

---

# CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation

---

**Yusuke Tashiro<sup>123\*</sup>, Jiaming Song<sup>1</sup>, Yang Song<sup>1</sup>, Stefano Ermon<sup>1</sup>**

<sup>1</sup>Department of Computer Science, Stanford University, Stanford, CA, USA

<sup>2</sup>Mitsubishi UFJ Trust Investment Technology Institute, Tokyo, Japan

<sup>3</sup>Japan Digital Design, Tokyo, Japan

{ytashiro, tsong, songyang, ermon}@cs.stanford.edu

## Abstract

The imputation of missing values in time series has many applications in healthcare and finance. While autoregressive models are natural candidates for time series imputation, score-based diffusion models have recently outperformed existing counterparts including autoregressive models in many tasks such as image generation and audio synthesis, and would be promising for time series imputation. In this paper, we propose Conditional Score-based Diffusion models for Imputation (CSDI), a novel time series imputation method that utilizes score-based diffusion models conditioned on observed data. Unlike existing score-based approaches, the conditional diffusion model is explicitly trained for imputation and can exploit correlations between observed values. On healthcare and environmental data, CSDI improves by 40-65% over existing probabilistic imputation methods on popular performance metrics. In addition, deterministic imputation by CSDI reduces the error by 5-20% compared to the state-of-the-art deterministic imputation methods. Furthermore, CSDI can also be applied to time series interpolation and probabilistic forecasting, and is competitive with existing baselines. The code is available at <https://github.com/ermongroup/CSDI>.

## 1 Introduction

Multivariate time series are abundant in real world applications such as finance, meteorology and healthcare. These time series data often contain missing values due to various reasons, including device failures and human errors [1–3]. Since missing values can hamper the interpretation of a time series, many studies have addressed the task of imputing missing values using machine learning techniques [4–6]. In the past few years, imputation methods based on deep neural networks have shown great success for both deterministic imputation [7–9] and probabilistic imputation [10]. These imputation methods typically utilize autoregressive models to deal with time series.

Score-based diffusion models – a class of deep generative models and generate samples by gradually converting noise into a plausible data sample through denoising – have recently achieved state-of-the-art sample quality in many tasks such as image generation [11, 12] and audio synthesis [13, 14], outperforming counterparts including autoregressive models. Diffusion models can also be used to impute missing values by approximating the scores of the posterior distribution obtained from the prior by conditioning on the observed values [12, 15, 16]. While these approximations may work well in practice, they do not correspond to the exact conditional distribution.

In this paper, we propose CSDI, a novel probabilistic imputation method that directly learns the conditional distribution with *conditional* score-based diffusion models. Unlike existing score-based approaches, the conditional diffusion model is designed for imputation and can exploit useful information in observed values. We illustrate the procedure of time series imputation with CSDI in

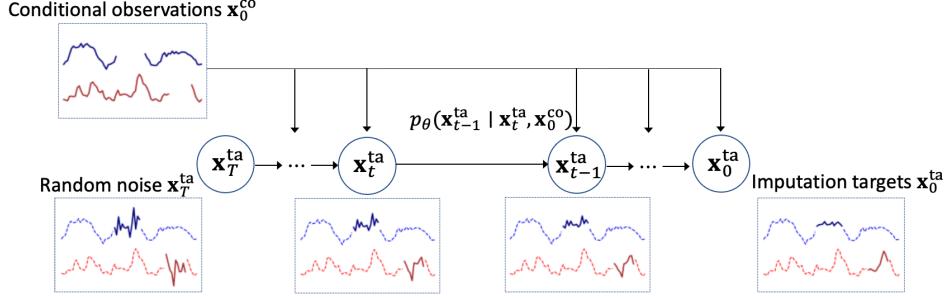


Figure 1: The procedure of time series imputation with CSDI. The reverse process  $p_\theta$  gradually converts random noise into plausible time series, conditioned on observed values  $x_0^{\text{co}}$ . Dashed lines in each box represent observed values, which are plotted in order to show the relationship with generated imputation and not included in each  $x_t^{\text{ta}}$ .

Figure 1. We start imputation from random noise on the left of the figure and gradually convert the noise into plausible time series through the reverse process  $p_\theta$  of the conditional diffusion model. At each step  $t$ , the reverse process removes noise from the output of the previous step ( $t + 1$ ). Unlike existing score-based diffusion models, the reverse process can take observations (on the top left of the figure) as a conditional input, allowing the model to exploit information in the observations for denoising. We utilize an attention mechanism to capture the temporal and feature dependencies of time series.

For training the conditional diffusion model, we need observed values (i.e., conditional information) and ground-truth missing values (i.e., imputation targets). However, in practice we do not know the ground-truth missing values, or training data may not contain missing values at all. Then, inspired by masked language modeling, we develop a self-supervised training method that separates observed values into conditional information and imputation targets. We note that CSDI is formulated for general imputation tasks, and is not restricted to time series imputation.

Our main contributions are as follows:

- We propose conditional score-based diffusion models for probabilistic imputation (CSDI), and implement CSDI for time series imputation. To train the conditional diffusion model, we develop a self-supervised training method.
- We empirically show that CSDI improves the continuous ranked probability score (CRPS) by 40-65% over existing probabilistic methods on healthcare and environmental data. Moreover, deterministic imputation with CSDI decreases the mean absolute error (MAE) by 5-20% compared to the state-of-the-art methods developed for deterministic imputation.
- We demonstrate that CSDI can also be applied to time series interpolations and probabilistic forecasting, and is competitive with existing baselines designed for these tasks.

## 2 Related works

**Time series imputations with deep learning** Previous studies have shown deep learning models can capture the temporal dependency of time series and give more accurate imputation than statistical methods. A popular approach using deep learning is to use RNNs, including LSTMs and GRUs, for sequence modeling [17, 8, 7]. Subsequent studies combined RNNs with other methods to improve imputation performance, such as GANs [9, 18, 19] and self-training [20]. Among them, the combination of RNNs with attention mechanisms is particularly successful for imputation and interpolation of time series [21, 22]. While these methods focused on deterministic imputation, GP-VAE [10] has been recently developed as a probabilistic imputation method.

**Score-based generative models** Score-based generative models, including score matching with Langevin dynamics [23] and denoising diffusion probabilistic models [11], have outperformed existing methods with other deep generative models in many domains, such as images [23, 11],

audio [13, 14], and graphs [24]. Most recently, TimeGrad [25] utilized diffusion probabilistic models for probabilistic time series forecasting. While the method has shown state-of-the-art performance, it cannot be applied to time series imputation due to the use of RNNs to handle past time series.

### 3 Background

#### 3.1 Multivariate time series imputation

We consider  $N$  multivariate time series with missing values. Let us denote the values of each time series as  $\mathbf{X} = \{x_{1:K,1:L}\} \in \mathbb{R}^{K \times L}$  where  $K$  is the number of features and  $L$  is the length of time series. While the length  $L$  can be different for each time series, we treat the length of all time series as the same for simplicity, unless otherwise stated. We also denote an observation mask as  $\mathbf{M} = \{m_{1:K,1:L}\} \in \{0, 1\}^{K \times L}$  where  $m_{k,l} = 0$  if  $x_{k,l}$  is missing, and  $m_{k,l} = 1$  if  $x_{k,l}$  is observed. We assume time intervals between two consecutive data entries can be different, and define the timestamps of the time series as  $\mathbf{s} = \{s_{1:L}\} \in \mathbb{R}^L$ . In summary, each time series is expressed as  $\{\mathbf{X}, \mathbf{M}, \mathbf{s}\}$ .

Probabilistic time series imputation is the task of estimating the distribution of the missing values of  $\mathbf{X}$  by exploiting the observed values of  $\mathbf{X}$ . We note that this definition of imputation includes other related tasks, such as interpolation, which imputes all features at target time points, and forecasting, which imputes all features at future time points.

#### 3.2 Denoising diffusion probabilistic models

Let us consider learning a model distribution  $p_\theta(\mathbf{x}_0)$  that approximates a data distribution  $q(\mathbf{x}_0)$ . Let  $\mathbf{x}_t$  for  $t = 1, \dots, T$  be a sequence of latent variables in the same sample space as  $\mathbf{x}_0$ , which is denoted as  $\mathcal{X}$ . Diffusion probabilistic models [26] are latent variable models that are composed of two processes: the forward process and the reverse process. The forward process is defined by the following Markov chain:

$$q(\mathbf{x}_{1:T} \mid \mathbf{x}_0) := \prod_{t=1}^T q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \text{ where } q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\sqrt{1 - \beta_t} \mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \quad (1)$$

and  $\beta_t$  is a small positive constant that represents a noise level. Sampling of  $\mathbf{x}_t$  has the closed-form written as  $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t} \mathbf{x}_0, (1 - \alpha_t) \mathbf{I})$  where  $\hat{\alpha}_t := 1 - \beta_t$  and  $\alpha_t := \prod_{i=1}^t \hat{\alpha}_i$ . Then,  $\mathbf{x}_t$  can be expressed as  $\mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + (1 - \alpha_t) \boldsymbol{\epsilon}$  where  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ . On the other hand, the reverse process denoises  $\mathbf{x}_t$  to recover  $\mathbf{x}_0$ , and is defined by the following Markov chain:

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}) &:= p(\mathbf{x}_T) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t), \quad \mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) &:= \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \sigma_\theta(\mathbf{x}_t, t) \mathbf{I}). \end{aligned} \quad (2)$$

Ho et al. [11] has recently proposed denoising diffusion probabilistic models (DDPM), which considers the following specific parameterization of  $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ :

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\alpha_t} \left( \mathbf{x}_t - \frac{\beta_t}{\sqrt{1 - \alpha_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right), \quad \sigma_\theta(\mathbf{x}_t, t) = \tilde{\beta}_t^{1/2} \text{ where } \tilde{\beta}_t = \begin{cases} \frac{1 - \alpha_{t-1}}{1 - \alpha_t} \beta_t & t > 1 \\ \beta_1 & t = 1 \end{cases} \quad (3)$$

where  $\boldsymbol{\epsilon}_\theta$  is a trainable denoising function. We denote  $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$  and  $\sigma_\theta(\mathbf{x}_t, t)$  in Eq. (3) as  $\boldsymbol{\mu}^{\text{DDPM}}(\mathbf{x}_t, t, \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))$  and  $\sigma^{\text{DDPM}}(\mathbf{x}_t, t)$ , respectively. The denoising function in Eq. (3) also corresponds to a rescaled score model for score-based generative models [23]. Under this parameterization, Ho et al. [11] have shown that the reverse process can be trained by solving the following optimization problem:

$$\min_{\theta} \mathcal{L}(\theta) := \min_{\theta} \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|_2^2 \quad \text{where } \mathbf{x}_t = \sqrt{\alpha_t} \mathbf{x}_0 + (1 - \alpha_t) \boldsymbol{\epsilon}. \quad (4)$$

The denoising function  $\boldsymbol{\epsilon}_\theta$  estimates the noise vector  $\boldsymbol{\epsilon}$  that was added to its noisy input  $\mathbf{x}_t$ . This training objective also be viewed as a weighted combination of denoising score matching used for training score-based generative models [23, 27, 12]. Once trained, we can sample  $\mathbf{x}_0$  from Eq. (2). We provide the details of DDPM in Appendix A.

### 3.3 Imputation with diffusion models

Here, we focus on general imputation tasks that are not restricted to time series imputation. Let us consider the following imputation problem: given a sample  $\mathbf{x}_0$  which contains missing values, we generate imputation targets  $\mathbf{x}_0^{\text{ta}} \in \mathcal{X}^{\text{ta}}$  by exploiting conditional observations  $\mathbf{x}_0^{\text{co}} \in \mathcal{X}^{\text{co}}$ , where  $\mathcal{X}^{\text{ta}}$  and  $\mathcal{X}^{\text{co}}$  are a part of the sample space  $\mathcal{X}$  and vary per sample. Then, the goal of probabilistic imputation is to estimate the true conditional data distribution  $q(\mathbf{x}_0^{\text{ta}} | \mathbf{x}_0^{\text{co}})$  with a model distribution  $p_\theta(\mathbf{x}_0^{\text{ta}} | \mathbf{x}_0^{\text{co}})$ . We typically impute all missing values using all observed values, and set all observed values as  $\mathbf{x}_0^{\text{co}}$  and all missing values as  $\mathbf{x}_0^{\text{ta}}$ , respectively. Note that time series imputation in Section 3.1 can be considered as a special case of this task.

