.865$_{\pm 0.108}$	0.234$_{\pm 0.062}$
	Wind	0.152$_{\pm 0.034}$	1.924$_{\pm 1.233}$	**0.107**$_{\pm 0.016}$	0.484$_{\pm 0.015}$	0.483$_{\pm 0.066}$	0.183$_{\pm 0.047}$
	Traffic	**0.009**$_{\pm 0.003}$	1.305$_{\pm 0.320}$	1.409$_{\pm 0.251}$	0.211$_{\pm 0.014}$	0.178$_{\pm 0.026}$	0.016$_{\pm 0.003}$
	Taxi	**0.011**$_{\pm 0.004}$	0.650$_{\pm 0.180}$	0.950$_{\pm 0.197}$	0.110$_{\pm 0.020}$	0.110$_{\pm 0.026}$	0.038$_{\pm 0.010}$
	Pedestrian	**0.014**$_{\pm 0.010}$	0.417$_{\pm 0.181}$	0.411$_{\pm 0.096}$	0.065$_{\pm 0.005}$	0.405$_{\pm 0.051}$	0.039$_{\pm 0.008}$
	Air	**0.027**$_{\pm 0.016}$	0.348$_{\pm 0.093}$	0.578$_{\pm 0.049}$	0.164$_{\pm 0.012}$	0.054$_{\pm 0.012}$	0.093$_{\pm 0.025}$
	Temperature	**0.171**$_{\pm 0.073}$	8.892$_{\pm 2.681}$	3.174$_{\pm 2.685}$	2.183$_{\pm 0.110}$	0.735$_{\pm 0.066}$	0.379$_{\pm 0.110}$
	Rain	**0.013**$_{\pm 0.012}$	0.506$_{\pm 0.174}$	0.432$_{\pm 0.099}$	0.160$_{\pm 0.022}$	0.047$_{\pm 0.018}$	0.065$_{\pm 0.018}$
	NN5	**0.054**$_{\pm 0.014}$	4.928$_{\pm 4.112}$	1.386$_{\pm 0.520}$	1.337$_{\pm 0.220}$	1.063$_{\pm 0.274}$	0.220$_{\pm 0.151}$
	Fred-MD	**0.203**$_{\pm 0.035}$	0.512$_{\pm 0.290}$	0.380$_{\pm 0.070}$	0.346$_{\pm 0.041}$	0.831$_{\pm 0.077}$	1.118$_{\pm 0.276}$
	Exchange	**1.866**$_{\pm 0.132}$	8.861$_{\pm 3.397}$	7.201$_{\pm 4.380}$	10.404$_{\pm 1.434}$	5.052$_{\pm 1.385}$	8.475$_{\pm 3.056}$

Table 1: Maximum mean discrepancy (MMD) and K-L divergence (K-L) of in-domain generation for sequence length 168. Best results are highlighted in bold face and second best results are underlined.

Experiment Settings

Datasets The experiments are conducted on 12 datasets across four time series domains: Electricity, Solar and Wind from the energy domain; Traffic, Taxi and Pedestrian from the transport domain, Air Quality, Temperature and Rain from the nature domain; NN5, Fred-MD and Exchange from the economic domain. All datasets are obtained by GluonTS package and Monash Time Series Forecasting Repository. We pre-process all datasets into non-overlapping uni-variate sequence slices with length in $\{24, 96, 168, 336\}$. More details on datasets are described in the technical appendix.

Baselines We compare our generation results with several representative state-of-the-art time series generation methods, including TimeGAN (Yoon, Jarrett, and van der Schaar 2019), GT-GAN (Jeon et al. 2022), TimeVAE (Desai et al. 2021), and TimeVQVAE (Lee, Malacarne, and Aune 2023). Note that in their original implementation, these methods are trained with single dataset from a single domain for each time. To align their implementations with our problem setting and method, we train these models with the multi-domain dataset which mix all of the twelve datasets together. Note that the TimeVQVAE baseline has a class-conditioned variant, which we considered as a conditional time series generation baseline that is able to sample by domain label after being trained with the multi-domain dataset.

