Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models

Anonymous authors
Paper under double-blind review

Abstract

The imputation of missing values represents a significant obstacle for many real-world data analysis pipelines. Here, we focus on time series data and put forward SSSD, an imputation model that relies on two emerging technologies, (conditional) diffusion models as state-of-the-art generative models and structured state space models as internal model architecture, which are particularly suited to capture long-term dependencies in time series data. We demonstrate that SSSD matches or even exceeds state-of-the-art probabilistic imputation and forecasting performance on a broad range of data sets and different missingness scenarios, including the challenging blackout-missing scenarios, where prior approaches failed to provide meaningful results.

1 Introduction

Missing input data is a common phenomenon in real-world machine learning applications, which can have many different reasons, ranging from inadequate data entry over equipment failures to file losses. Handling missing input data represents a major challenge for machine learning applications as most algorithms require data without missing values to train. Unfortunately, the imputation quality has a critical impact on downstream tasks, as demonstrated in prior work Shadbahr et al. (2022), and poor imputations can even introduce bias into the downstream analysis Zhang et al. (2022), which can potentially call into question the validity of the results achieved in them.

In this work, we focus on time series as a data modality, where missing data is particularly prevalent, for example, due to faulty sensor equipment. We consider a range of different missingness scenarios, see Figure 1 for a visual overview, where the example of faulty sensor equipment former example already suggests that not-at-random missingness scenarios are significant for real-world scenarios. Time series forecasting is naturally contained in this approach as special case of blackout missingness, where the location of the imputation window is at the end of the sequence. We also stress that the most realistic scenario to address imputation as an underspecified problem class is the use of probabilistic imputation methods, which do not provide only a single imputation but instead allow samples of different plausible imputations.

There is a large body of literature on time series imputation, see Osman et al. (2018) for a review, ranging from statistical methods Lin & Tsai (2020) to autoregressive models Atyabi et al. (2016); Bashir & Wei (2018). Recently, deep generative models started to emerge as a promising paradigm to model time series imputation of long sequences or time series forecasting problems at long horizons. However, many existing models remain limited to the random missing scenario or show unstable behavior during training. In addition, we demonstrate that state-of-the-art approaches even fail to deliver qualitatively meaningful imputations in blackout missing scenarios on certain data sets.

In this work, we aim to address these shortcomings by proposing a new generative-model-based approach for time series imputation. We use diffusion models as the current state-of-the-art in terms of generative modeling in different data modalities. The second principal component of our approach is the use of structured state-space models Gu et al. (2022a) instead of dilated convolutions or transformer layers as main computational building block of the model, which are particularly suited to handling long-term-dependencies in time series data.

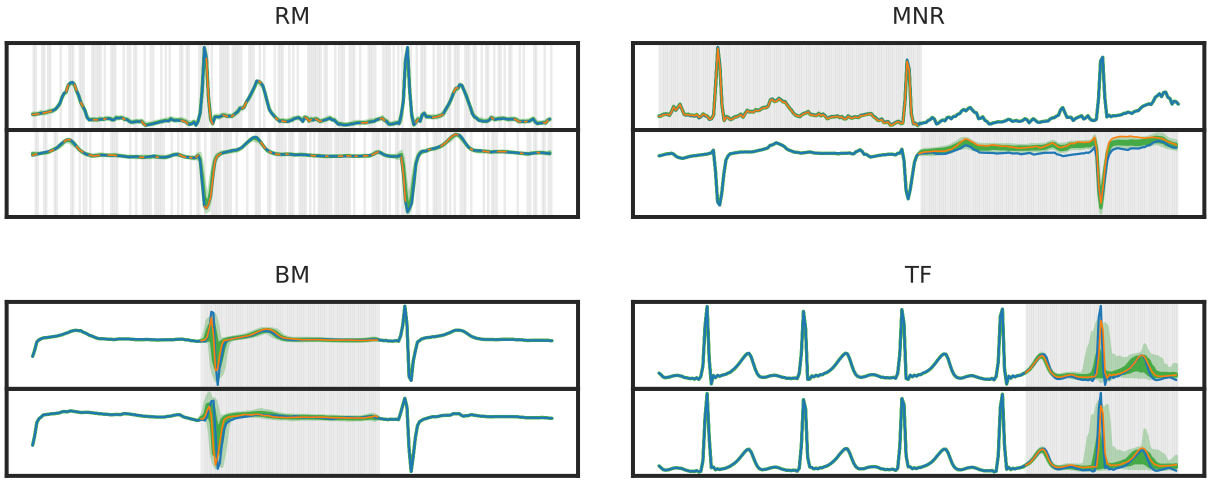

Figure 1: Color scheme introduction. The proposed model SSSD^{S4} provides imputations for different missingness scenarios that are not only quantitatively but even qualitatively superior, see below, on different data sets for different missingness scenarios (RM: random missing, MNR: missing not at random, BM: blackout missing TF: time series forecasting). Similarly, The signal is in blue, where the white background represents the conditioned ground truth, whereas the gray background represents the target signal. Prediction bands derived from 100 imputations represent quantiles from 0.05 to 0.95 as lighter shaded green and from 0.25 to 0.75 as darker shaded green. As these bands do not allow visually assessing the quality of individual imputations, we always additionally show a randomly selected single sample in orange.

To summarize, our main contributions are as follows: (1) We put forward a combination of state-space models as ideal building blocks to capture long-term dependencies in time series with (conditional) diffusion models as the current state-of-the-art technology for generative modeling . (2) We suggest modifications to the contemporary diffusion model architecture DiffWaveKong et al. (2021) to enhance its capability for time series modeling. In addition, we propose a simple yet powerful methodology in which the noise of the diffusion process is introduced just to the regions to be imputed, which turns out to be superior to approaches proposed in the context of image inpainting Lugmayr et al. (2022b). (3) We provide extensive experimental evidence for the superiority of the proposed approach compared to state-of-the-art approaches on different data sets for various missingness approaches, particularly for the most challenging blackout and forecasting scenarios.

2 Structured state space diffusion (SSSD) models for time series imputation

Time series imputation Let x_0 be a data sample with shape $\mathbb{R}^{L\times K}$, where L represents the number of time steps, and K represents the number of features or channels. Imputation targets are then typically specified in terms of binary masks that match the shape of the input data, i.e., $m_{\text{imp}} \in [0,1]^{L\times K}$, where the values to be conditioned on are marked by ones and zeros denote values to be imputed. In the case where there are also missing values in the input, one additionally requires a mask m_{mvi} of the same shape to distinguish values that are present in the input data (1) from those that are not (0).