Let us consider modeling  $p_\theta(\mathbf{x}_0^{\text{ta}} | \mathbf{x}_0^{\text{co}})$  with a diffusion model. In the unconditional case, the reverse process  $p_\theta(\mathbf{x}_{0:T})$  is used to define the final data model  $p_\theta(\mathbf{x}_0)$ . Then, a natural approach is to extend the reverse process in Eq. (2) to a conditional one:

$$\begin{aligned} p_\theta(\mathbf{x}_{0:T}^{\text{ta}} | \mathbf{x}_0^{\text{co}}) &:= p(\mathbf{x}_T^{\text{ta}}) \prod_{t=1}^T p_\theta(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}}), \quad \mathbf{x}_T^{\text{ta}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \\ p_\theta(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}}) &:= \mathcal{N}(\mathbf{x}_{t-1}^{\text{ta}}; \boldsymbol{\mu}_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}), \sigma_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}})\mathbf{I}). \end{aligned} \quad (5)$$

However, existing diffusion models are generally designed for data generation and do not take conditional observations  $\mathbf{x}_0^{\text{co}}$  as inputs. To utilize diffusion models for imputation, previous studies [12, 15, 16] approximated the conditional reverse process  $p_\theta(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}})$  with the reverse process in Eq. (2). With this approximation, in the reverse process they add noise to both the target and the conditional observations  $\mathbf{x}_0^{\text{co}}$ . While this approach can impute missing values, the added noise can harm useful information in the observations. This suggests that modeling  $p_\theta(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}})$  without approximations can improve the imputation quality. Hereafter, we call the model defined in Section 3.2 as the unconditional diffusion model.

## 4 Conditional score-based diffusion model for imputation (CSDI)

In this section, we propose CSDI, a novel imputation method based on a conditional score-based diffusion model. The conditional diffusion model allows us to exploit useful information in observed values for accurate imputation. We provide the reverse process of the conditional diffusion model, and then develop a self-supervised training method. We note that CSDI is not restricted to time series.

### 4.1 Imputation with CSDI

We focus on the conditional diffusion model with the reverse process in Eq. (5) and aim to model the conditional distribution  $p(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}})$  without approximations. Specifically, we extend the parameterization of DDPM in Eq. (3) to the conditional case. We define a conditional denoising function  $\epsilon_\theta : (\mathcal{X}^{\text{ta}} \times \mathbb{R} | \mathcal{X}^{\text{co}}) \rightarrow \mathcal{X}^{\text{ta}}$ , which takes conditional observations  $\mathbf{x}_0^{\text{co}}$  as inputs. Then, we consider the following parameterization with  $\epsilon_\theta$ :

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}) = \boldsymbol{\mu}^{\text{DDPM}}(\mathbf{x}_t^{\text{ta}}, t, \epsilon_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}})), \quad \sigma_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}) = \sigma^{\text{DDPM}}(\mathbf{x}_t^{\text{ta}}, t) \quad (6)$$

where  $\boldsymbol{\mu}^{\text{DDPM}}$  and  $\sigma^{\text{DDPM}}$  are the functions defined in Section 3.2. Given the function  $\epsilon_\theta$  and data  $\mathbf{x}_0$ , we can sample  $\mathbf{x}_0^{\text{ta}}$  using the reverse process in Eq. (5) and Eq. (6). For the sampling, we set all observed values of  $\mathbf{x}_0$  as conditional observations  $\mathbf{x}_0^{\text{co}}$  and all missing values as imputation targets  $\mathbf{x}_0^{\text{ta}}$ . Note that the conditional model is reduced to the unconditional one under no conditional observations and can also be used for data generation.

### 4.2 Training of CSDI

Since Eq. (6) uses the same parameterization as Eq. (3) and the difference between Eq. (3) and Eq. (6) is only the form of  $\epsilon_\theta$ , we can follow the training procedure for the unconditional model in Section 3.2. Namely, given conditional observations  $\mathbf{x}_0^{\text{co}}$  and imputation targets  $\mathbf{x}_0^{\text{ta}}$ , we sample noisy targets  $\mathbf{x}_t^{\text{ta}} = \sqrt{\alpha_t} \mathbf{x}_0^{\text{ta}} + (1 - \alpha_t) \boldsymbol{\epsilon}$ , and train  $\epsilon_\theta$  by minimizing the following loss function:

$$\min_\theta \mathcal{L}(\theta) := \min_\theta \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \|(\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}))\|_2^2 \quad (7)$$

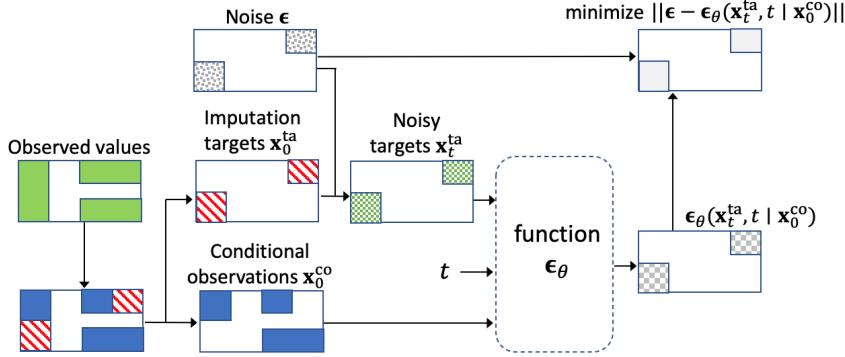


Figure 2: The self-supervised training procedure of CSDI. On the middle left rectangle, the green and white areas represent observed and missing values, respectively. The observed values are separated into red imputation targets  $x_0^{ta}$  and blue conditional observations  $x_0^{co}$ , and used for training of  $\epsilon_\theta$ . The colored areas in each rectangle mean the existence of values.

Table 1: Imputation targets  $x_0^{ta}$  and conditional observations  $x_0^{co}$  for CSDI at training and sampling.

	imputation targets $x_0^{ta}$	conditional observations $x_0^{co}$
sampling (imputation)	all missing values	all observed values
training	a subset of the observed values (sampled by a target choice strategy)	the remaining observed values

where the dimension of  $\epsilon$  corresponds to that of the imputation targets  $x_0^{ta}$ .

However, this training procedure has an issue. Since we do not know the ground-truth missing values in practice, it is not clear how to select  $x_0^{co}$  and  $x_0^{ta}$  from a training sample  $x_0$ . To address this issue, we develop a self-supervised learning method inspired by masked language modeling [28]. We illustrate the training procedure in Figure 2. Given a sample  $x_0$ , we separate observed values of  $x_0$  into two parts, and set one of them as imputation targets  $x_0^{ta}$  and the other as conditional observations  $x_0^{co}$ . We choose the targets  $x_0^{ta}$  through a target choice strategy, which is discussed in Section 4.3. Then, we sample noisy targets  $x_t^{ta}$  and train  $\epsilon_\theta$  by solving Eq. (7). We summarize how we set  $x_0^{co}$  and  $x_0^{ta}$  for training and sampling in Table 1. We also provide the algorithm of training and sampling in Appendix B.1.

### 4.3 Choice of imputation targets in self-supervised learning

In the proposed self-supervised learning, the choice of imputation targets is important. We provide four target choice strategies depending on what is known about the missing patterns in the test dataset. We describe the algorithm for these strategies in Appendix B.2.

(1) *Random* strategy : this strategy is used when we do not know about missing patterns, and randomly chooses a certain percentage of observed values as imputation targets. The percentage is sampled from [0%, 100%] to adapt to various missing ratios in the test dataset.

(2) *Historical* strategy: this strategy exploits missing patterns in the training dataset. Given a training sample  $x_0$ , we randomly draw another sample  $\tilde{x}_0$  from the training dataset. Then, we set the intersection of the observed indices of  $x_0$  and the missing indices of  $\tilde{x}_0$  as imputation targets. The motivation of this strategy comes from structured missing patterns in the real world. For example, missing values often appear consecutively in time series data. When missing patterns in the training and test dataset are highly correlated, this strategy helps the model learn a good conditional distribution.

(3) *Mix* strategy: this strategy is the mix of the above two strategies. The historical strategy may lead to overfitting to missing patterns in the training dataset. The *Mix* strategy can benefit from generalization by the random strategy and structured missing patterns by the historical strategy.

(4) *Test* pattern strategy: when we know the missing patterns in the test dataset, we just set the patterns as imputation targets. For example, this strategy is used for time series forecasting, since the missing patterns in the test dataset are fixed to given future time points.

## 5 Implementation of CSDI for time series imputation

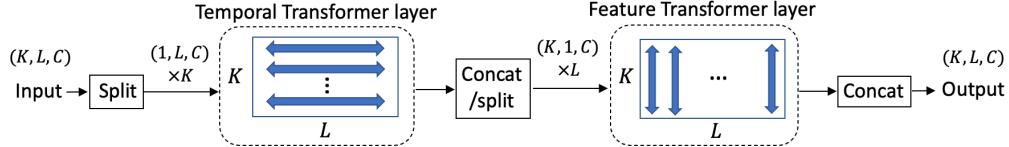


Figure 3: The architecture of 2D attention. Given a tensor with  $K$  features,  $L$  length, and  $C$  channels, the temporal Transformer layer takes tensors with  $(1, L, C)$  shape as inputs and learns temporal dependency. The feature Transformer layer takes tensors with  $(K, 1, C)$  shape as inputs and learns feature dependency. The output shape of each layer is the same as the input shape.

In this section, we implement CSDI for time series imputation. For the implementation, we need the inputs and the architecture of  $\epsilon_\theta$ .

First, we describe how we process time series data as inputs for CSDI. As defined in Section 3.1, a time series is denoted as  $\{\mathbf{X}, \mathbf{M}, \mathbf{s}\}$ , and the sample space  $\mathcal{X}$  of  $\mathbf{X}$  is  $\mathbb{R}^{K \times L}$ . We want to handle  $\mathbf{X}$  in the sample space  $\mathbb{R}^{K \times L}$  for learning dependencies in a time series using a neural network, but the conditional denoising function  $\epsilon_\theta$  takes inputs  $\mathbf{x}_t^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$  in varying sample spaces that are a part of  $\mathcal{X}$  as shown in white areas of  $\mathbf{x}_t^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$  in Figure 2. To address this issue, we adjust the conditional denoising function  $\epsilon_\theta$  to inputs in the fixed sample space  $\mathbb{R}^{K \times L}$ . Concretely, we fix the shape of the inputs  $\mathbf{x}_t^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$  to  $(K \times L)$  by applying zero padding to  $\mathbf{x}_t^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$ . In other words, we set zero values to white areas for  $\mathbf{x}_t^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$  in Figure 2. To indicate which indices are padded, we introduce the conditional mask  $\mathbf{m}^{\text{co}} \in \{0, 1\}^{K \times L}$  as an additional input to  $\epsilon_\theta$ , which corresponds to  $\mathbf{x}_0^{\text{co}}$  and takes value 1 for indices of conditional observations. For ease of handling, we also fix the output shape in the sample space  $\mathbb{R}^{K \times L}$  by applying zero padding. Then, the conditional denoising function  $\epsilon_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}, \mathbf{m}^{\text{co}})$  can be written as  $\epsilon_\theta : (\mathbb{R}^{K \times L} \times \mathbb{R} | \mathbb{R}^{K \times L} \times \{0, 1\}^{K \times L}) \rightarrow \mathbb{R}^{K \times L}$ . We discuss the effect of this adjustment on training and sampling in Appendix D.

Under the adjustment, we set conditional observations  $\mathbf{x}_0^{\text{co}}$  and imputation targets  $\mathbf{x}_0^{\text{ta}}$  for time series imputation by following Table 1. At sampling time, since conditional observations  $\mathbf{x}_0^{\text{co}}$  are all observed values, we set  $\mathbf{m}^{\text{co}} = \mathbf{M}$  and  $\mathbf{x}_0^{\text{co}} = \mathbf{m}^{\text{co}} \odot \mathbf{X}$  where  $\odot$  represents element-wise products. For training, we sample  $\mathbf{x}_0^{\text{ta}}$  and  $\mathbf{x}_0^{\text{co}}$  through a target choice strategy, and set the indices of  $\mathbf{x}_0^{\text{co}}$  as  $\mathbf{m}^{\text{co}}$ . Then,  $\mathbf{x}_0^{\text{co}}$  is written as  $\mathbf{x}_0^{\text{co}} = \mathbf{m}^{\text{co}} \odot \mathbf{X}$  and  $\mathbf{x}_0^{\text{ta}}$  is obtained as  $\mathbf{x}_0^{\text{ta}} = (\mathbf{M} - \mathbf{m}^{\text{co}}) \odot \mathbf{X}$ .