Implementation Details We follow common practice of diffusion models to utilize a U-Net architecture for our denoising model. The architecture details are discussed in the technical appendix. The number of prototypes are set to 16 for all the main evaluations. Models for each sequence length are trained for $50,000$ steps using a batch size of 128 and a learning rate of 1×10^{-4} with $1,000$ warm-up steps. For all baselines, we take the implementations and recommended hyper-parameter settings from their public codes.

Evaluation Metrics The major consideration for evaluating performance of time series generation methods are two fold: (1) the similarity between real and synthetic time series distributions; (2) the internal temporal dependency within each instance. In light of these two we apply the following three metrics for evaluating generation performance: (1) **Maximum Mean Discrepancy (MMD)** which maps the data points into a high-dimensional feature space with a kernel function and then compare the means of the two data distribution; (2) **Kullback-Leibler divergence (K-L)** which is the measure of how one probability distribution diverges from a second, reference probability distribution, indicating the distribution difference on variable level; (3) **Marginal distribution difference (MDD)** which calculates the empirical histogram for each time step and calculate the average absolute difference of real and synthetic data across bins.

Results

Evaluation of Generation Quality In this experiment, we evaluate the time series generation performance of our model on all 12 datasets across four domains, comparing

			Stock			Web		
		# Shots	3	10	100	3	10	100
Maximum Mean Discrepancy	Unconditional	TimeGAN	$0.517_{\pm 0.030}$	$0.823_{\pm 0.042}$	$0.944_{\pm 0.075}$	$0.197_{\pm 0.027}$	$0.089_{\pm 0.045}$	$0.126_{\pm 0.104}$
		GT-GAN	$0.466_{\pm 0.050}$	$0.790_{\pm 0.048}$	$0.692_{\pm 0.074}$	$0.109_{\pm 0.027}$	$0.117_{\pm 0.012}$	$0.112_{\pm 0.014}$
		TimeVAE	$0.351_{\pm 0.009}$	$0.870_{\pm 0.022}$	$0.898_{\pm 0.025}$	$0.203_{\pm 0.004}$	$0.146_{\pm 0.006}$	$0.099_{\pm 0.006}$
		TimeVQVAE	$0.105_{\pm 0.005}$	$0.111_{\pm 0.001}$	$0.096_{\pm 0.004}$	$0.078_{\pm 0.007}$	$\mathbf{0.063}_{\pm 0.009}$	$0.062_{\pm 0.006}$
	Fine-tuned	TimeGAN	$0.124_{\pm 0.081}$	$0.144_{\pm 0.034}$	$0.136_{\pm 0.093}$	$0.157_{\pm 0.034}$	$0.182_{\pm 0.054}$	$0.055_{\pm 0.018}$
		GT-GAN	$0.139_{\pm 0.076}$	$0.161_{\pm 0.113}$	$0.261_{\pm 0.132}$	$0.101_{\pm 0.053}$	$0.191_{\pm 0.037}$	$0.230_{\pm 0.068}$
		TimeVAE	$0.348_{\pm 0.007}$	$0.790_{\pm 0.028}$	$0.761_{\pm 0.032}$	$0.199_{\pm 0.013}$	$0.160_{\pm 0.045}$	$0.118_{\pm 0.019}$
		TimeVQVAE	$0.205_{\pm 0.089}$	$0.176_{\pm 0.095}$	$0.127_{\pm 0.050}$	$0.278_{\pm 0.226}$	$0.198_{\pm 0.077}$	$0.085_{\pm 0.060}$