In the literature, see, e.g., Lin & Tsai (2020), one distinguishes different missingness scenarios, which we review here for completeness: We define *random missing (RM)* as a situation where the zero-entries of an imputation mask are sampled randomly according to a uniform distribution across all channels of the whole input sample. Secondly, *missing not at random (MNR)* assumes that a subset of consecutive time steps is missing in a given channel, for one channel at a time. Finally, *blackout missing (BM)* assumes that there is subset of time steps where data is missing across all channels. As mentioned above, *time series forecasting (TF)* can be seen as a particular case of BM imputation, where the imputation region spans a consecutive region of t time steps, where t denotes the forecasting horizon across all channels located at the very end of the sequence.

Diffusion models Diffusion models Sohl-Dickstein et al. (2015) represent a class of generative models that demonstrated state-of-the-art performance on a range of different data modalities, from image Dhariwal & Nichol (2021); Ho et al. (2020; 2022a) over speech Chen et al. (2020); Kong et al. (2021) to video data Ho et al. (2022b).

Diffusion models learn a mapping from a latent space to the original signal space by learning to remove noise in a backward process that was added sequentially in a Markovian fashion during a so-called forward process. These two processes, therefore, represent the backbone of the diffusion model. For simplicity, we restrict ourselves to the unconditional case at the beginning of this section and discuss modifications for the conditional case further below. The forward process is parameterized as

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

where $q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t} x_{t-1}, \beta_t \mathbb{1})[x_t]$ and the (fixed or learnable) forward-process variances β_t adjust the noise level. Equivalently, x_t can be expressed in closed form as $x_t = \sqrt{\alpha_t} x_0 + (1 - \alpha_t)\epsilon$ for $\epsilon \sim \mathcal{N}(0, \mathbb{1})$, where $\alpha_t = \sum_{i=1}^{t}(1 - \beta_t)$.

The backward process is parameterized as

$$p_\theta(x_0, \ldots, x_{t-1} | x_T) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1} | x_t) \tag{2}$$

where $x_T \sim \mathcal{N}(0, \mathbb{1})$. Again, $p_\theta(x_{t-1}|x_t)$ is assumed as normal-distributed (with diagonal covariance matrix) with learnable parameters. Using a particular parametrization of $p_\theta(x_{t-1}|x_t)$, Ho et al. (2020) showed that the reverse process can be trained using the following objective,

$$L = \min_\theta \mathbb{E}_{x_0 \sim \mathcal{D}, \epsilon \sim \mathcal{N}(0, \mathbb{1}), t \sim \mathcal{U}(1,T)} ||\epsilon - \epsilon_\theta(\sqrt{\alpha_t} x_0 + (1 - \alpha_t)\epsilon, t)||_2^2 \,, \tag{3}$$

where $\mathcal{D}$ refers to the data distribution and $\epsilon_\theta(x_t, t)$ is parameterized using a neural network. This objective can be seen as a weighted variational bound on the negative log-likelihood that down-weights the importance of terms at small t, i.e., at small noise levels.

Extending the unconditional diffusion process described so far, one can consider conditional variants where the backward process is conditioned on additional information, i.e. $\epsilon_\theta = \epsilon_\theta(x_t, t, c)$, where the precise nature of the conditioning information c depends on the application at hand and ranges from global to local information such as spectrograms Kong et al. (2021). In our case, it is given by the concatenation of input (masked according to the imputation mask) and the imputation mask itself, i.e., $c = \text{Concat}(x_0 \odot (m_{\text{imp}} \odot m_{\text{mvi}}), (m_{\text{imp}} \odot m_{\text{mvi}})$, where $\odot$ denotes point-wise multiplication. In this work, we consider two different setups, denoted as D_0 and D_1, respectively, where we apply the diffusion process to the full signal or to the regions to be imputed only. In any case, the evaluation of the loss function in table 3 is only supposed to be on the input values for which ground truth information is available, i.e., where $m_{\text{mvi}} = 1$. For D_0, this can be seen as a reconstruction loss for the input values corresponding to non-zero portions of the imputation mask (where conditioning is available) and an imputation loss corresponding to input tokens at which the imputation mask vanishes, c.f. also Du et al. (2022). For D_1, the reconstruction loss vanishes by construction. Finally, we also investigate an approach using a model trained in an unconditional fashion, where the conditional information is only included during inference Lugmayr et al. (2022b).

At this point, it is appropriate to compare our proposed approach to Tashiro et al. (2021), as the only other diffusion-based imputer in the literature. As in Tashiro et al. (2021), we base our parametrization of $\epsilon_\theta(x_t, t, c)$ on the DiffWave architecture Kong et al. (2021) as a versatile diffusion model architecture proposed in the context of audio generation. However, for the diffusion mechanism we do not work with an extended four-dimensional internal representation of shape (batch dimension, diffusion dimension, input channel dimension, time dimension), which necessitates processing time and feature dimensions alternatively since many modern architectures for sequential data, such as transformers or structured state space models, to be discussed below, are only able to process sequential, i.e. three-dimensional, input batches. We take the conceptionally simpler path of mapping the input channels into the diffusion dimension and performing only diffusion along the time dimension, i.e. processing batches of shape (batch dimension, diffusion dimension, time dimension). Additionally, we modify the internals of the DiffWave architecture using S4 layers Gu et al. (2022a) that are better suited to processing time series data than the dilated convolutions used in the original architecture. We postpone a detailed discussion of the model architecture to the proposed approaches section below.

State space models The recently introduced structured state-space model (SSM) Gu et al. (2022a) represents a promising modeling paradigm to efficiently capture long-term dependencies in time series data. At its heart, the formalism draws on a linear state space transition equation, connecting a one-dimensional input sequence $u(t)$ to a one-dimensional output sequence $y(t)$ via a N-dimensional hidden state $x(t)$. Explicitly, this transition equation reads

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

where A, B, C, D are transition matrices. After discretization, the relation between input and output can be written as a convolution operation that can be evaluated efficiently on modern GPUs Gu et al. (2022a). The ability to capture long-term dependencies relates to a particular initialization of $A \in \mathbb{R}^{N\times N}$ according to HiPPO theory Gu et al. (2020; 2022b). In Gu et al. (2022a), the authors put forward a Structured State Space sequence model (S4) by stacking several copies of the above SSM blocks with appropriate normalization layers and point-wise fully-connected layers in the style of a transformer layer, demonstrating excellent performance on various sequence classification tasks. In fact, the resulting S4 layer parametrizes a shape-preserving mapping of data with shape (batch, model dimension, length dimension) and can therefore be used as a drop-in replacement for transformer, RNN, or one-dimensional convolution layers (with appropriate padding). Building on the S4 layer, the authors presented SaShiMi, a generative model architecture for sequence generation Goel et al. (2022) obtained by combining S4 layers in a U-net-inspired configuration. While the model was proposed as an autoregressive model, the authors already pointed out the ability to use the (non-causal) SaShiMi as a component in state-of-the-art non-autoregressive models such as DiffWave Kong et al. (2021).