Next, we describe the architecture of  $\epsilon_\theta$ . We adopt the architecture in DiffWave [13] as the base, which is composed of multiple residual layers with residual channel  $C$ . We refine this architecture for time series imputation. We set the diffusion step  $T = 50$ . We discuss the main differences from DiffWave (see Appendix E.1 for the whole architecture and details).

**Attention mechanism** To capture temporal and feature dependencies of multivariate time series, we utilize a two dimensional attention mechanism in each residual layer instead of a convolution architecture. As shown in Figure 3, we introduce temporal Transformer layer and a feature Transformer layer, which are 1-layer Transformer encoders. The temporal Transformer layer takes tensors for each feature as inputs to learn temporal dependency, whereas the feature Transformer layer takes tensors for each time point as inputs to learn temporal dependency.

Note that while the length  $L$  can be different for each time series as mentioned in Section 3.1, the attention mechanism allows the model to handle various lengths. For batch training, we apply zero padding to each sequence so that the lengths of the sequences are the same.

**Side information** In addition to the arguments of  $\epsilon_\theta$ , we provide some side information as additional inputs to the model. First, we use time embedding of  $s = \{s_{1:L}\}$  to learn the temporal dependency. Following previous studies [29, 30], we use 128-dimensions temporal embedding. Second, we exploit categorical feature embedding for  $K$  features, where the dimension is 16.

## 6 Experimental results

In this section, we demonstrate the effectiveness of CSDI for time series imputation. Since CSDI can be applied to other related tasks such as interpolation and forecasting, we also evaluate CSDI for these tasks to show the flexibility of CSDI. Due to the page limitation, we provide the detailed setup for experiments including train/validation/test splits and hyperparameters in Appendix E.2.

### 6.1 Time series imputation

**Dataset and experiment settings** We run experiments for two datasets. The first one is the healthcare dataset in PhysioNet Challenge 2012 [1], which consists of 4000 clinical time series with 35 variables for 48 hours from intensive care unit (ICU). Following previous studies [7, 8], we process the dataset to hourly time series with 48 time steps. The processed dataset contains around 80% missing values. Since the dataset has no ground-truth, we randomly choose 10/50/90% of observed values as ground-truth on the test data.

The second one is the air quality dataset [2]. Following previous studies [7, 21], we use hourly sampled PM2.5 measurements from 36 stations in Beijing for 12 months and set 36 consecutive time steps as one time series. There are around 13% missing values and the missing patterns are not random. The dataset contains artificial ground-truth, whose missing patterns are also structured.

For both dataset, we run each experiment five times. As the target choice strategy for training, we adopt the random strategy for the healthcare dataset and the mix of the random and historical strategy for the air quality dataset, based on the missing patterns of each dataset.

**Results of probabilistic imputation** CSDI is compared with three baselines. 1) Multitask GP [31]: the method learns the covariance between timepoints and features simultaneously. 2) GP-VAE [10]: the method showed the state-of-the-art results for probabilistic imputation. 3) V-RIN [32]: a deterministic imputation method that uses the uncertainty quantified by VAE to improve imputation. For V-RIN, we regard the quantified uncertainty as probabilistic imputation. In addition, we compare CSDI with imputation using the unconditional diffusion model in order to show the effectiveness of the conditional one (see Appendix C for training and imputation with the unconditional diffusion model).

We first show quantitative results. We adopt the continuous ranked probability score (CRPS) [33] as the metric, which is frequently used for evaluating probabilistic time series forecasting and measures the compatibility of an estimated probability distribution with an observation. We generate 100 samples to approximate the probability distribution over missing values and report the normalized average of CRPS for all missing values following previous studies [34] (see Appendix E.3 for details of the computation).

Table 2: Comparing CRPS for probabilistic imputation baselines and CSDI (lower is better). We report the mean and the standard error of CRPS for five trials.

	healthcare			air quality
	10% missing	50% missing	90% missing	
Multitask GP [31]	0.489(0.005)	0.581(0.003)	0.942(0.010)	0.301(0.003)
GP-VAE [10]	0.574(0.003)	0.774(0.004)	0.998(0.001)	0.397(0.009)
V-RIN [32]	0.808(0.008)	0.831(0.005)	0.922(0.003)	0.526(0.025)
unconditional	0.360(0.007)	0.458(0.008)	0.671(0.007)	0.135(0.001)
<b>CSDI (proposed)</b>	<b>0.238(0.001)</b>	<b>0.330(0.002)</b>	<b>0.522(0.002)</b>	<b>0.108(0.001)</b>

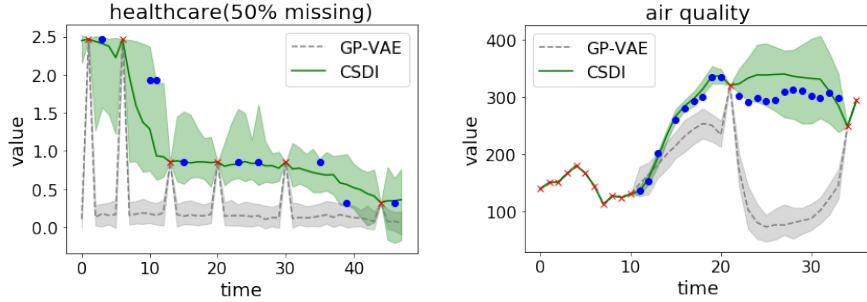


Figure 4: Examples of probabilistic time series imputation for the healthcare dataset with 50% missing (left) and the air quality dataset (right). The red crosses show the observed values and the blue circles show the ground-truth imputation targets. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

Table 2 represents CRPS for each method. CSDI reduces CRPS by 40-65% compared to the existing baselines for both datasets. This indicates that CSDI generates more realistic distributions than other methods. We also observe that the imputation with CSDI outperforms that with the unconditional model. This suggests CSDI benefits from explicitly modeling the conditional distribution.

We provide imputation examples in Figure 4. For the air quality dataset, CSDI (green solid line) provides accurate imputations with high confidence, while those by GP-VAE (gray dashed line) are far from ground-truth. CSDI also gives reasonable imputations for the healthcare dataset. These results indicate that CSDI exploits temporal and feature dependencies to provide accurate imputations. We give more examples in Appendix G.

Table 3: Comparing MAE for deterministic imputation methods and CSDI. We report the mean and the standard error for five trials. The asterisks mean the results of the method are cited from the original paper.

	healthcare			air quality
	10% missing	50% missing	90% missing	
V-RIN [32]	0.271(0.001)	0.365(0.002)	0.606(0.006)	25.4(0.62)
BRITS [7]	0.284(0.001)	0.368(0.002)	0.517(0.002)	14.11(0.26)
BRITS [7] (*)	0.278	—	—	11.56
GLIMA [21] (*)	0.265	—	—	10.54
RDIS [20]	0.319(0.002)	0.419(0.002)	0.631(0.002)	22.11(0.35)
unconditional	0.326(0.008)	0.417(0.010)	0.625(0.010)	12.13(0.07)
<b>CSDI (proposed)</b>	<b>0.217(0.001)</b>	<b>0.301(0.002)</b>	<b>0.481(0.003)</b>	<b>9.60(0.04)</b>

**Results of deterministic imputation** We demonstrate that CSDI also provides accurate deterministic imputations, which are obtained as the median of 100 generated samples. We compare CSDI with four baselines developed for deterministic imputation including GLIMA [21], which combined recurrent imputations with an attention mechanism to capture temporal and feature dependencies and showed the state-of-the-art performance. These methods are based on autoregressive models. We use the original implementations except RDIS.

We evaluate each method by the mean absolute error (MAE). In Table 3, CSDI improves MAE by 5-20% compared to the baselines. This suggests that the conditional diffusion model is effective to learn temporal and feature dependencies for imputation. For the healthcare dataset, the gap between the baselines and CSDI is particularly significant when the missing ratio is small, because more observed values help CSDI capture dependencies.

Table 4: Comparing the state-of-the-art interpolation methods with CSDI for the healthcare dataset. We report the mean and the standard error of CRPS for five trials.

	10% missing	50% missing	90% missing
Latent ODE [35]	0.700(0.002)	0.676(0.003)	0.761(0.010)
mTANs [22]	0.526(0.004)	0.567(0.003)	0.689(0.015)
<b>CSDI (proposed)</b>	<b>0.380(0.002)</b>	<b>0.418(0.001)</b>	<b>0.556(0.003)</b>

## 6.2 Interpolation of irregularly sampled time series

**Dataset and experiment settings** We use the same healthcare dataset as the previous section, but process the dataset as irregularly sampled time series, following previous studies [22, 35]. Since the dataset has no ground-truth, we randomly choose 10/50/90% of *time* points and use observed values at these time points as ground-truth on the test data. As the target choice strategy for training, we adopt the random strategy, which is adjusted for interpolation so that some *time* points are sampled.

**Results** We compare CSDI with two baselines including mTANs [22], which utilized an attention mechanism and showed state-of-the-art results for the interpolation of irregularly sampled time series. We generate 100 samples to approximate the probability distribution as with the previous section. The result is shown in Table 4. CSDI outperforms the baselines for all cases.

Table 5: Comparing probabilistic forecasting methods with CSDI. We report the mean and the standard error of CRPS-sum for three trials. The baseline results are cited from the original paper. ‘TransMAF’ is the abbreviation for ‘Transformer MAF’.

	solar	electricity	traffic	taxi	wiki
GP-copula [34]	0.337(0.024)	0.024(0.002)	0.078(0.002)	0.208(0.183)	0.086(0.004)
TransMAF [36]	0.301(0.014)	0.021(0.000)	0.056(0.001)	0.179(0.002)	0.063(0.003)
TLAE [37]	<b>0.124(0.033)</b>	0.040(0.002)	0.069(0.001)	0.130(0.006)	0.241(0.001)
TimeGrad [25]	0.287(0.020)	0.021(0.001)	0.044(0.006)	<b>0.114(0.020)</b>	0.049(0.002)
<b>CSDI (proposed)</b>	0.298(0.004)	<b>0.017(0.000)</b>	<b>0.020(0.001)</b>	0.123(0.003)	<b>0.047(0.003)</b>

## 6.3 Time series Forecasting

**Dataset and Experiment settings** We use five datasets that are commonly used for evaluating probabilistic time series forecasting. Each dataset is composed of around 100 to 2000 features. We predict all features at future time steps using past time series. We use the same prediction steps as previous studies [34, 37]. For the target choice strategy, we adopt the *Test* pattern strategy.

**Results** We compare CSDI with four baselines. Specifically, TimeGrad [25] combined the diffusion model with a RNN-based encoder. We evaluate each method for CRPS-sum, which is CRPS for the distribution of the sum of all time series across  $K$  features and accounts for joint effect (see Appendix E.3 for details).

In Table 5, CSDI outperforms the baselines for electricity and traffic datasets, and is competitive with the baselines as a whole. The advantage of CSDI over baselines for forecasting is smaller than that for imputation in Section 6.1. We hypothesize it is because the datasets for forecasting seldom contains missing values and are suitable for existing encoders including RNNs. For imputation, it is relatively difficult for RNNs to handle time series due to missing values.

## 7 Conclusion

In this paper, we have proposed CSDI, a novel approach to impute multivariate time series with conditional diffusion models. We have shown that CSDI outperforms the existing probabilistic and deterministic imputation methods.

There are some interesting directions for future work. One direction is to improve the computation efficiency. While diffusion models generate plausible samples, sampling is generally slower than other generative models. To mitigate the issue, several recent studies leverage an ODE solver to accelerate the sampling procedure [12, 38, 13]. Combining our method with these approaches would likely improve the sampling efficiency.

Another direction is to extend CSDI to downstream tasks such as classifications. Many previous studies have shown that accurate imputation improves the performance on downstream tasks [7, 18, 22]. Since conditional diffusion models can learn temporal and feature dependencies with uncertainty, joint training of imputations and downstream tasks using conditional diffusion models would be helpful to improve the performance of the downstream tasks.

Finally, although our focus was on time series, it would be interesting to explore CSDI as imputation technique on other modalities.

## Acknowledgements and Disclosure of Funding

This research was supported by NSF(#1651565, #1522054, #1733686), ONR (N000141912145), AFOSR (FA95501910024), ARO (W911NF-21-1-0125) and Sloan Fellowship.