	Prompted	TimeDP	$\mathbf{0.048}_{\pm 0.014}$	$\mathbf{0.050}_{\pm 0.012}$	$\mathbf{0.014}_{\pm 0.005}$	$\mathbf{0.068}_{\pm 0.035}$	$0.065_{\pm 0.014}$	$\mathbf{0.021}_{\pm 0.004}$
K-L	Unconditional	TimeGAN	$5.077_{\pm 1.314}$	$5.213_{\pm 1.211}$	$3.717_{\pm 1.563}$	$1.485_{\pm 0.766}$	$0.432_{\pm 0.375}$	$0.485_{\pm 0.340}$
		GT-GAN	$2.010_{\pm 0.302}$	$2.086_{\pm 0.294}$	$1.541_{\pm 0.120}$	$0.645_{\pm 0.123}$	$1.351_{\pm 0.291}$	$1.626_{\pm 0.274}$
		TimeVAE	$1.872_{\pm 0.094}$	$2.175_{\pm 0.106}$	$1.666_{\pm 0.075}$	$0.345_{\pm 0.021}$	$\mathbf{0.056}_{\pm 0.007}$	$0.104_{\pm 0.007}$
		TimeVQVAE	$\mathbf{0.840}_{\pm 0.117}$	$\mathbf{0.458}_{\pm 0.084}$	$0.692_{\pm 0.125}$	$\mathbf{0.175}_{\pm 0.035}$	$0.448_{\pm 0.034}$	$0.579_{\pm 0.045}$
	Fine-tuned	TimeGAN	$3.164_{\pm 0.361}$	$1.844_{\pm 0.684}$	$1.304_{\pm 0.450}$	$1.739_{\pm 0.662}$	$0.606_{\pm 0.119}$	$0.357_{\pm 0.140}$
		GT-GAN	$1.183_{\pm 0.266}$	$0.482_{\pm 0.253}$	$0.681_{\pm 0.292}$	$0.588_{\pm 0.313}$	$1.230_{\pm 0.248}$	$1.640_{\pm 0.553}$
		TimeVAE	$1.331_{\pm 0.269}$	$1.855_{\pm 0.170}$	$1.344_{\pm 0.080}$	$0.309_{\pm 0.050}$	$0.058_{\pm 0.010}$	$0.482_{\pm 0.097}$
		TimeVQVAE	$1.139_{\pm 0.416}$	$0.614_{\pm 0.303}$	$0.864_{\pm 0.688}$	$1.176_{\pm 1.338}$	$0.540_{\pm 0.318}$	$0.383_{\pm 0.295}$
	Prompted	TimeDP	$1.613_{\pm 0.065}$	$0.721_{\pm 0.041}$	$\mathbf{0.090}_{\pm 0.037}$	$0.245_{\pm 0.027}$	$0.254_{\pm 0.030}$	$\mathbf{0.091}_{\pm 0.010}$
MDD	Unconditional	TimeGAN	$0.837_{\pm 0.014}$	$0.803_{\pm 0.007}$	$0.790_{\pm 0.015}$	$0.084_{\pm 0.008}$	$0.054_{\pm 0.013}$	$0.062_{\pm 0.022}$
		GT-GAN	$0.795_{\pm 0.007}$	$0.804_{\pm 0.002}$	$0.756_{\pm 0.006}$	$0.050_{\pm 0.005}$	$0.064_{\pm 0.002}$	$0.059_{\pm 0.002}$
		TimeVAE	$0.814_{\pm 0.003}$	$0.858_{\pm 0.003}$	$0.859_{\pm 0.003}$	$0.076_{\pm 0.002}$	$0.057_{\pm 0.002}$	$0.045_{\pm 0.001}$
		TimeVQVAE	$\mathbf{0.711}_{\pm 0.005}$	$0.702_{\pm 0.003}$	$0.706_{\pm 0.006}$	$0.047_{\pm 0.005}$	$0.072_{\pm 0.003}$	$0.073_{\pm 0.002}$
	Fine-tuned	TimeGAN	$0.750_{\pm 0.031}$	$0.770_{\pm 0.046}$	$0.760_{\pm 0.038}$	$0.066_{\pm 0.022}$	$0.079_{\pm 0.003}$	$0.062_{\pm 0.012}$
		GT-GAN	$0.717_{\pm 0.026}$	$0.711_{\pm 0.032}$	$0.731_{\pm 0.018}$	$0.048_{\pm 0.010}$	$0.072_{\pm 0.010}$	$0.080_{\pm 0.010}$
		TimeVAE	$0.801_{\pm 0.012}$	$0.847_{\pm 0.005}$	$0.840_{\pm 0.006}$	$0.077_{\pm 0.002}$	$0.056_{\pm 0.004}$	$0.071_{\pm 0.006}$
		TimeVQVAE	$0.741_{\pm 0.022}$	$0.718_{\pm 0.029}$	$0.708_{\pm 0.024}$	$0.076_{\pm 0.046}$	$0.087_{\pm 0.015}$	$0.062_{\pm 0.025}$
	Prompted	TimeDP	$0.720_{\pm 0.012}$	$\mathbf{0.690}_{\pm 0.009}$	$\mathbf{0.653}_{\pm 0.002}$	$\mathbf{0.042}_{\pm 0.006}$	$\mathbf{0.051}_{\pm 0.005}$	$\mathbf{0.038}_{\pm 0.002}$