Proposed approaches We propose and investigate a number of different diffusion imputer architectures. It is worth stressing that there is no prior work on direct applications of conditional diffusion models for time series imputation except for Tashiro et al. (2021), which is based on DiffWave, which was proposed in the context of speech synthesis. Thus, we propose as a baseline a direct adaptation of the DiffWave-architecture Kong et al. (2021) for time series imputation, see the technical appendix more technical details, but in particular, we use the proposed setup where the diffusion process is only applied to the parts of the sequence that are supposed to imputed. Based on this model, we put forward SSSD^{S4}, the main architectural contribution of this work, where instead of bidirectional dilated convolutions, we used S4 layer as a diffusion layer within each of its residual blocks after adding the diffusion embedding. As a second modification, we include a second S4 layer after the addition assignment with the conditional information, which gives the model additional flexibility after joining processed inputs and the conditional information. The effectiveness of this modification is demonstrated in an ablation study in the technical appendix. The architecture is depicted schematically in Figure 2. Second, under the name SSSD^{SA} we explore an extension of the non-autoregressive of the SaShiMi architecture for time series imputation through appropriate conditioning. Third, we investigate CSDI^{S4}, a modification of the CSDI Tashiro et al. (2021) architecture, where we replace the transformer layer operating in the time direction by an S4 model. In this way, we aim to assess potential improvements of

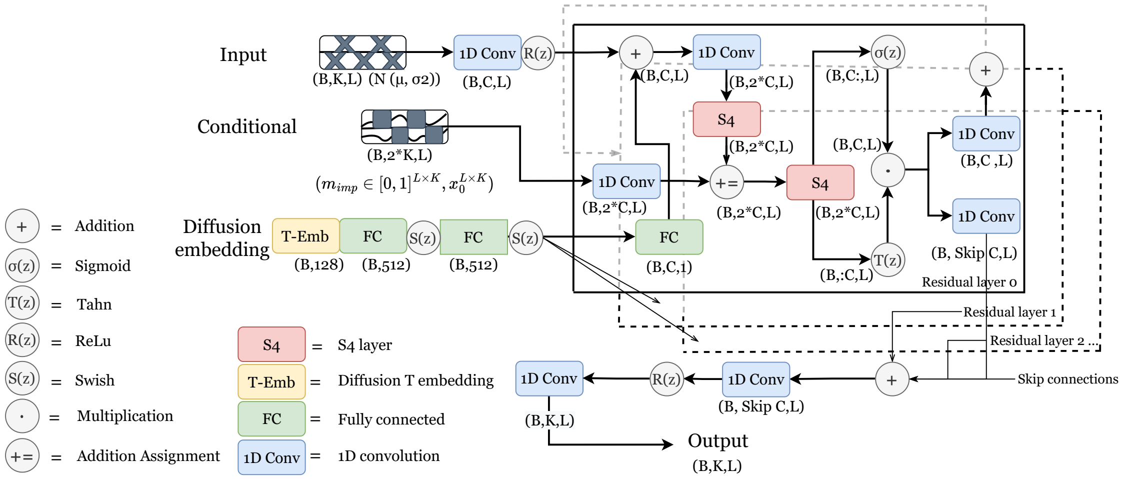

Figure 2: Proposed SSSD$^{\text{S4}}$ model architecture.

an architecture that is more adapted to the domain of time series. A more detailed discussion of the model internals including hyperparameter settings can be found in the technical appendix.

3 Related Work

Deep-learning based time series imputation Time series imputation is a very rich topic. A complete discussion– even of the deep learning literature alone– is clearly beyond the scope of this work, and we refer the reader to a recent review on the topic Fang & Wang (2020). Deep-learning-based time series imputation methods can be broadly categorized based on the technology used: (1) RNN-based approaches such as BRITS Cao et al. (2018), GRU-D Che et al. (2016), NAOMI Liu et al. (2019), and M-RNN Yoon et al. (2019) use single or multi-directional RNNs to model the time series. However, these algorithms suffer of diverse training limitation, and as pointed out in recent works, many of them might only show sub-optimal performance across diverse missing scenarios and at different missing ratios Cini et al. (2022) Du et al. (2022). (2) Generative models represent the second dominant approach in the field. This includes GAN-based approaches such as E^2-GAN Luo et al. (2019), and GRUI-GAN Luo et al. (2018) or VAE-based approaches such as GP-VAE Fortuin et al. (2020). Many of these were found to suffer from unstable training and failed to reach state-of-the-art performance Du et al. (2022). Recently, also diffusion models such as CSDI Tashiro et al. (2021), the closest competitor to our work, were explored with very strong results. (3) Finally, there is a collection of approaches relying in modern architectures such as graph neural networks (GRIN) Cini et al. (2022), and permutation equivariant networks (NRTSI) Shan et al. (2021), *self-attention* to capture temporal and feature correlations (SAITS) Du et al. (2022), controlled differential equations networks (NeuralCDE) Morrill et al. (2021) and ordinal differential equations networks (Latent-ODE) Rubanova et al. (2019).

Conditional generative modeling with diffusion models Diffusion models have been used for related tasks such as inpainting, in particular in the image domain Saharia et al. (2021); Lugmayr et al. (2022a). With appropriate modifications, such methods from the image domain are also directly applicable in the time series domain. Sound is a very special time series, and diffusion models such as DiffWave Kong et al. (2021) conditioned on different global labels or Mel spectrograms showed excellent performance in different speech generation tasks. Returning to general time series, as already discussed above, the closest competitor is CSDI Tashiro et al. (2021). CSDI and this work represent diffusion models and can be seen as DiffWave-variants. The main differences between the two approaches are (1) using SSMs instead of transformers (2) the conceptually more straightforward setup of a diffusion process in time direction only as opposed to feature and time direction (3) a different training objective to denoise just the segments to be imputed (D_1). In all of our experiments, we compare to CSDI to demonstrate our approach's superiority and that exchanging single components (as in CSDI$^{\text{S4}}$) is not sufficient to reach a qualitative improvement of the sample quality in certain BM scenarios.

Time series forecasting The literature on time series forecasting is even richer than the literature on time series imputation. One large body of works includes recurrent architectures, such as LSTNet Lai et al. (2018), and LSTMa Bahdanau et al. (2015). Recently, also modern transformer-based architectures with encoder-decoder design such as Autoformer Wu et al. (2021b), and Informer Zhou et al. (2021), showed excellent performance on long-sequence forecasting. The range of methods that have been applied to this task is very diverse and includes for example GP-Copula Salinas et al. (2019), Transformer MAF Rasul et al. (2021b), and TLAE Nguyen & Quanz (2021). Finally, with Time-Grad Rasul et al. (2021a) there is another diffusion model that showed good performance on various forecasting tasks. However, due to its autoregressive nature its domain of applicability cannot be straightforwardly extended to include time series imputation.