## References

- [1] Ikaro Silva, George Moody, Daniel J Scott, Leo A Celi, and Roger G Mark. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *Computing in Cardiology*, pages 245–248. IEEE, 2012.
- [2] Xiuwen Yi, Yu Zheng, Junbo Zhang, and Tianrui Li. ST-MVL: filling missing values in geo-sensory time series data. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 2704–2710, 2016.
- [3] Huachun Tan, Guangdong Feng, Jianshuai Feng, Wuhong Wang, Yu-Jin Zhang, and Feng Li. A tensor-based method for missing traffic data completion. *Transportation Research Part C: Emerging Technologies*, 28:15–27, 2013.
- [4] Fulufhelo V Nelwamondo, Shakir Mohamed, and Tshilidzi Marwala. Missing data: A comparison of neural network and expectation maximization techniques. *Current Science*, pages 1514–1521, 2007.
- [5] Andrew T Hudak, Nicholas L Crookston, Jeffrey S Evans, David E Hall, and Michael J Falkowski. Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data. *Remote Sensing of Environment*, 112(5):2232–2245, 2008.
- [6] S van Buuren and Karin Groothuis-Oudshoorn. MICE: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68, 2010.
- [7] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: Bidirectional recurrent imputation for time series. In *Advances in Neural Information Processing Systems*, 2018.
- [8] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):1–12, 2018.
- [9] Yonghong Luo, Xiangrui Cai, Ying Zhang, Jun Xu, and Xiaojie Yuan. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1603–1614, 2018.
- [10] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, 2020.
- [11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

- [12] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.
- [13] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.
- [14] Nanxin Chen, Yu Zhang, Heiga Zen, Ron J Weiss, Mohammad Norouzi, and William Chan. WaveGrad: Estimating gradients for waveform generation. In *International Conference on Learning Representations*, 2021.
- [15] Zahra Kadkhodaie and Eero P Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser. *arXiv preprint arXiv:2007.13640*, 2020.
- [16] Gautam Mittal, Jesse Engel, Hawthorne Curtis, and Ian Simon. Symbolic music generation with diffusion models. *arXiv preprint arXiv:2103.16091*, 2021.
- [17] Jinsung Yoon, William R Zame, and Mihaela van der Schaar. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Transactions on Biomedical Engineering*, 66(5):1477–1490, 2018.
- [18] Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. E2GAN: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 3094–3100, 2019.
- [19] Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. Generative semi-supervised learning for multivariate time series imputation. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.
- [20] Tae-Min Choi, Ji-Su Kang, and Jong-Hwan Kim. RDIS: Random drop imputation with self-training for incomplete time series data. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.
- [21] Qiuling Suo, Weida Zhong, Guangxu Xun, Jianhui Sun, Changyou Chen, and Aidong Zhang. GLIMA: Global and local time series imputation with multi-directional attention learning. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 798–807. IEEE, 2020.
- [22] Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021.
- [23] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.
- [24] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.
- [25] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. *arXiv preprint arXiv:2101.12072*, 2021.
- [26] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.
- [27] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, 2020.
- [28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- [30] Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, 2020.
- [31] Edwin V Bonilla, Kian Ming A Chai, and Christopher KI Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, 2008.
- [32] Ahmad Wisnu Mulyadi, Eunji Jun, and Heung-II Suk. Uncertainty-aware variational-recurrent imputation network for clinical time series. *IEEE Transactions on Cybernetics*, 2021. to appear.
- [33] James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.
- [34] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems*, 2019.
- [35] Yulia Rubanova, Ricky TQ Chen, and David Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2019.
- [36] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multi-variate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2021.
- [37] Nam Nguyen and Brian Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.
- [38] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.
- [39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.
- [40] Phil Wang. Linear attention transformer. <https://github.com/lucidrains/linear-attention-transformer>, 2020.
- [41] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021.
- [42] Alex Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. *arXiv preprint arXiv:2102.09672*, 2021.
- [43] Jacob R Gardner, Geoff Pleiss, David Bindel, Kilian Q Weinberger, and Andrew Gordon Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.
- [44] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.
- [45] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 95–104, 2018.

## A Details of denoising diffusion probabilistic models

In this section, we describe the details of denoising diffusion probabilistic models in Section 3.2.

Diffusion probabilistic models [26] are latent variable models that are composed of two processes: the forward process and the reverse process. The forward process and the reverse process are defined by Eq. (1) and 2, respectively. Then, the parameters  $\theta$  are learned by maximizing variational lower bound (ELBO) of likelihood  $p_\theta(\mathbf{x}_{0:T})$ :

$$\mathbb{E}_{q(\mathbf{x}_0)} [\log p_\theta(\mathbf{x}_0)] \geq \mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T)} [\log p_\theta(\mathbf{x}_{0:T}) - \log q(\mathbf{x}_{1:T} | \mathbf{x}_0)] := \text{ELBO}. \quad (8)$$

To analyse this ELBO, Ho et al. [11] proposed denoising diffusion probabilistic models (DDPM), which considered the parameterization given by Eq. (3). Under the parameterization, Ho et al. [11] showed ELBO satisfies the following equation:

$$-\text{ELBO} = c + \sum_{t=1}^T \kappa_t \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + (1 - \alpha_t)\epsilon, t)\|_2^2 \quad (9)$$

where  $c$  is a constant and  $\{\kappa_{1:T}\}$  are positive coefficients depending on  $\alpha_{1:T}$  and  $\beta_{1:T}$ . The diffusion process can be trained by minimizing Eq. (9). In addition, Ho et al. [11] found that minimizing the following unweighted version of ELBO leads to good sample quality:

$$\min_{\theta} L(\theta) := \min_{\theta} \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + (1 - \alpha_t)\epsilon, t)\|_2^2. \quad (10)$$

The function  $\epsilon_\theta$  estimates noise  $\epsilon$  in the noisy input. Once trained, we can sample  $\mathbf{x}_0$  from Eq. (2).

## B Algorithms

### B.1 Algorithm for training and sampling of CSDI

We provide the training procedure of CSDI in Algorithm 1 and the imputation (sampling) procedure with CSDI in Algorithm 2, which are described in Section 4.

---

#### Algorithm 1 Training of CSDI

- 1: **Input:** distribution of training data  $q(\mathbf{x}_0)$ , a target choice strategy  $\mathcal{T}$ , the number of iteration  $N_{\text{iter}}$ , the sequence of noise levels  $\{\alpha_t\}$
  - 2: **Output:** Trained denoising function  $\epsilon_\theta$
  - 3: **for**  $i = 1$  to  $N_{\text{iter}}$  **do**
  - 4:    $t \sim \text{Uniform}(\{1, \dots, T\})$ ,  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  - 5:   Separate observed values of  $\mathbf{x}_0$  into conditional information  $\mathbf{x}_0^{\text{co}}$  and imputation targets  $\mathbf{x}_0^{\text{ta}}$  by the target choice strategy  $\mathcal{T}$
  - 6:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  where the dimension of  $\epsilon$  corresponds to  $\mathbf{x}_0^{\text{ta}}$
  - 7:   Calculate noisy targets  $\mathbf{x}_t^{\text{ta}} = \sqrt{\alpha_t} \mathbf{x}_0^{\text{ta}} + (1 - \alpha_t)\epsilon$
  - 8:   Take gradient step on  $\nabla_{\theta} \|(\epsilon - \epsilon_\theta(\mathbf{x}_t^{\text{ta}}, t | \mathbf{x}_0^{\text{co}}))\|_2^2$  according to Eq. (7)
- 

---

#### Algorithm 2 Imputation (Sampling) with CSDI

- 1: **Input:** a data sample  $\mathbf{x}_0$ , trained denoising function  $\epsilon_\theta$
  - 2: **Output:** Imputed missing values  $\mathbf{x}_0^{\text{ta}}$
  - 3: Denote observed values of  $\mathbf{x}_0$  as  $\mathbf{x}_0^{\text{co}}$
  - 4:  $\mathbf{x}_T^{\text{ta}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  where the dimension of  $\mathbf{x}_T^{\text{ta}}$  corresponds to the missing indices of  $\mathbf{x}_0$
  - 5: **for**  $t = T$  to 1 **do**
  - 6:   Sample  $\mathbf{x}_{t-1}^{\text{ta}}$  using Eq. (5) and Eq. (6)
- 

### B.2 Target choice strategies for self-supervised training

We describe the target choice strategies for self-supervised training of CSDI, which is discussed in Section 4.3. We give the algorithm of the random strategy in Algorithm 3 and that of the historical

---

**Algorithm 3** Target choice with the random strategy

---

- 1: **Input:** a training sample  $\mathbf{x}_0$
  - 2: **Output:** conditional information  $\mathbf{x}_0^{\text{co}}$ , imputation targets  $\mathbf{x}_0^{\text{ta}}$
  - 3: Draw target ratio  $r \sim \text{Uniform}(0, 100)$
  - 4: Randomly choose  $r\%$  of the observed values of  $\mathbf{x}_0$  and denote the chosen observations as  $\mathbf{x}_0^{\text{ta}}$ , and denote the remaining observations as  $\mathbf{x}_0^{\text{co}}$
- 

**Algorithm 4** Target choice with the historical strategy

---

- 1: **Input:** a training sample  $\mathbf{x}_0$ , missing pattern dataset  $D_{\text{miss}}$
  - 2: **Output:** conditional information  $\mathbf{x}_0^{\text{co}}$ , imputation targets  $\mathbf{x}_0^{\text{ta}}$
  - 3: Draw a data sample  $\tilde{\mathbf{x}}_0$  from  $D_{\text{miss}}$
  - 4: Denote the indices of observed values of  $\mathbf{x}_0$  as  $J$
  - 5: Denote the indices of missing values of  $\tilde{\mathbf{x}}_0$  as  $\tilde{J}$
  - 6: Take the intersection of  $J$  and  $\tilde{J}$ , and denote values of  $\mathbf{x}_0$  for the intersection as  $\mathbf{x}_0^{\text{ta}}$
  - 7: Set the remaining observations of  $\mathbf{x}_0$  as  $\mathbf{x}_0^{\text{co}}$
- 

strategy in Algorithm 4. On the historical strategy, we use the training dataset as missing pattern dataset  $D_{\text{miss}}$ , unless otherwise stated. The mix strategy draws one of the two strategies with a ratio of 1:1 for each training sample. The test pattern strategy just uses the fixed missing pattern in the test dataset to choose imputation targets.

## C Training and imputation for unconditional diffusion model

### C.1 Imputation with unconditional diffusion model

We describe the imputation method with the unconditional diffusion model used for the experiments in Section 6.1. We followed the method described in previous studies [12]. To utilize unconditional diffusion models for imputation, they approximated the conditional reverse process  $p_\theta(\mathbf{x}_{t-1}^{\text{ta}} | \mathbf{x}_t^{\text{ta}}, \mathbf{x}_0^{\text{co}})$  in Eq. (5) with the unconditional reverse process in Eq. (2). Given a test sample  $\mathbf{x}_0$ , they set all observed values as conditional observations  $\mathbf{x}_0^{\text{co}}$  and all missing values as imputation targets  $\mathbf{x}_0^{\text{ta}}$ . Then, instead of conditional observations  $\mathbf{x}_0^{\text{co}}$ , they considered noisy conditional observations  $\mathbf{x}_t^{\text{co}} := \sqrt{\alpha_t} \mathbf{x}_0^{\text{co}} + (1 - \alpha_t)\epsilon$  and exploited  $\mathbf{x}_t = [\mathbf{x}_t^{\text{co}}; \mathbf{x}_t^{\text{ta}}] \in \mathcal{X}$  for the input to the distribution  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$  in Eq. (2), where  $[\mathbf{x}_t^{\text{co}}; \mathbf{x}_t^{\text{ta}}]$  combines  $\mathbf{x}_t^{\text{co}}$  and  $\mathbf{x}_t^{\text{ta}}$  to create a sample in  $\mathcal{X}$ . Using this approximation, we can sample  $\mathbf{x}_{t-1}$  from  $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t = [\mathbf{x}_t^{\text{co}}; \mathbf{x}_t^{\text{ta}}])$  and obtain  $\mathbf{x}_{t-1}^{\text{ta}}$  by extracting target indices from  $\mathbf{x}_{t-1}$ . By repeating the sampling procedure from  $t = T$  to  $t = 1$ , we can generate imputation targets  $\mathbf{x}_0^{\text{ta}}$ .