Table 2: Maximum mean discrepancy (MMD), K-L divergence (K-L) and Marginal distribution difference (MDD) in unseen domain settings of Stock and Web datasets for generation sequence length 168. Best results are highlighted in bold face.

with the baselines on MMD, K-L and MDD. The proposed model and baseline models are evaluated after trained with the multi-domain dataset. While the baselines are not designed for learning time series from multiple domains simultaneously, we modify them to align with our setting that treating the multi-domain dataset as a whole. Additionally, while TimeVQVAE can be trained in a class-conditional manner, we also test on this variant, using the source of data domain as class label to obtain a cross-domain version, as shown in column "TimeVQVAE-C". Each run is repeated 5 times with different random seeds. The average and the standard deviation of MMD and K-L results for the sequence length of 168 are shown in Table 1. The MDD results and other results for different sequence length settings are reported in the appendix.

As shown in Table 1, our approach achieves the best results on most of the twelve in-domain datasets, demonstrating superior performance on generating new time series samples that have the closest distribution to real dataset samples, both jointly and marginally. Most second best scores are obtained with the class-conditional version of TimeVQVAE model. The results indicates that, simply mixing datasets as the multi-domain training approach generally fails to generate samples that are close to real data in distribution, majorly due to the diverse intrinsic pattern among different domains. Although using dataset labels helps greatly on the training with mixed data as shown with TimeVQVAE-C, our model outperforms it without being explicitly supervised with class labels, indicating strong representation disentanglement capability. The results for other sequence length settings are included in the technical appendix. The results show that TimeDP consistently demonstrate the best generation performance on both long and short sequence length scenarios.

Evaluation of Unseen Domain Time Series Synthesis In this experiment, we evaluate the synthesis performance on unseen domain, where datasets that are not included in the training data. We randomly select a few samples from the new datasets as demonstrations for all models, and ensure a test set that does not overlap with these samples. For TimeDP, these samples are used to extract domain prompts. For baseline methods that are not designed to generate following prompts or can not generate with unseen class labels,