4 Experiments

4.1 Experimental protocol

As already discussed above, we do not keep the (input) channel dimension as an explicit dimension during the diffusion process but only keep it implicitly by mapping the channel dimension to the diffusion dimension. This is inspired by the original DiffWave approach, which was designed for single-channel audio data. As we will demonstrate below, this approach leads to outstanding results in scenarios where the number of input channels remains limited to less than about 100 input channels, which covers for example typical single-patient sensor data such as electrocardiogram (ECG) or electroencephalogram (EEG) in the healthcare domain. For more input channels, the model often shows convergence issues and one has to resort to different training strategies, e.g., by splitting the input channels. As we will also demonstrate below, this approach without any further modifications already leads to competitive results on data sets with a few hundred input channels, but can certainly be improved by more elaborate procedures and is therefore not within the main scope of this work. For this reason also the mains experimental evaluation of our work focuses primarily on data sets with less than 100 input channels.

Throughout this section, we always train and evaluate imputation models on identical missingness scenarios and ratios, e.g., we train on 20% RM and evaluate based on the same setting. The training on the models was performed on single NVIDIA A30 cards. In our experiment, we cover a very broad spectrum of popular data sets both for imputation and forecasting along with a corresponding diverse selection of baseline methods from different communities, to demonstrate the robustness of our proposed approach. There are diverse performance metrics utilized in this work (where in all cases, lower scores signify better imputation results), most of them involve comparing single imputations to the ground truth, others, incorporate imputations distribution and are therefore specific to probabilistic imputers. We refer to our technical appendix for a discussion on them. Similarly, we present small and concise tables in the main text to support our claims. We refer to the technical appendix for more results, including additional baselines, further details on data sets, and preprocessing procedures.

4.2 Time series imputation

SSSD$^{\text{S4}}$outperforms state-of-the-art imputers on ECG data As first data set, we consider ECG data from the *PTB-XL data set* Wagner et al. (2020a;b); Goldberger et al. (2000). ECG data represents an interesting benchmark case as producing coherent imputations beyond the random missing scenario requires to capture the consistent periodic structure of the signal across several beats. We preprocessed the ECG signals at a sampling rate of 100 Hz and considered $L = 250$ time steps (248 in the case of SSSD$^{\text{SA}}$). We considered three different missingness scenarios, RM, MNR, and BM. We present the investigated diffusion model variants, such as and a conditional adaptation of the original DiffWave model, CSDI$^{\text{S4}}$, SSSD$^{\text{SA}}$, and

SSSD^{S4}, where applicable both for training objectives D_0 and D_1. As baselines we consider the deterministic LAMC Chen et al. (2021) and CSDI Tashiro et al. (2021) as a strong probabilistic baseline. We report averaged MAE, RMSE for 100 samples generated for each sample in the test set.

Table 1: Imputation for RM, MNR and BM scenarios on the PTB-XL data set (extracts, see Table 20 for further baseline and settings comparisons).

Model	**MAE**	**RMSE**
	20% RM on PTB-XL	
LAMC	0.0678	0.1309
CSDI	0.0038±2e-6	0.0189±5e-5
DiffWave	0.0043±4e-4	0.0177±4e-4
CSDI^{S4}	**0.0031±1e-7**	0.0171±6e-4
SSSD^{SA}	0.0045±3e-7	0.0181±4e-6
SSSD^{S4}	0.0034±4e-6	**0.0119±1e-4**
	20% MNR on PTB-XL	
LAMC	0.0759	0.1498
CSDI	0.0186±1e-5	0.0435±2e-4
DiffWave	0.0250±1e-3	0.0808±5e-3
CSDI^{S4}	0.0222±2e-5	0.0573±1e-3
SSSD^{SA}	0.0170±1e-4	0.0492±1e-2
SSSD^{S4}	**0.0103±3e-3**	**0.0226±9e-4**
	20% BM on PTB-XL	
LAMC	0.0840	0.1171
CSDI	0.1054±4e-5	0.2254±7e-5
DiffWave	0.0451±7e-4	0.1378±5e-3
CSDI^{S4}	0.0792±2e-4	0.1879±1e-4
SSSD^{SA}	0.0435±3e-3	0.1167±1e-2
SSSD^{S4}	**0.0324±3e-3**	**0.0832±8e-3**

Across all model types and missingness scenarios, applying the diffusion process to the portions of the sample to be imputed (D_1) consistently yields better results than the diffusion process applied to the entire sample(D_0). Also the unconditional training approach from Lugmayr et al. (2022b) proposed in the context of a state-of-the-art diffusion-based method for image inpainting, lead to clearly inferior performance, see Appendix 19. In the following, we will therefore restrict ourselves to the D_1 setting. The proposed SSSD^{S4} outperforms the rest of the imputer models by a significant margin in most scenarios, in particular for BM, where we find a reduction in MAE of more than 50% compared to CSDI. Similarly, we note that DiffWave as a baseline shows very strong results, which are in some scenarios on par with the technically more advanced SSSD^{SA}, nevertheless, the proposed SSSD^{S4} demonstrate a clear improvement for time series imputation and generation across all the settings. Also, there is a clear improvement that the S4 layer on CSDI^{S4} provides to CSDI across RM and BM settings, especially for the RM scenario where CSDI^{S4} outperforms the rest of methods with lower MAE. We hypothesize that the CSDI-approach is helpful in the RM setting, where consistency across features and time has to be reached, whereas SSSD^{S4}and its variants show clear advantages for MNR and BM (and TF as discussed below) where modeling the time dependence is of primary importance.

Existing approaches fail to produce meaningful BM imputations Figure 3 shows imputations on a BM scenario for a subset of models from the PTB-XL imputation task. The main purpose is to demonstrate that the achieved improvements through the proposed approaches lead to superior samples in a way that is even apparent to the naked eye. The top figure demonstrates that the state-of-the-art imputer is unable to produce any meaningful imputations (as visible both from the shaded quantiles as well as from the exemplary imputation). As an example, the identification of a QRS complex with a duration inside the range of 0.08 sec to 0.12 sec is considered as normal signals. However, the model fails to detect the complex and rather