### C.2 Training procedure of unconditional diffusion models for time series imputation

In Section 3.2, we described the training procedure of the unconditional diffusion model, which expects the training dataset does not contain missing values. However, the training dataset that we use for time series imputation contains missing values. To handle missing values, we slightly modify the training procedure. Given a training sample  $\mathbf{x}_0$  with missing values, we treat the missing values like observed values by filling dummy values to the missing indices of  $\mathbf{x}_0$ . We adopt zeros for the dummy values and denote the training sample after filling zeros as  $\hat{\mathbf{x}}_0$ . Since all indices of  $\hat{\mathbf{x}}_0$  contain values, we can sample noisy targets  $\sqrt{\alpha_t} \hat{\mathbf{x}}_0 + (1 - \alpha_t)\epsilon$  as with the training procedure in Section 3.2. We consider denoising the noisy target for training, but we are only interested in estimating the noises added to the observed indices since the dummy values contain no information about the data distribution. To exclude the missing indices, we introduce an observation mask  $\mathbf{m} \in \{0, 1\}^{K \times L}$ , which takes value 1 for observed indices. Then, instead of Eq. (4), we use the following loss function for training under the existence of missing values:

$$\min_{\theta} L(\theta) := \min_{\theta} \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t} \|(\epsilon - \epsilon_\theta(\sqrt{\alpha_t} \hat{\mathbf{x}}_0 + (1 - \alpha_t)\epsilon, t)) \odot \mathbf{m}\|_2^2. \quad (11)$$

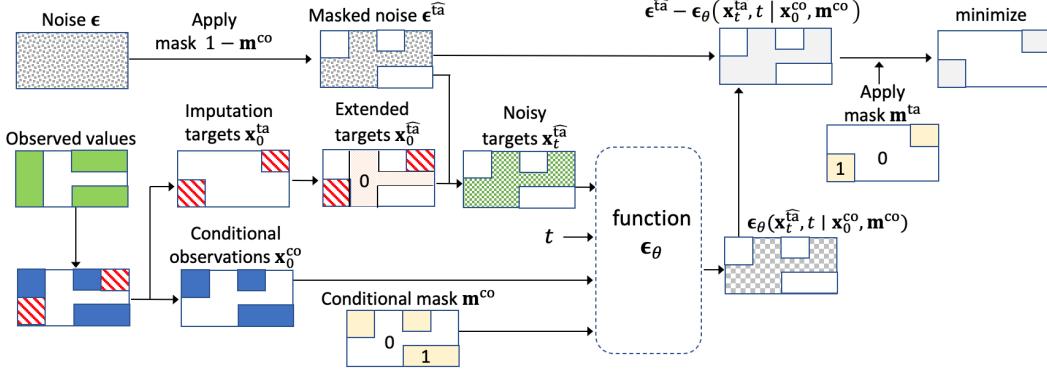


Figure 5: The self-supervised training procedure of CSDI for implementation of time series imputation. The colored areas in each rectangle represent the existence of values. The green and white areas represent observed and missing values, respectively, and white areas are padded with zeros to fix the shape of the inputs. Zero padding is also applied to all white areas. As with Figure 2, the observed values are separated into red imputation targets  $x_0^{\text{ta}}$  and blue conditional observations  $x_0^{\text{co}}$ . For the extended targets  $x_0^{\widehat{\text{ta}}}$ , the area of value 0 shows dummy values.

## D CSDI for implementation of time series imputation

In this section, we discuss the effect of adjusting the function  $\epsilon_\theta$ , described in Section 5. First, let us consider the effect of the adjustment on sampling. The adjustment does not essentially affect the model at sampling time, because all values of data are either conditional observations or imputation targets as shown in Table 1 and the model can distinguish the type of each value through the mask  $\mathbf{m}^{\text{co}}$ . Since the output shape is adjusted, we need to recover the shape by extracting the indices of the imputation targets from the output, so that we substitute the outputs into Eq. (6).

Next, we focus on the effect of the adjustment on training. Unlike sampling, the model at training time cannot distinguish imputation targets and missing values since we ignore missing values during training as shown in Table 1. In order to handle the missing values, we need to modify the inputs to  $\epsilon_\theta$ . Here, we use a similar approach to the training procedure with the unconditional model in Section C.2. Namely, we treat the missing indices like a part of imputation targets. We illustrate the extended training procedure in Figure 5. First, we set zeros to the missing indices as dummy values. We denote the extended imputation targets as  $x_0^{\widehat{\text{ta}}}$ . Then, we sample noisy targets  $x_t^{\widehat{\text{ta}}} = \sqrt{\alpha_t} x_0^{\widehat{\text{ta}}} + (1 - \alpha_t) \epsilon^{\widehat{\text{ta}}}$ , where  $\epsilon^{\widehat{\text{ta}}}$  is masked noise and is given by  $\epsilon^{\widehat{\text{ta}}} := (1 - \mathbf{m}^{\text{co}}) \odot \epsilon$ , as shown in Figure 5. We denoise the noisy targets for training. We only estimate the noise for the original imputation targets, since the dummy values contain no information about the data distribution. In other words, we train  $\epsilon_\theta$  by solving the following optimization problem:

$$\min_{\theta} \mathcal{L}(\theta) := \min_{\theta} \mathbb{E}_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0, I), t} \| (\epsilon - \epsilon_\theta(x_t^{\widehat{\text{ta}}}, t | x_0^{\text{co}}, \mathbf{m}^{\text{co}})) \odot \mathbf{m}^{\text{ta}} \|_2^2 \quad (12)$$

where  $\mathbf{m}^{\text{ta}}$  is a mask which corresponds to  $x_0^{\text{ta}}$  and takes value 1 for the original imputation targets.

## E Details of architectures and experiment settings

### E.1 Details of implementation of CSDI

We describe the details of architectures and hyperparameters for the conditional diffusion model described in Section 5. First, we provide the whole architecture of CSDI in Figure 6. Since the architecture in Figure 6 is based on DiffWave [13], we mainly explain the difference from DiffWave.

On the top of the figure, the models take  $x_0^{\text{co}}$  and  $x_t^{\text{ta}}$  as inputs since  $\epsilon_\theta$  is the conditional denoising function. For the diffusion step  $t$ , we use the following 128-dimensions embedding following previous works [29, 13]:

$$t_{\text{embedding}}(t) = \left( \sin(10^{0.4/63} t), \dots, \sin(10^{63.4/63} t), \cos(10^{0.4/63} t), \dots, \cos(10^{63.4/63} t) \right). \quad (13)$$

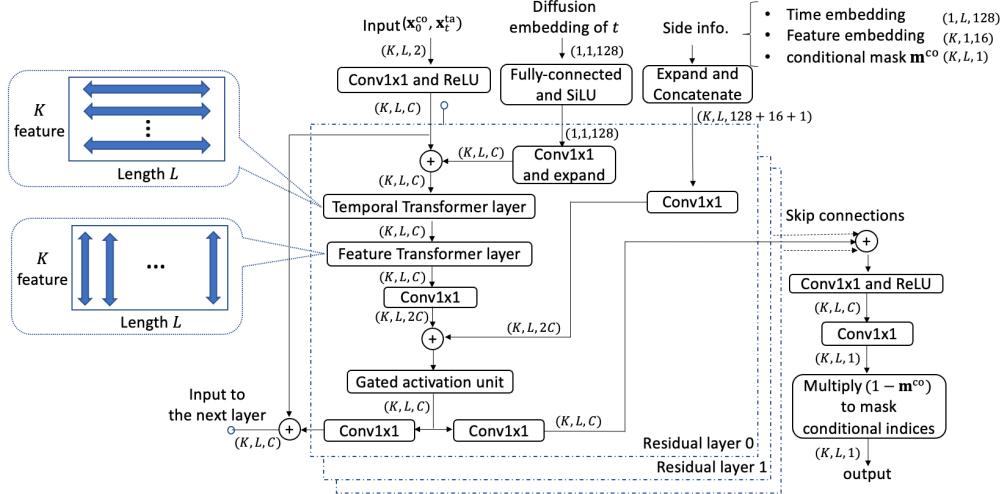


Figure 6: Architecture of  $\epsilon_\theta$  in CSDI for multivariate time series imputation.

Similarly, we utilize time embedding of  $s = \{s_{1:L}\}$  as a side information. We use 128-dimensions temporal embedding following previous studies [29, 30]:

$$s_{embedding}(s_l) = \left( \sin(s_l/\tau^{0/64}), \dots, \sin(s_l/\tau^{63/64}), \cos(s_l/\tau^{0/64}), \dots, \cos(s_l/\tau^{63/64}) \right) \quad (14)$$

where  $\tau = 10000$ . On the top right of the figure, we expand each side information and concatenate all the side information. On the bottom right of the figure, we multiply the output by a mask  $(1 - m^{co})$  in order to mask the indices of the conditional observations of the output.

As for Transformer layers, we used 1-layer TransformerEncoder implemented in PyTorch [39], which is composed of a multi-head attention layer, fully-connected layers and layer normalization. Only for forecasting tasks, we adopted the "linear attention transformer" package [40] to improve computational efficiency, since the forecasting datasets we used contained many features and long sequences. The package implements an efficient attention mechanism [41], and we only used global attention in the package.

## E.2 Details of experiment settings in Section 6

In this section, we provide the details of the experiment settings in Section 6. When we evaluated baseline methods with the original implementation in each section, we used their original hyperparameters and model size. Although we also ran experiments under the same model size as our model, the performance did not improve in more than half of the cases and did not outperform our model in all cases.

### E.2.1 Experiment settings for imputation in Section 6.1

First, we explain additional information for the air quality dataset. The dataset is composed of air quality data in Beijing from 2014/05/01 to 2015/04/30. The dataset contains artificial ground-truth, whose missing patterns are created based on those in the next month.

Next, we describe data splits. For the healthcare dataset, we randomly divided the dataset into five parts and used one of them as test data for each run. We also randomly split the remaining data into train and validation data with a ratio of 7:1. For the air quality dataset, following [2], we used the 3rd, 6th, 9th and 12th months as test data. To avoid evaluating imputation for each missing value multiple times, we separated the test data of each month every 36 consecutive time steps without overlap. When the length of a monthly data was not divisible by 36, we allowed the last sequence to overlap with the previous one and did not aggregate the result for the overlapping parts. For each run, we selected a month as validation data and used the rest as training data. We note that we excluded the 4th, 7th, 10th, and 1st months from missing pattern dataset for the historical strategy, because these months were used for creating missing patterns of the artificial ground-truth.

On the healthcare dataset, due to the different scales of features, we evaluate the performance on normalized data following previous studies [7]. For training of all tasks, we normalize each feature to have zero mean and unit variance.

As for hyperparameters, we set the batch size as 16 and the number of epochs as 200. We used Adam optimizer with learning rate 0.001 that is decayed to 0.0001 and 0.00001 at 75% and 90% of the total epochs, respectively. As for the model, we set the number of residual layers as 4, residual channels as 64, and attention heads as 8. We followed DiffWave[13] for the number of channels and decided the number of layers based on the validation loss and the parameter size. The number of the parameter in the model is about 415,000.

We also provide hyperparameters for the diffusion model as follows. We set the number of the diffusion step  $T = 50$ , the minimum noise level  $\beta_1 = 0.0001$ , and the maximum noise level  $\beta_T = 0.5$ . Since recent studies[38, 42] reported that gentle decay of  $\alpha_t$  could improve the sample quality, we adopted the following quadratic schedule for other noise levels:

$$\beta_t = \left( \frac{T-t}{T-1} \sqrt{\beta_1} + \frac{t-1}{T-1} \sqrt{\beta_T} \right)^2. \quad (15)$$

With regard to the baselines for probabilistic imputation, we used their original implementations for GP-VAE and V-RIN. For Multitask GP, we utilized GPyTorch [43] for the implementation. We used RBF kernel for the covariance between timepoints and low-rank IndexKernel with  $rank = 10$  for that between features.

Finally, we describe the baselines for deterministic imputation, which were used for comparison. 1) BRITS [7]: the method utilizes a bi-directional recurrent neural network to handle multiple correlated missing values. 2) V-RIN [32]: the method utilizes the uncertainty learned with VAE to improve recurrent imputation. 3) GLIMA [21]: the method combines recurrent imputations with an attention mechanism to capture cross-time and cross-feature dependencies and shows the state-of-the-art performance. 4) RDIS [20]: the method applies random drops to given training data for self-training. We used the original implementation for BRITS and V-RIN. For RDIS, we set the number of models as 8, hidden units as 108, drop rate as 30%, threshold as 0.1, update epoch as 200, and total epochs as 1000.

### E.2.2 Experiment settings for interpolation in Section 6.2

First, we explain how we process the dataset. We processed the healthcare dataset as irregularly sampled time series. Following previous studies [22, 35], we rounded observation times to the nearest minute. Then, there are  $48 \times 60$  possible measurement times per time series, and the lengths of time series samples can be different each other.

We used almost the same experiment settings as those for imputation in Section E.2.1. Since the length of each irregularly sampled time series is different, we applied zero padding to each time series in order to fix the length for each batch. The padding does not affect the result since the attention mechanisms in the implementation of CSDI can deal with the padding by using padding masks.

We describe the baselines which were used for comparison. We used the original implementation. 1) Latent ODE [35]: the method consists of an ODE-RNN model as the encoder and a neural ODE model as the decoder. 2) mTANs [22]: the method utilized an attention mechanism and showed state-of-the-art results for the interpolation of irregularly sampled time series.