		TimeDP					- PAM	- Prompt
	# Prototypes	4	8	16	32	64		
Maximum Mean Discrepancy	Electricity	$0.004_{\pm 0.002}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	**0.001**$_{\pm 0.001}$	$0.004_{\pm 0.002}$	$0.016_{\pm 0.015}$
	Solar	$0.061_{\pm 0.025}$	$0.041_{\pm 0.018}$	$0.041_{\pm 0.011}$	$0.037_{\pm 0.007}$	**0.037**$_{\pm 0.006}$	$0.545_{\pm 0.030}$	$0.520_{\pm 0.029}$
	Wind	$0.029_{\pm 0.017}$	$0.027_{\pm 0.017}$	$0.025_{\pm 0.017}$	$0.025_{\pm 0.017}$	**0.025**$_{\pm 0.016}$	$0.134_{\pm 0.028}$	$0.074_{\pm 0.047}$
	Traffic	$0.101_{\pm 0.025}$	$0.077_{\pm 0.018}$	$0.083_{\pm 0.034}$	$0.081_{\pm 0.023}$	**0.067**$_{\pm 0.005}$	$0.270_{\pm 0.015}$	$0.288_{\pm 0.020}$
	Taxi	$0.111_{\pm 0.021}$	$0.095_{\pm 0.013}$	$0.095_{\pm 0.023}$	$0.095_{\pm 0.020}$	**0.093**$_{\pm 0.016}$	$0.149_{\pm 0.026}$	$0.140_{\pm 0.014}$
	Pedestrian	$0.049_{\pm 0.021}$	**0.040**$_{\pm 0.010}$	$0.044_{\pm 0.020}$	$0.044_{\pm 0.016}$	$0.041_{\pm 0.016}$	$0.052_{\pm 0.009}$	$0.057_{\pm 0.007}$
	Air	$0.016_{\pm 0.003}$	$0.011_{\pm 0.001}$	$0.011_{\pm 0.003}$	$0.012_{\pm 0.003}$	**0.011**$_{\pm 0.002}$	$0.061_{\pm 0.031}$	$0.017_{\pm 0.007}$
	Temperature	$0.221_{\pm 0.034}$	$0.209_{\pm 0.014}$	$0.219_{\pm 0.022}$	$0.221_{\pm 0.023}$	$0.224_{\pm 0.027}$	$0.354_{\pm 0.011}$	**0.179**$_{\pm 0.043}$
	Rain	$0.056_{\pm 0.020}$	$0.047_{\pm 0.020}$	$0.057_{\pm 0.039}$	$0.051_{\pm 0.030}$	$0.057_{\pm 0.057}$	$0.056_{\pm 0.046}$	**0.043**$_{\pm 0.018}$
	NN5	$0.198_{\pm 0.059}$	$0.157_{\pm 0.003}$	$0.164_{\pm 0.010}$	**0.155**$_{\pm 0.008}$	$0.158_{\pm 0.005}$	$0.348_{\pm 0.021}$	$0.257_{\pm 0.021}$
	Fred-MD	**0.001**$_{\pm 0.000}$	$0.002_{\pm 0.001}$	$0.002_{\pm 0.001}$	$0.002_{\pm 0.001}$	$0.002_{\pm 0.001}$	$0.338_{\pm 0.114}$	$0.005_{\pm 0.005}$
	Exchange	$0.146_{\pm 0.031}$	$0.146_{\pm 0.022}$	$0.151_{\pm 0.024}$	$0.150_{\pm 0.024}$	$0.152_{\pm 0.023}$	$1.059_{\pm 0.008}$	**0.117**$_{\pm 0.028}$
	Average	0.083	**0.071**	0.074	0.073	0.072	0.281	0.143
K-L	Electricity	$0.003_{\pm 0.002}$	$0.011_{\pm 0.014}$	$0.012_{\pm 0.016}$	$0.014_{\pm 0.019}$	$0.012_{\pm 0.018}$	**0.011**$_{\pm 0.009}$	$0.013_{\pm 0.013}$
	Solar	$0.028_{\pm 0.021}$	$0.016_{\pm 0.007}$	$0.016_{\pm 0.005}$	**0.013**$_{\pm 0.002}$	$0.014_{\pm 0.005}$	$0.019_{\pm 0.004}$	$0.042_{\pm 0.036}$
	Wind	$0.151_{\pm 0.027}$	$0.153_{\pm 0.044}$	$0.152_{\pm 0.034}$	$0.156_{\pm 0.037}$	$0.155_{\pm 0.032}$	**0.072**$_{\pm 0.010}$	$0.156_{\pm 0.073}$
	Traffic	$0.024_{\pm 0.030}$	**0.007**$_{\pm 0.003}$	$0.009_{\pm 0.003}$	$0.012_{\pm 0.009}$	$0.011_{\pm 0.004}$	$0.014_{\pm 0.013}$	$0.047_{\pm 0.027}$
	Taxi	$0.051_{\pm 0.033}$	$0.008_{\pm 0.003}$	$0.011_{\pm 0.004}$	$0.012_{\pm 0.010}$	$0.009_{\pm 0.005}$	**0.003**$_{\pm 0.001}$	$0.032_{\pm 0.022}$
	Pedestrian	$0.031_{\pm 0.028}$	**0.010**$_{\pm 0.008}$	$0.014_{\pm 0.010}$	$0.014_{\pm 0.009}$	$0.012_{\pm 0.008}$	$0.021_{\pm 0.010}$	$0.041_{\pm 0.047}$
	Air	$0.047_{\pm 0.027}$	$0.024_{\pm 0.008}$	$0.027_{\pm 0.016}$	$0.027_{\pm 0.010}$	$0.025_{\pm 0.012}$	**0.020**$_{\pm 0.005}$	$0.036_{\pm 0.017}$
	Temperature	$0.304_{\pm 0.079}$	$0.176_{\pm 0.061}$	$0.171_{\pm 0.073}$	$0.179_{\pm 0.087}$	$0.166_{\pm 0.074}$	**0.132**$_{\pm 0.009}$	$0.168_{\pm 0.096}$
	Rain	$0.033_{\pm 0.009}$	$0.008_{\pm 0.004}$	$0.013_{\pm 0.012}$	**0.008**$_{\pm 0.005}$	$0.013_{\pm 0.008}$	$0.009_{\pm 0.008}$	$0.009_{\pm 0.004}$
	NN5	$0.217_{\pm 0.223}$	$0.050_{\pm 0.021}$	$0.054_{\pm 0.014}$	**0.045**$_{\pm 0.007}$	$0.046_{\pm 0.006}$	$0.049_{\pm 0.016}$	$0.073_{\pm 0.031}$
	Fred-MD	**0.188**$_{\pm 0.012}$	$0.190_{\pm 0.013}$	$0.203_{\pm 0.035}$	$0.216_{\pm 0.052}$	$0.201_{\pm 0.024}$	$0.196_{\pm 0.008}$	$0.196_{\pm 0.016}$
	Exchange	$1.863_{\pm 0.223}$	$1.828_{\pm 0.131}$	$1.866_{\pm 0.132}$	$2.313_{\pm 1.007}$	$1.875_{\pm 0.235}$	$1.623_{\pm 0.146}$	**1.506**$_{\pm 0.086}$
	Average	0.245	0.207	0.212	0.251	0.212	**0.181**	0.193