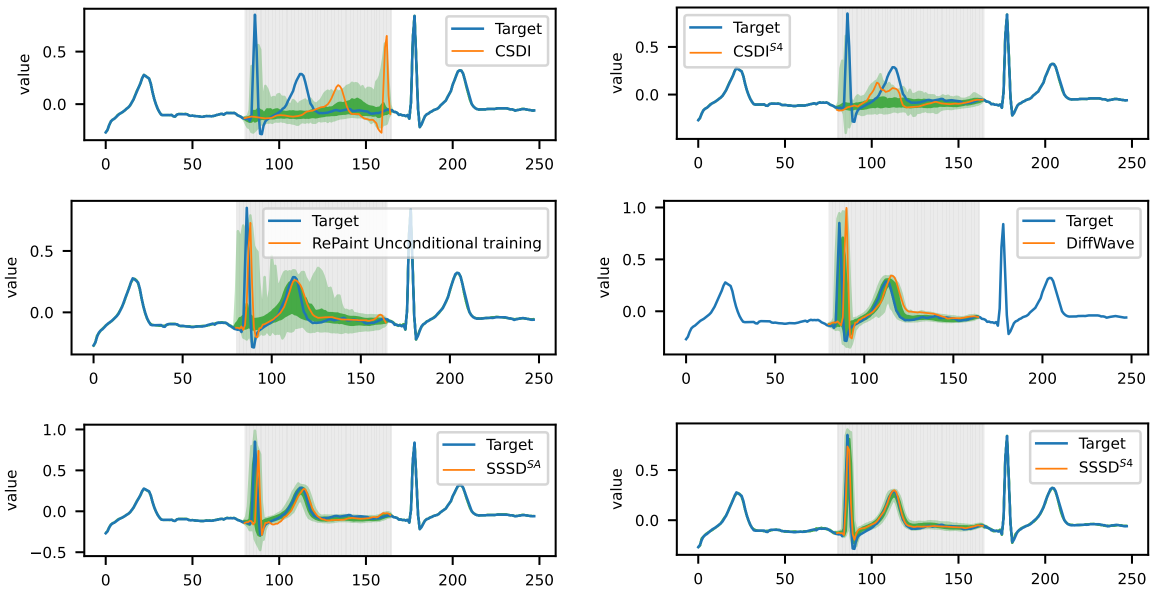

Figure 3: PTB-XL BM imputations for the V5 lead of an ECG from a healthy patient.

shows a misplaced R peak. The imputation quality improves qualitatively with the proposed CSDI$^{\text{S4}}$ variant but still misses essential signal features. Only the proposed DiffWave imputer, SSSD$^{\text{SA}}$ and SSSD$^{\text{S4}}$ models capture all essential signal features. The qualitatively and quantitatively best result is achieved by SSSD$^{\text{S4}}$, which excels at the task and shows well-controlled quantile bands as expected for a normal ECG sample.

SSSD$^{\text{S4}}$ shows competitive imputation performance compared to state-of-the-art approaches on other data sets and high missing ratios To demonstrate that the excellent performance of SSSD$^{\text{S4}}$ extends to further data sets, we collected the MuJoCo data set Rubanova et al. (2019) from Shan et al. (2021) to test SSSD$^{\text{S4}}$ on highly sparse RM scenarios such as 70%, 80%, and 90%, for which we compare performance against the baselines RNN GRU-D Che et al. (2016), ODE-RNN Rubanova et al. (2019), NeuralCDE Morrill et al. (2021), Latent-ODE Rubanova et al. (2019), NAOMI Liu et al. (2019), and NRTSI Shan et al. (2021). We report an averaged MSE for a single imputation per sample on the test set over 3 trials. All baselines results were collected from Shan et al. (2021).

Table 2: Imputation MSE results for the MuJoCo data set. Here, we use a concise error notation where the values in brackets affect the least significant digits e.g. 0.572(12) signifies 0.572 ± 0.012.

Model	**70% RM**	**80% RM**	**90% RM**
RNN GRU-D	11.34e-3	14.21e-3	19.68e-3
ODE-RNN	9.86e-3	12.09e-3	16.47e-3
ODE-RNN	9.86e-3	12.09e-3	16.47e-3
NeuralCDE	8.35e-3	10.71e-3	13.52e-3
Latent-ODE	3.00e-3	2.95e-3	3.60e-3
NAOMI	1.46e-3	2.32e-3	4.42e-3
NRTSI	0.63e-3	1.22e-3	4.06e-3
SSSD$^{\text{S4}}$	**0.59(8)e-3**	**1.00(5)e-3**	**1.90(3)e-3**

Table 2 shows the empirical RM results on the MuJoCo data set, where SSSD$^{\text{S4}}$ outperformed all the baselines for all the missingness scenarios, in particular, on the highest RM ratio 90%, where SSSD$^{\text{S4}}$ achieved a error reduction of more than 50%.

SSSD^{S4} performance on high-dimensional data sets We also explore the potential of SSSD^{S4} on data sets with more than 100 channels following the simple but potentially sub-optimal channel splitting strategy described above. We implemented the RM imputation task on the Electricity data set Dua & Graff (2017) from Du et al. (2022) which contains 370 features at different missingness ratios such as 10%, 30% and 50%. As baselines, we consider M-RNN Yoon et al. (2019), GP-VAE Fortuin et al. (2020), BRITS Cao et al. (2018), SAITS Du et al. (2022) and a transformer variant from Du et al. (2022). We report an averaged MAE, RMSE, and MRE from one sample generated per test sample over a 3 trial period.

Table 3: 50% RM imputation results for the Electricity data set (extracts, see Table 21 for further settings comparison).

Model	MAE	RMSE	MRE
M-RNN	1.283	1.902	68.7%
GP-VAE	1.097	1.572	58.8%
BRITS	1.037	1.538	55.5%
Transformer	0.895	1.410	47.9%
SAITS	0.876	1.377	46.9%
SSSD^{S4}	**0.532(1)**	**0.821(1)**	**28.5(1)%**

Overall, SSSD^{S4} excelled at the imputation task, demonstrating significant error reductions against the strongest baseline SAITS, as example Table 3 contains the results for the 50% RM imputation task, where SAITS presented 0.876, 1.377, and 49.9% of MAE, RMSE, and MRE respectively, while we achieved with SSSD^{S4} 0.532, 0.821, and 28.5%, representing outstanding error reductions of 39.3%, 40.4%, and 39.2% respectively. For the rest of the RM settings, SSSD^{S4} still shows significant error reductions, see Table 21 in the technical appendix for details. Similarly, we tested on a 25% RM task from Cini et al. (2022) on the PEMS-Bay and METR-LA data sets Li et al. (2018) which contains 325 and 207 features respectively. On the PEMS-Bay data set, SSSD^{S4} outperformed well-established baselines such as MICE White et al. (2011), rGAIN Miao et al. (2021), BRITS Cao et al. (2018), and MPGRU Huang et al. (2019) in terms of all three metrics MAE, MSE, and MRE, while it was only superseded by the recently proposed GRIN Cini et al. (2022). On the METR-LA data set, SSSD^{S4}is again outperformed by GRIN but on par with the remaining models. We refer to Table 22 in the technical appendix for a more in-depth results discussion.