Table 6: Description of datasets for time series forecasting.

feature $K$	total time step	history steps $L_1$	prediction steps $L_2$	test sample	epochs
solar	137	10392	168	24	7
electricity	370	5833	168	24	7
traffic	963	7009	168	24	7
taxi	1214	1488	48	24	56
wiki	2000	792	90	30	5

### E.2.3 Datasets and Experiment settings for forecasting in Section 6.3

First we describe the datasets we used. We used five open datasets that are commonly used for evaluating probabilistic time series forecasting. The datasets were preprocessed in Salinas et al. [34] and provided in GluonTS<sup>1</sup>[44]:

- solar [45]: hourly solar power production records of 137 stations in Alabama State.
- electricity<sup>2</sup>: hourly electricity consumption of 370 customers.
- traffic<sup>3</sup>: hourly occupancy rate of 963 San Francisco freeway car lanes.
- taxi<sup>4</sup>: half hourly traffic time series of New York taxi rides taken at 1214 locations in the months of January 2015 for training and January 2016 for test.
- wiki: daily page views of 2000 Wikipedia pages.

We summarize the characteristics of each dataset in Table 6. The task for these datasets is to predict the future  $L_2$  steps by exploiting the latest  $L_1$  steps where  $L_1$  and  $L_2$  depend on datasets as shown in Table 6. We set  $L_1$  and  $L_2$  referring to previous studies [37]. For training, we randomly selected  $L_1 + L_2$  consecutive time steps as one time series and set the last  $L_2$  steps as imputation targets. We followed the train/test split in previous studies. We used the last five samples of training data as validation data.

As for experiment settings, since we basically followed the setting for time series imputation in Section E.2.1, we only describe the difference from it. We ran each experiment three times with different random seeds. We set batch size as 8 because of longer sequence length, and utilized an efficient Transformer as mentioned in Section E.1.

Since the number of features  $K$  is large, we adopted subset sampling of features for training. For each time series in a training batch, we randomly chose a subset of features and only used the subset for the batch. The attention mechanism allows the model to take varying length inputs. We set the subset size as 64. Due to the subset sampling, we need large epochs when the number of features  $K$  is large. Therefore, we set training epochs based on the number of features and the validation loss. We provide the epochs in Table 6.

Finally, we describe the baselines which were used for comparison. 1) GP-copula [34]: the method combines a RNN-based model with a Gaussian copula process to model time-varying correlations. 2) Transformer MAF [36]: the method uses Transformer to learn temporal dynamics and a conditioned normalizing flow to capture feature dependencies. 3) TLAE [37]: the method combines a RNN-based model with autoencoders to learn latent temporal patterns. 4) TimeGrad [25]: the method has shown the state-of-the-art results for probabilistic forecasting by combining a RNN-based model with diffusion models.

### E.3 Computations of CRPS

We describe the definition and computation of the CRPS metric.

The continuous ranked probability score (CRPS) [33] measures the compatibility of an estimated probability distribution  $F$  with an observation  $x$ , and can be defined as the integral of the quantile loss  $\Lambda_\alpha(q, z) := (\alpha - \mathbb{1}_{z < q})(z - q)$  for all quantile levels  $\alpha \in [0, 1]$ :

$$\text{CRPS}(F^{-1}, x) = \int_0^1 2\Lambda_\alpha(F^{-1}(\alpha), x)d\alpha \quad (16)$$

where  $\mathbb{1}$  is the indicator function. We generated 100 samples to approximate the distribution  $F$  over each missing value. We computed quantile losses for discretized quantile levels with 0.05 ticks. Namely, we approximated CRPS with

$$\text{CRPS}(F^{-1}, x) \simeq \sum_{i=1}^{19} 2\Lambda_{i*0.05}(F^{-1}(i * 0.05), x)/19. \quad (17)$$

---

<sup>1</sup><https://github.com/awslabs/gluon-ts>

<sup>2</sup><https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014>

<sup>3</sup><https://archive.ics.uci.edu/ml/datasets/PEMS-SF>

<sup>4</sup><https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data>

Then, we evaluated the following normalized average of CRPS for all features and time steps:

$$\frac{\sum_{k,l} \text{CRPS}(F_{k,l}^{-1}, x_{k,l})}{\sum_{k,l} |x_{k,l}|} \quad (18)$$

where  $k$  and  $l$  indicates features and time steps of imputation targets, respectively.

For probabilistic forecasting, we evaluated CRPS-sum. CRPS-sum is CRPS for the distribution  $F$  of the sum of all  $K$  features and is computed by the following equation:

$$\frac{\sum_l \text{CRPS}(F^{-1}, \sum_k x_{k,l})}{\sum_{k,l} |x_{k,l}|} \quad (19)$$

where  $\sum_k x_{k,l}$  is the sum of forecasting targets for all features at time point  $l$ .

## F Additional results and experiments

Table 7: Comparing the two dimension attention mechanism of various architectures. For ablations, we report the mean and the standard error for three trials.

	healthcare (10% missing)		air quality	
	MAE	CRPS	MAE	CRPS
no-temporal	0.439(0.004)	0.475(0.001)	26.63(0.23)	0.292(0.002)
no-feature	0.352(0.001)	0.386(0.002)	14.44(0.11)	0.162(0.001)
flatten	0.383(0.002)	0.418(0.002)	12.26(0.09)	0.139(0.001)
Bi-RNN	0.272(0.001)	0.301(0.001)	12.56(0.26)	0.142(0.003)
dilated conv	0.279(0.002)	0.305(0.002)	11.67(0.11)	0.130(0.001)
2D attention (proposed)	<b>0.217(0.001)</b>	<b>0.238(0.001)</b>	<b>9.60(0.04)</b>	<b>0.108(0.001)</b>

### F.1 Effectiveness of two dimensional attention mechanism

In this paper, we utilized a two dimensional attention mechanism to learn temporal and feature dependencies. To show the effectiveness of the attention mechanism, we demonstrate an ablation study. We replace the attention mechanism with the following architecture baselines and compare the performance:

- no temporal: remove temporal attention layers
- no feature: remove feature attention layers
- flatten: flatten 2D tensor ( $K$  features x  $L$  length) to 1D, and input the 1D vector to transformer layers
- Bi-RNN: replace the attention mechanism with Bi-directional RNN which is a popular architecture for multivariate time series imputation
- dilated conv: replace temporal and feature attention layers with 1D dilated convolution layers, respectively. The dilated convolution was used in previous studies for diffusion models[13, 25]

We set hyperparameters of each architecture so that the number of parameters is almost the same as our attention mechanism. We show the result in Table 7. Our attention mechanism outperforms all of the other architectures. The comparison with “no temporal” and “no feature” shows that both temporal and feature correlations are important for accurate imputation. The comparison with “flatten”, “Bi-RNN”, and “dilated conv” shows that our attention mechanism is effective to learn temporal and feature dependency compared with existing methods. In summary, the result of the ablation indicates the proposed attention mechanism plays a key role in improving the imputation performance by a large margin.

Table 8: Comparison of the negative log likelihood (NLL) and CRPS for various schedules. We report the mean for three trials.

method	schedule	healthcare (10% missing)		air quality	
		NLL	CRPS	NLL	CRPS
GP-VAE	—	< 1.22	0.574	< 1.09	0.397
proposed	quad. (in paper)	< 1.63	<b>0.238</b>	< 0.97	<b>0.108</b>
proposed	linear	< 29.70	0.240	< 18.55	0.110
proposed	quad. (large min. noise)	< <b>0.07</b>	0.239	< <b>-0.70</b>	0.109

## F.2 Comparison of negative log likelihood for probabilistic imputation

The negative log likelihood (NLL) is a popular metric for evaluating probabilistic methods and ELBO is often utilized to estimate NLL. A reason why we mainly focused on other metrics is that ELBO is sometimes far from NLL and uncorrelated with the quality of generated samples. Specifically, in the proposed method, the choice of the noise schedule highly affects the ELBO while it has little effect on the sample quality.

To demonstrate this, we performed an experiment. We chose the following three noise schedules for CSDI and calculated NLL and CRPS for each schedule.

- quadratic (used in the paper): quadratic spaced schedule between  $\beta_{\min} = 0.0001$  and  $\beta_{\max} = 0.5$
- linear: linear spaced schedule with the same  $\beta_{\min}$  and  $\beta_{\max}$  as those in the paper
- quadratic (large minimum noise): quadratic schedule with large minimum noise level  $\beta_{\min} = 0.001$ , which makes the model ignore small noise

We also calculated the metrics for GP-VAE. The result is shown in Table 8. While CRPS by the proposed method is almost independent from the choice of schedules, NLL significantly depends on the schedule. This phenomenon happens because time series data is generally noisy and it is difficult to denoise small noise during imputation. Estimated scores by the model could be inaccurate when inputs to the model (i.e. imputation targets) only contain small noise. These inaccurate scores could make the estimated ELBO loose, whereas small noise does not affect the sample quality. When the minimum noise level  $\beta_{\min}$  is large, since the model does not denoise small noise in sampling steps, ELBO by the proposed method is tightly estimated and smaller than that by GP-VAE. Therefore, ELBO is not suitable for evaluating the sample quality and we adopted other metrics such as CRPS and MAE.

## F.3 Experimental results for other metrics in Section 6

We show the experimental results in Section 6 for different metrics in Table 9 to 12. Table 9 evaluates RMSE for deterministic imputation methods. We added SSGAN [19] as an additional baseline, which has shown the state-of-the-art performance for RMSE in the healthcare dataset. We can confirm that CSDI outperforms all baselines for RMSE. The advantage of CSDI is particularly large when the missing ratio is low. This result is consistent with that in Section 6.1.

Table 10 evaluates MAE and RMSE for interpolation methods. The result is consistent with Table 4. Table 11 and 12 report CRPS and MSE for probabilistic forecasting methods, respectively. We exclude TimeGrad [25] from the baselines, as they did not report these metrics. We can see that CSDI is competitive with baselines for these metrics as with CRPS-sum.

## F.4 Effect of the number of generated samples

For the experiments in Section 6, we generated 100 samples to estimate the distribution of imputation. We demonstrate the relationship between the number of samples and the performance in Figure 7. We can see that five or ten samples are enough to estimate good distributions and outperform the baselines. Increasing the number of samples further improves the performance, and the improvement becomes marginal over 50 samples.

Table 9: Comparing deterministic imputation methods with CSDI for RMSE. The results correspond to Table 3. We report the mean and the standard error for five trials. The asterisk means the values are cited from the original paper.

	healthcare			air quality
	10% missing	50% missing	90% missing	
V-RIN [32]	0.628(0.025)	0.693(0.022)	0.928(0.013)	40.11(1.14)
BRITS [7]	0.619(0.022)	0.693(0.023)	0.836(0.015)	24.47(0.73)
RDIS [20]	0.633(0.021)	0.741(0.018)	0.934(0.013)	37.49(0.28)
SSGAN [19] (*)	0.598	0.762	0.818	—
unconditional	0.621(0.020)	0.734(0.024)	0.940(0.018)	22.58(0.23)
<b>CSDI</b> (proposed)	<b>0.498(0.020)</b>	<b>0.614(0.017)</b>	<b>0.803(0.012)</b>	<b>19.30(0.13)</b>

Table 10: Comparing the state-of-the-art interpolation method with CSDI for MAE and RMSE. The results correspond to Table 4. We report the mean and the standard error for five trials.