Table 3: Maximum mean discrepancy (MMD) and K-L divergence (K-L) results for ablation study on generation sequence length 168. Best results are highlighted in bold face. Second best results are underlined.

we evaluate their unconditional generation outputs. The results for these baseline models are labeled with "unconditional" at the table. Additionally, we conduct a fine-tuning on all baseline models to evaluate their few-shot learning setting. The fine-tuned model are labeled with "Fine-tuned" at the table. We use $3, 10$ and 100 samples as "prompt" and fine-tuning data in this experiment. The MMD and K-L results of our model and baselines for the generation sequence length 168 are shown in Table 2.

Table 2 shows that our model demonstrates robust zero-shot time series synthesis capability, obtaining the best general MMD and K-L scores compared to baselines. The fine-tuned models do not show consistent performance improvement against the unconditional model, indicating that in the low data regime, fine-tuning is not effective for fitting the data distribution. On the contrary, our model is able to infer the unseen domain distribution using domain prompts without additional tuning, and the performance improves as the number of few-shot samples grows, showing strong unseen domain generation capability.

Ablation Study In this section, we evaluate the sensitivity of PAM design by modifying the number of prototypes, and evaluate its effectiveness by removing it and using $\phi(\boldsymbol{x})$ as domain prompt instead, which is shown with "-PAM". We also test the unconditional version of model, labeled with "-Prompt". Table 3 shows the results on MMD and K-L scores, and MDD scores are shown in the appendix. Our model's performance remains consistent when the number is large enough. Removing PAM and removing conditioning mechanism both lead to great drop in MMD score while maintaining stable K-L scores, indicating that diffusion model backbone is strong enough to capture marginal distribution while PAM and domain prompts are essential in capturing sequence-wise time series distribution.

Conclusion

In this paper, we propose a multi-domain time series diffusion model with domain prompts, named **TimeDP**. TimeDP learns prototypes to represent time series basis, serving as "word" with time series semantics. A prototype assignment module is applied to extract the domain specific prototype weights, for learning domain prompts as generation condition. During sampling, we construct "domain prompt" with few-shot samples from the target domain and use the domain prompts as condition to generate time series samples. Experiments demonstrate that our method outperforms baselines to provide the state-of-the-art in-domain generation quality and strong unseen domain generation capability.

Acknowledgments

This work is supported by NSFC Project (No.62192783, No.12326615)

References

Coletta, A.; Gopalakrishnan, S.; Borrajo, D.; and Vyetrenko, S. 2023. On the Constrained Time-Series Generation Problem. In *Advances in Neural Information Processing Systems*, 61048–61059.

Coletta, A.; Prata, M.; Conti, M.; Mercanti, E.; Bartolini, N.; Moulin, A.; Vyetrenko, S.; and Balch, T. 2021. Towards realistic market simulations: a generative adversarial networks approach. In *Proceedings of the ACM International Conference on AI in Finance*, 1–9.

Das, A.; Kong, W.; Sen, R.; and Zhou, Y. 2024. A decoder-only foundation model for time-series forecasting. In *Proceedings of the International Conference on Machine Learning*, 10148–10167.