4.3 Time series forecasting

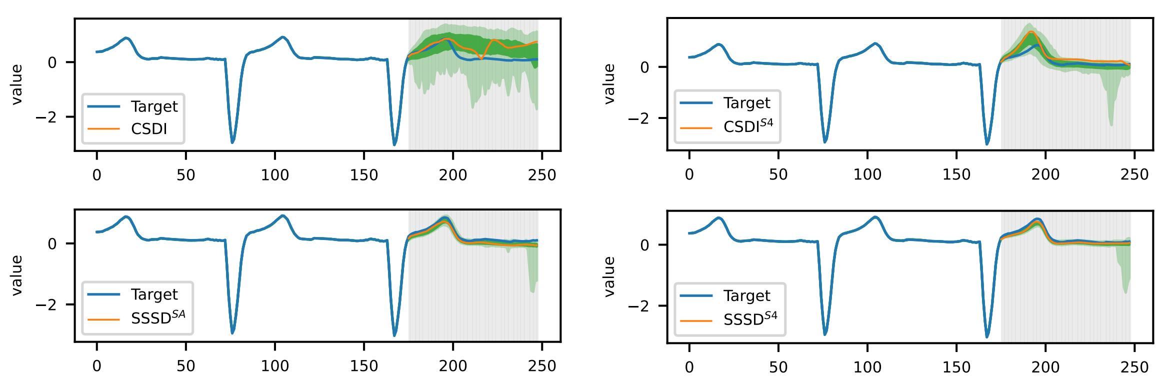

Figure 4: PTB-XL TF for the V1 lead of an ECG from a patient with a complete left bundle branch block (CLBBB).

SSSD^{S4} on proposed data set We implemented the CSDI, CSDI^{S4}, SSSD^{SA}, and SSSD^{S4} models on two data sets. For both, we report MAE, and RMSE as metrics for hundred samples generated per test sample in three trials. As first application, we reconsider the case of ECG data from the *PTB-XL data set*. As before, we work at a sampling frequency of 100 Hz, however, for this task at $L = 1000$ time steps per sample, for which we condition on 800 time steps and forecast on 200. SSSD$^{\mathrm{SA}}$ outperformed on MAE with 0.087, while SSSD$^{\mathrm{S4}}$ achieved a slightly larger error of 0.090. SSSD$^{\mathrm{S4}}$ outperformed on RMSE with 0.219, CSDI$^{\mathrm{S4}}$ achieved smaller errors than CSDI on the three metrics. Again, the samples presented in table 4 demonstrate a clear improvement that is again even visible to the naked eye.

SSSD$^{\mathrm{S4}}$ shows competitive forecasting performance compared to state-of-the-art approaches on various data sets We test on the *Solar* data set collected from GluonTS Alexandrov et al. (2020), a forecasting task where the conditional values and forecast horizon are 168 and 24 time steps respectively. As baselines, we consider CSDI Tashiro et al. (2021), GP-copula Salinas et al. (2019), Transformer MAF (TransMAF) Rasul et al. (2021b), and TLAE Nguyen & Quanz (2021). All baseline results were collected from its respective original publications. We report an averaged MSE of 100 samples generated per test sample over three trials.

Table 4: Time series forecasting results for the solar data set.

Model	**MSE**
GP-copula	9.8e2 ± 5.2e1
TransMAF	9.3e2
TLAE	6.8e2 ± 7.5e1
CSDI	9.0e2 ± 6.1e1
SSSD$^{\mathrm{S4}}$	**2.7e2 ± 4.32**

The empirical results for the solar data set demonstrate the excellent forecasting capabilities of SSSD$^{\mathrm{S4}}$. It achieved a MSE of 2.7e-2 corresponding to a 60% error reduction compared to TLAE as strongest baseline.

Finally, we demonstrate SSSD$^{\mathrm{S4}}$'s forecasting capabilities on conventional benchmark data sets for long-horizon forecasting. To this end, we collected the preprocessed ETTm1 data set from Zhou et al. (2021) and used it for forecasting at five different forecasting settings, where the forecasting length is of 24, 48, 96, 288 and 672 time steps, and the conditional values are 96, 48, 284, 288, and 384 time steps, respectively. We compare to LSTnet Lai et al. (2018), LSTMa Bahdanau et al. (2015), Reformer Kitaev et al. (2020), LogTrans Li et al. (2019), Informer Zhou et al. (2021), one Informer baseline called Informer(†), and Autoformer Wu et al. (2021a). We report an averaged MAE and MSE for a single sample generated for each test sample over 2 trials. All baseline results were collected from Zhou et al. (2021); Wu et al. (2021a).

The experimental results confirm again SSSD$^{\mathrm{S4}}$'s robust forecasting capabilities, specifically for long-horizon forecasting, where also the conditional time steps increases. For the first setting, SSSD$^{\mathrm{S4}}$ outperformed the rest of the baselines on MAE. For the second setting, on shorter forecast conditional and target lengths SSSD$^{\mathrm{S4}}$ scores are comparable with Autoformer and Informer, while on the remaining three settings, SSSD$^{\mathrm{S4}}$ outperformed the all baselines with in parts significant error reductions but does not reach the performance of Autoformer. It is worth stressing that Autoformer represents a very strong baseline, which was tailored to time series forecasting including a decomposition into seasonal and long-term trends, which is perfectly adapted to the forecasting task on the ETTm1. It is an interesting question for future research if SSSD$^{\mathrm{S4}}$ can also profit from a stronger inductive bias and/or if there are forecasting scenarios, which do not follow the decomposition into seasonal and long-term trends so perfectly, where such a kinds of inductive bias can even be harmful. To summarize, we see it as a very encouraging sign that a versatile method such as SSSD$^{\mathrm{S4}}$, which is capable of performing imputation as well as forecasting, is clearly competitive with most of the forecasting baselines.

Table 5: Time series forecasting results on the ETTm1 data set.

Model	**24**	**48**	**96**	**288**	**672**
			TF on ETTm1 (MAE)		
LSTNet	1.170	1.215	1.542	2.076	2.941
LSTMa	0.629	0.939	0.913	1.124	1.555
Reformer	0.607	0.777	0.945	1.094	1.232
LogTrans	0.412	0.583	0.792	1.320	1.461
Informer†	0.371	0.470	0.612	0.879	1.103
Informer	0.369	0.503	0.614	0.786	0.926
Autoformer	0.403	**0.453**	**0.463**	**0.528**	**0.542**
SSSD$^{\mathbf{S4}}$	**0.361(6)**	0.479(8)	0.547(12)	0.648(10)	0.783(66)
			TF on ETTm1 (MSE)		
LSTNet	1.968	1.999	2.762	1.257	1.917
LSTMa	0.621	1.392	1.339	1.740	2.736
Reformer	0.724	1.098	1.433	1.820	2.187
LogTrans	0.419	0.507	0.768	1.462	1.669
Informer†	**0.306**	0.465	0.681	1.162	1.231
Informer	0.323	0.494	0.678	1.056	1.192
Autoformer	0.383	**0.454**	**0.481**	**0.634**	**0.606**
SSSD^{S4}	0.351(9)	0.612(2)	0.538(13)	0.797(5)	0.804(45)

5 Conclusion

In this work, we proposed the combination of structured state space models as emerging model paradigm for sequential data with long-term dependencies and diffusion models as the current state-of-the-art approach for generative modeling. The proposed SSSD^{S4}outperforms existing state-of-the-art imputers on various data sets under different missingness scenarios, with a particularly strong performance in blackout missing and forecasting scenarios, provided the number of input channels does not grow too large. In particular, we present examples where the qualitative improvement in imputation quality is even apparent to the naked eye. We see the proposed technology as a very promising technology for generative models in the time series domain, which opens the possibility to build generative models conditioned on various kinds of information from global labels to local information such as semantic segmentation masks, which in turn enables a broad range of further downstream applications. The source code underlying our experiments is available under `https://anonymous.4open.science/r/SSSD`.