		10% missing	50% missing	90% missing
MAE	Latent ODE [35]	0.522(0.002)	0.506(0.003)	0.578(0.009)
	mTANs [22]	0.389(0.003)	0.422(0.003)	0.533(0.005)
	<b>CSDI</b> (proposed)	<b>0.362(0.001)</b>	<b>0.394(0.002)</b>	<b>0.518(0.003)</b>
RMSE	Latent ODE [35]	0.799(0.012)	0.783(0.012)	0.865(0.017)
	mTANs [22]	0.749(0.037)	0.721(0.014)	<b>0.836(0.018)</b>
	<b>CSDI</b> (proposed)	<b>0.722(0.043)</b>	<b>0.700(0.013)</b>	0.839(0.009)

Table 11: Comparing probabilistic forecasting methods with CSDI for CRPS. The results correspond to Table 5. We report the mean and the standard error for three trials. The results for baseline methods are cited from their paper. 'TransMAF' is the abbreviation for 'Transformer MAF'.

	solar	electricity	traffic	taxi	wiki
GP-copula [34]	0.371(0.022)	0.056(0.002)	0.133(0.001)	0.360(0.201)	0.236(0.000)
TransMAF [36]	0.368(0.001)	0.052(0.000)	0.134(0.001)	0.377(0.002)	0.274(0.007)
TLAE [37]	<b>0.335(0.025)</b>	0.058(0.002)	0.097(0.001)	0.369(0.006)	0.298(0.001)
<b>CSDI</b> (proposed)	0.338(0.012)	<b>0.041(0.000)</b>	<b>0.073(0.000)</b>	<b>0.271(0.001)</b>	<b>0.207(0.002)</b>

Table 12: Comparing probabilistic forecasting methods with CSDI for MSE. The results correspond to Table 5. We report the mean and the standard error for three trials. The results for baseline methods are cited from their paper. 'TransMAF' is the abbreviation for 'Transformer MAF'. 'TransMAF' did not report the standard error.

	solar	electricity	traffic	taxi	wiki
GP-copula [34]	9.8e2(5.2e1)	2.4e5(5.5e4)	6.9e-4(2.2e-5)	3.1e1(1.4e0)	4.0e7(1.6e9)
TransMAF [36]	9.3e2	2.0e5	5.0e-4	4.5e1	<b>3.1e7</b>
TLAE [37]	<b>6.8e2(7.5e1)</b>	2.0e5(9.2e4)	4.0e-4(2.9e-6)	2.6e1(8.1e-1)	3.8e7(4.2e4)
<b>CSDI</b> (proposed)	9.0e2(6.1e1)	<b>1.1e5(2.8e3)</b>	<b>3.5e-4(7.0e-7)</b>	<b>1.7e1(6.8e-2)</b>	3.5e7(4.4e4)

Table 13: The effect of the target choice strategy for the air quality dataset. We report the mean and the standard error for five trials.

	CRPS	MAE
random	<b>0.108(0.001)</b>	<b>9.58(0.08)</b>
historical	0.113(0.001)	10.12(0.05)
mix	<b>0.108(0.001)</b>	9.60(0.04)

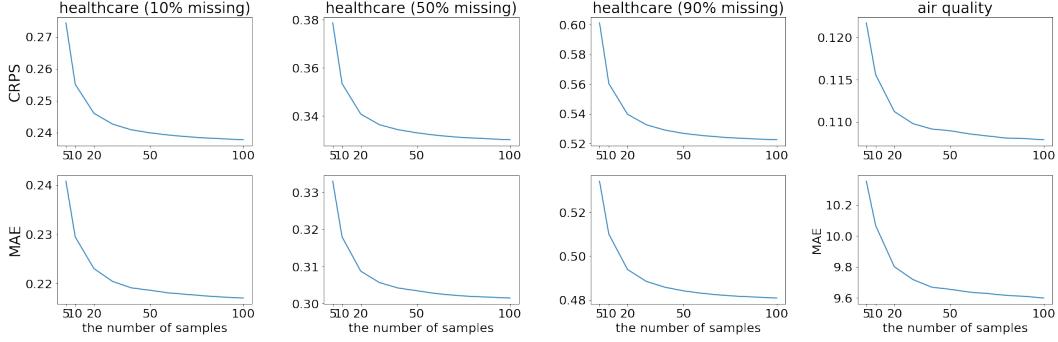


Figure 7: The effect of the number of generated samples. The first row shows the effect on probabilistic imputation in Table 2 and the second row shows the effect on deterministic imputation in Table 3.

### E.5 Effect of target choice strategy

In the experiment for the air quality dataset in Section 6.1, we adopted the mix strategy for the target choice. Here, we provide the result for other strategies and show the effect of the target choice strategy on imputation quality. In Table 13, the performances of the mix strategy and the random strategy are almost the same, and the performance of the historical strategy is slightly worse than that of the other strategies. This means that the historical strategy is not effective for the air quality dataset even though the dataset contains structured missing patterns. This is due to the difference of missing patterns between training dataset and test dataset. Note that all strategies outperform the baselines in Table 2 and Table 3.

## G Additional examples of probabilistic imputation

In this section, we illustrate various imputation examples to show the characteristic of imputed samples. We pick a multivariate time series from the results of each experiment in Section 6.1 and show imputation results for all features of each time series in Figure 8 to 11. We compare CSDI with GP-VAE in Figure 8 to 11. Note that the scales of the  $y$  axis depend on the features. For the healthcare dataset with 90% missing ratio in Figure 10, while GP-VAE fails to learn the distribution, CSDI gives reasonable probabilistic imputation for most of the features. For the air quality dataset in Figure 11, CSDI learns the dependency between features and provides more accurate imputation than GP-VAE. In Figure 12 to 15, we compare CSDI with the unconditional diffusion model. In all figures, CSDI tends to provide tighter uncertainty than the unconditional diffusion model. We hypothesize that it is due to the approximation discussed in Section 3.3. Since the unconditional model approximates the conditional distribution by using noisy observed values, the estimated imputation become less confident than that with the conditional model.

## H Potential negative societal impacts

Since score-based diffusion models are generative models, our proposed model has negative impacts as well as other generative models. For example, the model can potentially memorize private information and be used to generate fake data.



Figure 8: Comparison of imputation between GP-VAE and CSDI for the healthcare dataset (10% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and GP-VAE, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.



Figure 9: Comparison of imputation between GP-VAE and CSDI for the healthcare dataset (50% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and GP-VAE, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

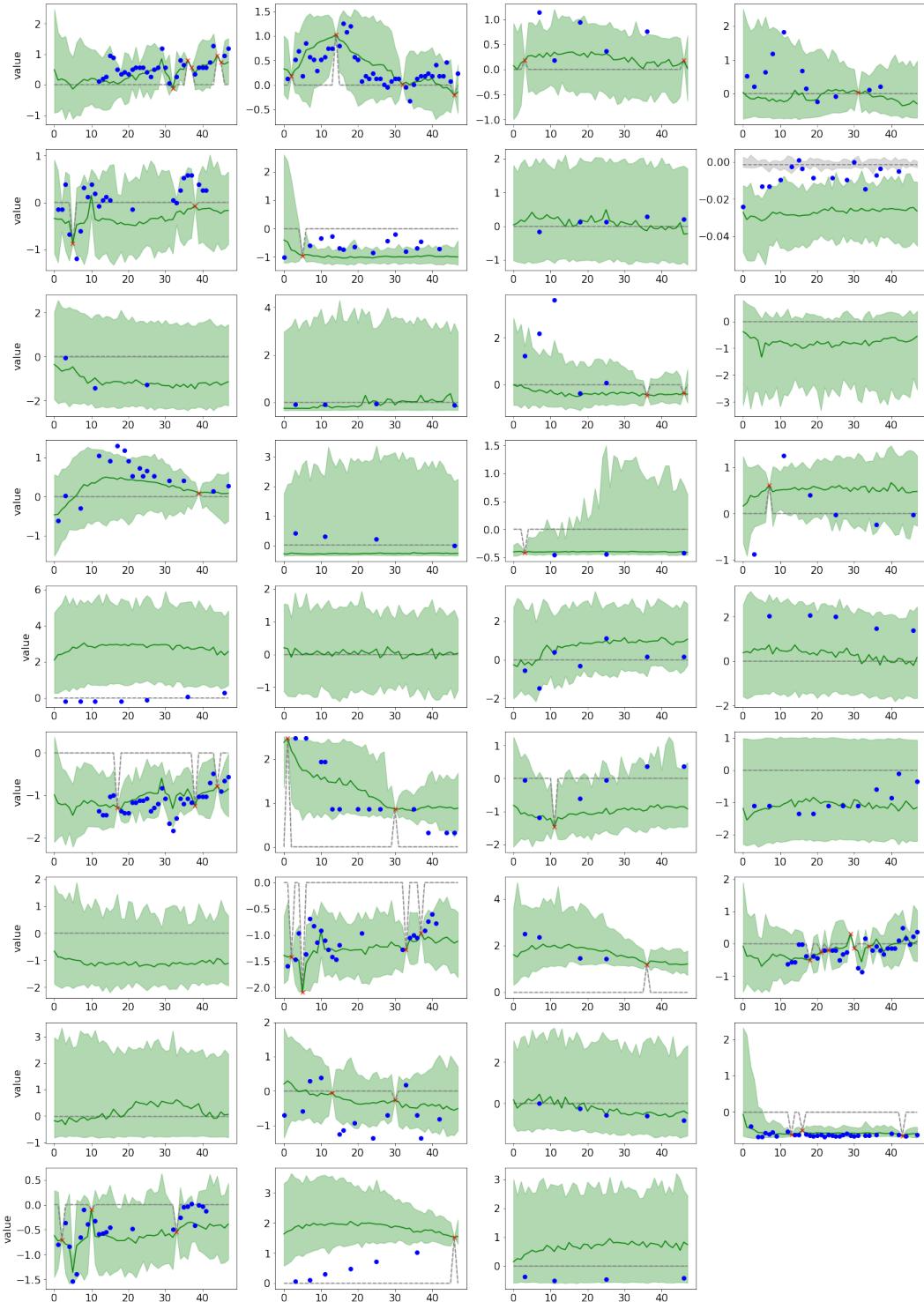


Figure 10: Comparison of imputation between GP-VAE and CSDI for the healthcare dataset (90% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and GP-VAE, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

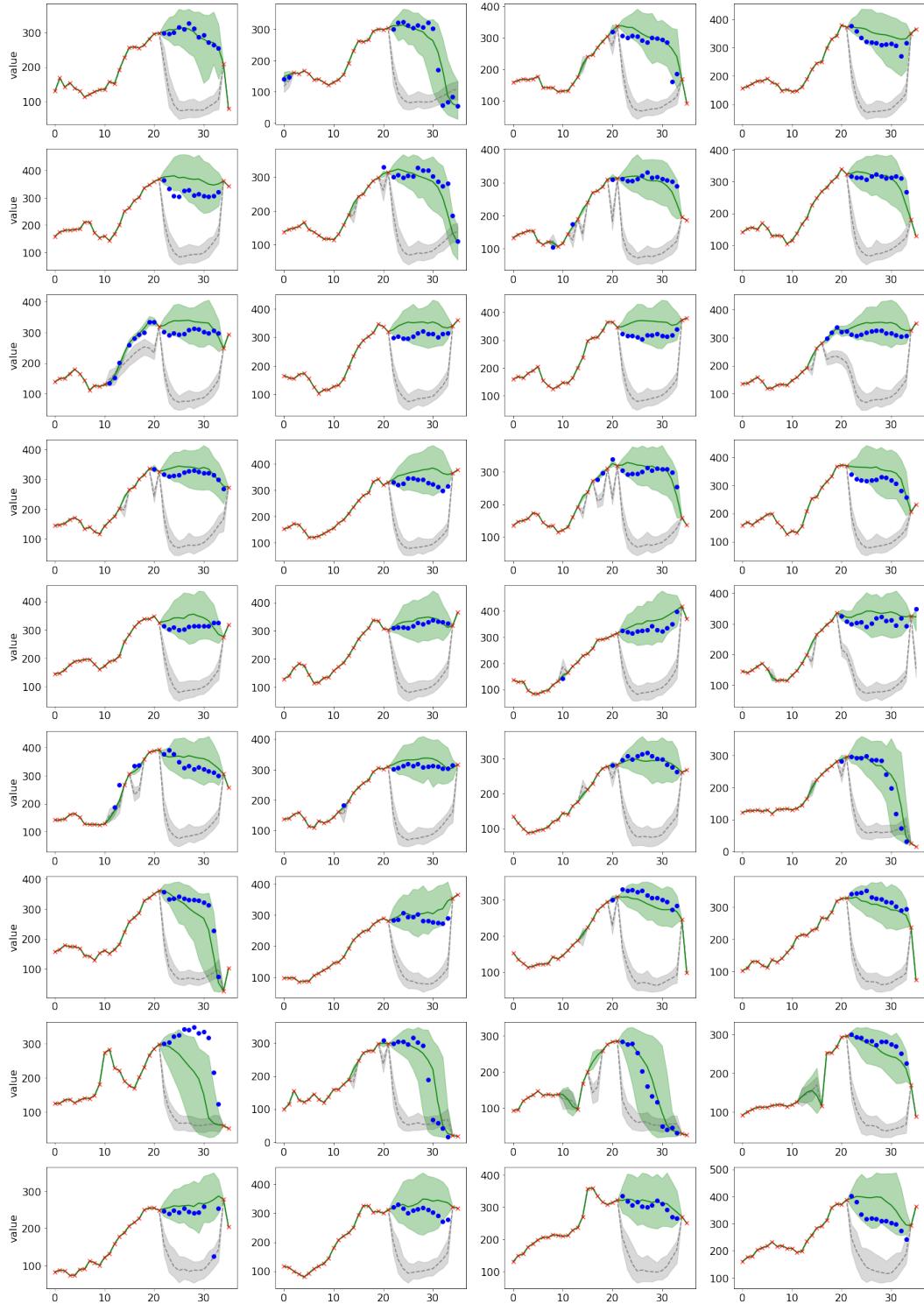


Figure 11: Comparison of imputation between GP-VAE and CSDI for the air quality dataset. The result is for a time series sample with all 36 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and GP-VAE, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.