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.

Fan, X.; Wu, Y.; Xu, C.; Huang, Y.; Liu, W.; and Bian, J. 2024. MG-TSD: Multi-Granularity Time Series Diffusion Models with Guided Learning Process. In *International Conference on Learning Representations*.

Gao, S.; Koker, T.; Queen, O.; Hartvigsen, T.; Tsiligkaridis, T.; and Zitnik, M. 2024. UniTS: Building a Unified Time Series Model. *arXiv preprint*, arXiv:2403.00131.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 6840–6851.

Ho, J.; and Salimans, T. 2021. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

Huang, Y.-H.; Xu, C.; Liu, Y.; Liu, W.; Li, W.-J.; and Bian, J. 2024. Controllable Financial Market Generation with Diffusion Guided Meta Agent. *arXiv preprint*, arXiv:2408.12991.

Jeon, J.; KIM, J.; Song, H.; Cho, S.; and Park, N. 2022. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 36999–37010.

Jiang, Y.; Pan, Z.; Zhang, X.; Garg, S.; Schneider, A.; Nevmyvaka, Y.; and Song, D. 2024. Empowering Time Series Analysis with Large Language Models: A Survey. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 8095–8103.

Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. 2024. Time-LLM: Time Series Forecasting by Reprogramming Large Language Models. In *International Conference on Learning Representations*.

Kollovieh, M.; Ansari, A. F.; Bohlke-Schneider, M.; Zschiegner, J.; Wang, H.; and Wang, Y. 2023a. Predict, refine, synthesize: self-guiding diffusion models for probabilistic time series forecasting. In *Advances in Neural Information Processing Systems*, 28341–28364.

Kollovieh, M.; Ansari, A. F.; Bohlke-Schneider, M.; Zschiegner, J.; Wang, H.; and Wang, Y. B. 2023b. Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting. In *Advances in Neural Information Processing Systems*, 28341–28364.

Kraus, M.; Divo, F.; Steinmann, D.; Dhami, D. S.; and Kersting, K. 2024. United We Pretrain, Divided We Fail! Representation Learning for Time Series by Pretraining on 75 Datasets at Once. *arXiv preprint*, arXiv:2402.15404.

Lee, D.; Malacarne, S.; and Aune, E. 2023. Vector Quantized Time Series Generation with a Bidirectional Prior Model. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 7665–7693.

Li, H.; Yu, S.; and Príncipe, J. C. 2023. Causal Recurrent Variational Autoencoder for Medical Time Series Generation. In Williams, B.; Chen, Y.; and Neville, J., eds., *Proceedings of the AAAI Conference on Artificial Intelligence*, 8562–8570.

Ni, Z.; Yu, H.; Liu, S.; Li, J.; and Lin, W. 2023. Basisformer: Attention-based time series forecasting with learnable and interpretable basis. *Advances in Neural Information Processing Systems*, 71222–71241.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *International Conference on Learning Representations*.

Rasul, K.; Seward, C.; Schuster, I.; and Vollgraf, R. 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the International Conference on Machine Learning*, 8857–8868.

Shen, L.; and Kwok, J. T. 2023. Non-autoregressive conditional diffusion models for time series prediction. In *Proceedings of the International Conference on Machine Learning*, 31016–31029.

Sohl-Dickstein, J.; Weiss, E. A.; Maheswaranathan, N.; and Ganguli, S. 2015. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the International Conference on Machine Learning*, 2256–2265.

Woo, G.; Liu, C.; Kumar, A.; Xiong, C.; Savarese, S.; and Sahoo, D. 2024. Unified Training of Universal Time Series Forecasting Transformers. In *Proceedings of the International Conference on Machine Learning*, 53140–53164.

Yang, Y.; Jin, M.; Wen, H.; Zhang, C.; Liang, Y.; Ma, L.; Wang, Y.; Liu, C.; Yang, B.; Xu, Z.; Bian, J.; Pan, S.; and Wen, Q. 2024. A Survey on Diffusion Models for Time Series and Spatio-Temporal Data. *arXiv preprint*, arXiv:2404.18886.

Yoon, J.; Jarrett, D.; and van der Schaar, M. 2019. Time-series Generative Adversarial Networks. In *Advances in Neural Information Processing Systems*, 5509–5519.

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