References

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. URL `http://jmlr.org/papers/v21/19-820.html`.

Adham Atyabi, Frédérick Shic, and Adam J. Naples. Mixture of autoregressive modeling orders and its implication on single trial eeg classification. *Expert systems with applications*, 65:164–180, 2016. doi: 10.1016/j.eswa.2016.08.044.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Faraj Bashir and Hua-Liang Wei. Handling missing data in multivariate time series using a vector autoregressive model-imputation (var-im) algorithm. *Neurocomput.*, 276(C):23–30, feb 2018. ISSN 0925-2312. doi: 10.1016/j.neucom.2017.03.097. URL `https://doi.org/10.1016/j.neucom.2017.03.097`.

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*, volume 31, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/734e6bfcd358e25ac1db0a4241b95651-Paper.pdf`.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values, 2016. URL `https://arxiv.org/abs/1606.01865`.

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*, 2020.

Xinyu Chen, Mengying Lei, Nicolas Saunier, and Lijun Sun. Low-rank autoregressive tensor completion for spatiotemporal traffic data imputation. *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2021. doi: 10.1109/TITS.2021.3113608.

Andrea Cini, Ivan Marisca, and Cesare Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=kOu3-S3wJ7`.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021. URL `https://proceedings.neurips.cc/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf`.

Wenjie Du, David Cote, and Yan Liu. SAITS: Self-Attention-based Imputation for Time Series. *arXiv preprint 2202.08516*, 2022.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Chenguang Fang and Chen Wang. Time series data imputation: A survey on deep learning approaches. *arXiv preprint arXiv:2011.11347*, 2020.

Vincent Fortuin, Dmitry Baranchuk, Gunnar Raetsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 1651–1661, 26–28 Aug 2020. URL `https://proceedings.mlr.press/v108/fortuin20a.html`.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Re. It's raw! Audio generation with state-space models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 7616–7633, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/goel22a.html`.

Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. PhysioBank, PhysioToolkit, and PhysioNet. *Circulation*, 101(23):e215–e220, 2000. doi: 10.1161/01.CIR.101.23.e215.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1474–1487, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/102f0bb6efb3a6128a3c750dd16729be-Paper.pdf`.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022a.

Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. *arXiv preprint 2206.12037*, 2022b. doi: 10.48550/ARXIV.2206.12037. URL `https://arxiv.org/abs/2206.12037`.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`.

Jonathan Ho, Chitwan Saharia, William Chan, David Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23:47:1–47:33, 2022a.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint 2204.03458*, 2022b.

Yujun Huang, Yunpeng Weng, Shuai Yu, and Xu Chen. Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting. In *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications*, pp. 678–685, 2019. doi: 10.1109/TrustCom/BigDataSE.2019.00096. URL `https://doi.org/10.1109/TrustCom/BigDataSE.2019.00096`.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *9th International Conference on Learning Representations, ICLR 2021*, 2021. URL `https://openreview.net/forum?id=a-xFK8Ymz5J`.

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*, pp. 95–104, 2018. doi: 10.1145/3209978.3210006. URL `https://doi.org/10.1145/3209978.3210006`.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL `https://proceedings.neurips.cc/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf`.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations (ICLR '18)*, 2018.

WC. Lin and CF. Tsai. Missing value imputation: a review and analysis of the literature. *Artificial Intelligence Review*, 53(2):1487–1509, 2020. doi: 10.1007/s10462-019-09709-4. URL `https://link.springer.com/article/10.1007/s10462-019-09709-4`.

Yukai Liu, Rose Yu, Stephan Zheng, Eric Zhan, and Yisong Yue. Naomi: Non-autoregressive multiresolution sequence imputation. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL `https://proceedings.neurips.cc/paper/2019/file/50c1f44e426560f3f2cdcb3e19e39903-Paper.pdf`.

Andreas Lugmayr, Martin Danelljan, Andreas Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. RePaint: Inpainting using denoising diffusion probabilistic models. In *Computer Vision and Pattern Recognition*, 2022a.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models, 2022b. URL `https://arxiv.org/abs/2201.09865`.

Yonghong Luo, Xiangrui Cai, Ying ZHANG, Jun Xu, and Yuan xiaojie. Multivariate time series imputation with generative adversarial networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL `https://proceedings.neurips.cc/paper/2018/file/96b9bff013acedfb1d140579e2fbeb63-Paper.pdf`.

Yonghong Luo, Ying Zhang, Xiangrui Cai, and Xiaojie Yuan. E^{2}gan: End-to-end generative adversarial network for multivariate time series imputation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3094–3100, 7 2019. doi: 10.24963/ijcai.2019/429. URL `https://doi.org/10.24963/ijcai.2019/429`.

Xiaoye Miao, Yangyang Wu, Jun Wang, Yunjun Gao, Xudong Mao, and Jianwei Yin. Generative semi-supervised learning for multivariate time series imputation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):8983–8991, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17086.

James Morrill, Cristopher Salvi, Patrick Kidger, and James Foster. Neural rough differential equations for long time series. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 7829–7838, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/morrill21b.html.

Nam Nguyen and Brian Quanz. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9117–9125, May 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17101.

Muhammad S. Osman, Adnan M. Abu-Mahfouz, and Philip R. Page. A survey on data imputation techniques: Water distribution system as a use case. *IEEE Access*, 6:63279–63291, 2018. doi: 10.1109/ACCESS.2018.2877269.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pp. 8857–8868. PMLR, 2021a.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M. Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *9th International Conference on Learning Representations, ICLR 2021*, 2021b.

Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf.

Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=c7NBMfDXbW.

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*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/0b105cf1504c4e241fcc6d519ea962fb-Paper.pdf.

Tolou Shadbahr, Michael Roberts, Jan Stanczuk, Julian D. Gilbey, Philip Teare, Sören Dittmer, Matthew Thorpe, Ramón Viñas Torné, Evis Sala, Pietro Lio', Mishal N. Patel, AIX-COVNET Collaboration, James H. F. Rudd, Tuomas Mirtti, Antti S. Rannikko, John A. D. Aston, Jing Tang, and Carola-Bibiane Schonlieb. Classification of datasets with imputed missing values: does imputation quality matter? *arXiv preprint 2206.08478*, 2022.