Figure 12: Comparison of imputation between the unconditional diffusion model and CSDI for the healthcare dataset (10% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and the unconditional model, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

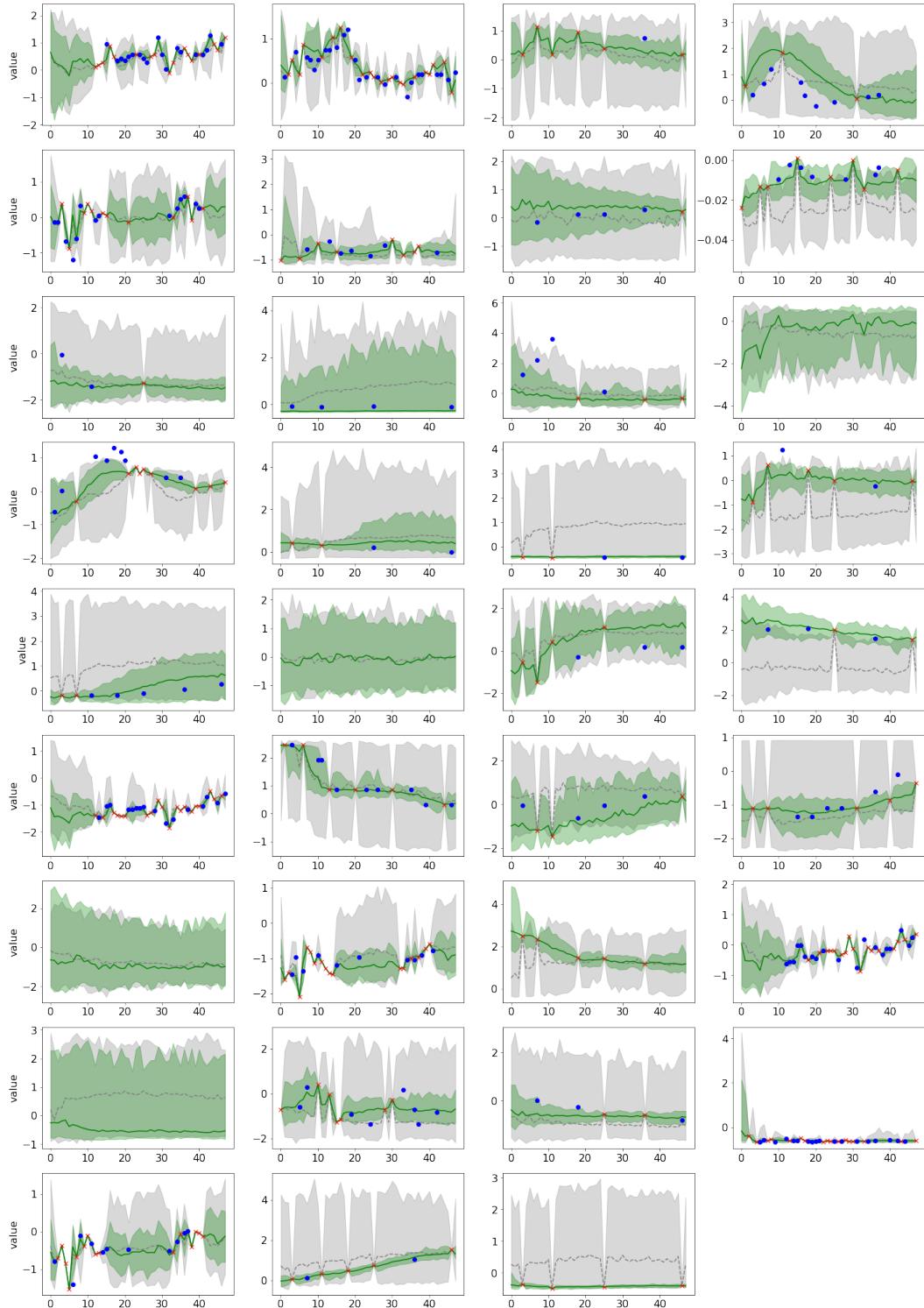


Figure 13: Comparison of imputation between the unconditional diffusion model and CSDI for the healthcare dataset (50% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and the unconditional model, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

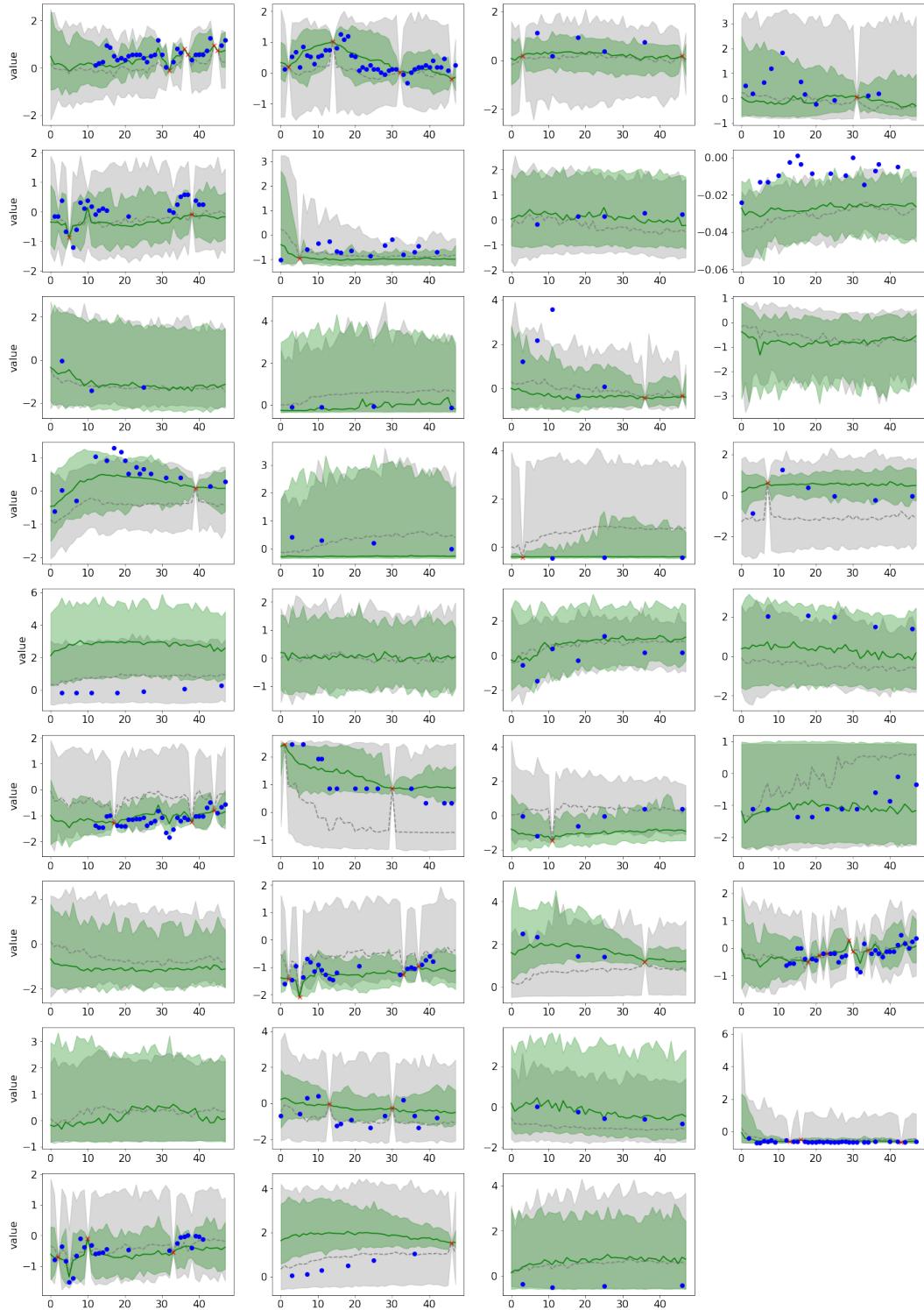


Figure 14: Comparison of imputation between the unconditional diffusion model and CSDI for the healthcare dataset (90% missing). The result is for a time series sample with all 35 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and the unconditional model, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.

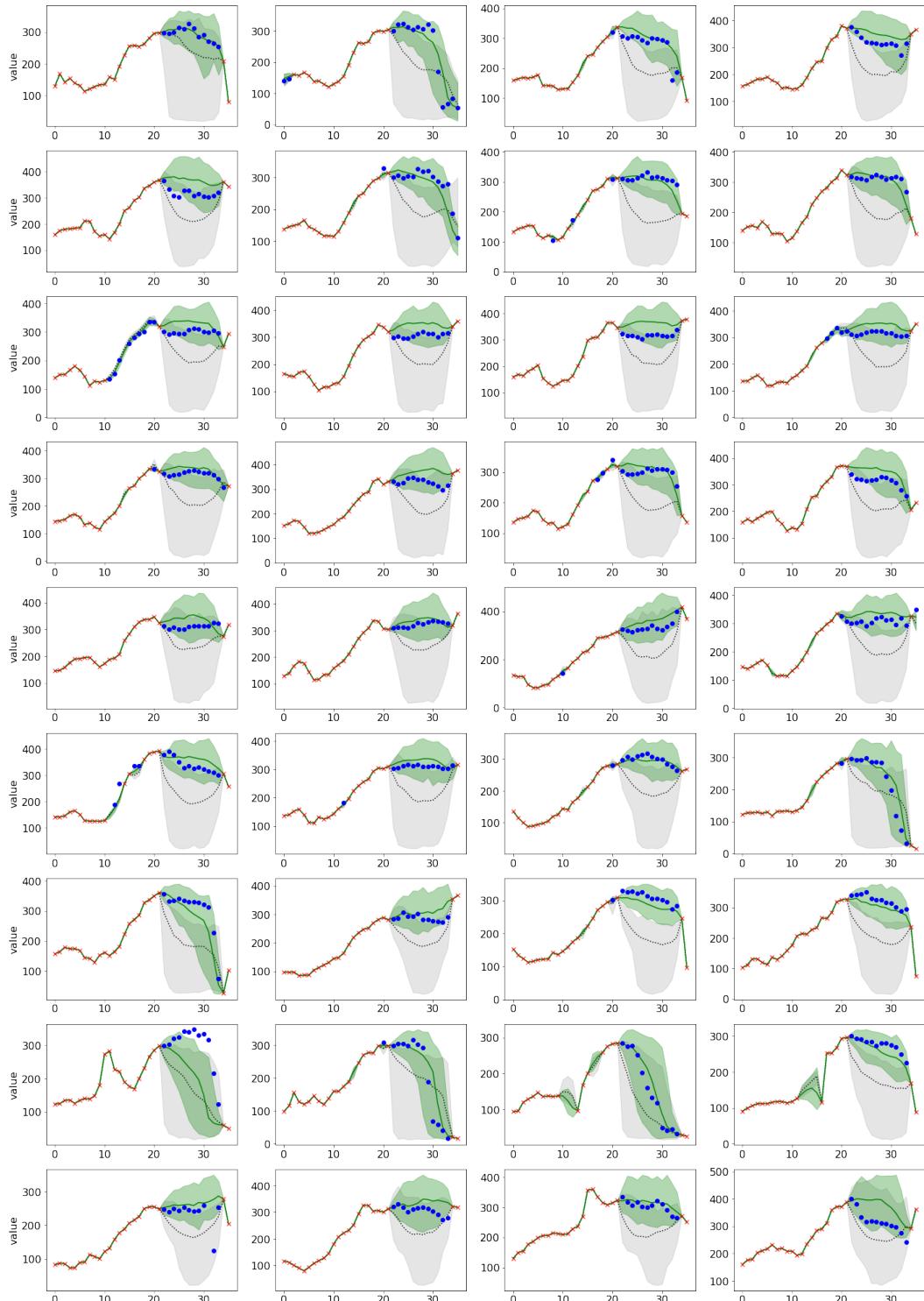


Figure 15: Comparison of imputation between the unconditional diffusion model and CSDI for the air quality dataset. The result is for a time series sample with all 36 features. The red crosses show observed values and the blue circles show ground-truth imputation targets. Green and gray colors correspond to CSDI and the unconditional model, respectively. For each method, median values of imputations are shown as the line and 5% and 95% quantiles are shown as the shade.