Siyuan Shan, Yang Li, and Junier B. Oliva. Nrtsi: Non-recurrent time series imputation, 2021. URL https://arxiv.org/abs/2102.03340.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265, 07–09 Jul 2015. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34: 24804–24816, 2021.

Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7 (1):154, 2020a. doi: 10.1038/s41597-020-0495-6. URL `https://doi.org/10.1038/s41597-020-0495-6`.

Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset, 2020b.

Ian R White, Patrick Royston, and Angela M Wood. Multiple imputation using chained equations: Issues and guidance for practice. *Stat. Med.*, 30(4):377–399, February 2011.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430, 2021a.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2021b. URL `https://arxiv.org/abs/2106.13008`.

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, 2019. doi: 10.1109/TBME.2018.2874712.

Zhihui Zhang, Xiangjun Xiao, Wen Zhou, Dakai Zhu, and Christopher I Amos. False positive findings during genome-wide association studies with imputation: influence of allele frequency and imputation accuracy. *Human Molecular Genetics*, 31(1):146–155, 2022. doi: 10.1093/hmg/ddab203. URL `https://doi.org/10.1093/hmg/ddab203`.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, volume 35, pp. 11106–11115, 2021.

A Technical Appendix

A.1 Training hyperparameters and architectures

Table 6: LAMC hyperparameters.

Hyperparameter	Value
Rhos	0.001, 0.01, 0.1, 1, 2, 3, 5, 10, 20.
Lambdas	0.001, 0.01, 0.1, 1, 2, 3, 5, 6, 8, 10.
Epsilons	1e-4, 1e-3, 1e-2, 1, 2, 5, 8, 10.
Ranks	2, 5, 10, 20, 50, 80, 100.
Iterations	100

LAMC For the LAMC algorithm Chen et al. (2021), we defined a range of hyperparameters to be chosen by an exhaustive grid search given the best test metrics. Table 6 contains the LAMC hyperparameter implemented through all the experiments. We decided to implement this algorithm as a baseline as it represents a qualitatively different methodology for time series imputation compared to our probabilistic diffusion models, which relies on matrix completion approach.

Table 7: CSDI hyperparameters.

Hyperparameter	Value
Residual layers	4
Residual channels	64
Diffusion embedding dim.	128
Schedule	Quadratic
Diffusion steps T	50
B_0	0.0001
B_1	0.5
Feature embedding dim.	128
Time embedding dim.	16
Self-attention layers time dim.	1
Self-attention heads time dim.	8
Self-attention layers feature dim.	1
Self-attention heads feature dim.	8
Optimizer	Adam
Learning rate	1×10^{-3}
Weight decay	1×10^{-6}

CSDI and CSDI$^{\mathbf{S4}}$ We use the official implementation of CSDI Tashiro et al. (2021) as a baseline with the authors' default hyperparameters settings. The only modification for CSDI$^{\text{S4}}$ was to replace the temporal transformer layer by a S4 layer. Table 7 contains all the hyperparameters that for CSDI and CSDI$^{\text{S4}}$ training and architecture. For the CSDI$^{\text{S4}}$ setting, the S4 model implemented in all experiments is a bidirectional layer with layer normalization, no dropout, and $N = 64$ as internal state dimensionality.

DiffWave Table 8 contains all the hyperparameters for training and architecture of the DiffWave imputer. As previously discussed, DiffWave was released by its authors Kong et al. (2021) as diffusion model for speech synthesis. Here, we are referring to DiffWave model as a custom implementation in the context of time series imputation/forecasting. Some of the architectural differences are a large number of residual layers and

Table 8: DiffWave hyperparameters.

Hyperparameter	Value
Residual layers	36
Residual channels	256
Skip channels	256
Diffusion embedding dim. 1	128
Diffusion embedding dim. 2	512
Diffusion embedding dim. 3	512
Schedule	Linear
Diffusion steps T	200
B_0	0.0001
B_1	0.02
Optimizer	Adam
Learning rate	2×10^{-4}

channels, but most importantly our proposed training procedure which involves a different use of conditional information, such as the concatenation conditioning information and binary mask.

Table 9: SSSD$^{\text{SA}}$ hyperparameters.

Hyperparameter	Value
Residual layers	6
Pooling factor	[2,2]
Feature expansion	2
Diffusion embedding dim. 1	128
Diffusion embedding dim. 2	512
Diffusion embedding dim. 3	512
Schedule	Linear
Diffusion steps T	200
B_0	0.0001
B_1	0.02
Optimizer	Adam
Learning rate	2×10^{-4}

SSSD$^{\text{SA}}$ Table 9 contains the SSSD$^{\text{SA}}$ hyperparameters for all our experiments. SSSD$^{\text{SA}}$ is a variant of the SaShiMi Goel et al. (2022) model, a 128-dimensional U-Net model with six residual layers, consisting of an S4 and feed-forward layers in a block manner at each pooling level, where the pooling factor in decreasing the sequence length is 2 and 2 and its respective feature expansion of 2. There is a single three-layer diffusion embedding with dimensions 128, 512, and 512, respectively for all the residual layers. As for S4, we use a bidirectional layer with layer normalization, no dropout, and an internal state dimension of $N = 64$, but with a gated linear unit in each layer.

SSSD$^{\text{S4}}$ We present in Table 10 the proposed SSSD$^{\text{S4}}$ hyperparameters and training settings used throughout all our experiments. The SSSD$^{\text{S4}}$ a model builds on DiffWave Kong et al. (2021) and consists of 36 stacked residual layers with 256 residual and skip channels. As for SSSD$^{\text{SA}}$, SSSD$^{\text{S4}}$ uses a three-layer diffusion embedding of 128, 256, and 256 hidden units dimensions with a swish activation function after the second and third layer. After the addition of diffusion embedding, we implemented a convolutional layer to double the channel dimension of the input before computing the first S4 diffusion. Similarly, after a similar expansion of the conditional information and its addition to the input, there is the application of a second S4 layer.

Table 10: SSSD^{S4} hyperparameters.

Hyperparameter	**Value**
Residual layers	36
Residual channels	256
Skip channels	256
Diffusion embedding dim. 1	128
Diffusion embedding dim. 2	512
Diffusion embedding dim. 3	512
Schedule	Linear
Diffusion steps T	200
B_0	0.0001
B_1	0.02
Optimizer	Adam
Learning rate	2×10^{-4}

Then the output is passed through a gated tanh activation function as non-linearity, from which we then project back from residual channels to the channel dimensionality with a convolutional layer. We used 200 time steps on a linear schedule for diffusion configuration from a beta of 0.0001 to 0.02. We utilized Adam as an optimizer with a learning rate of 2×10^{-4}. For the S4 model, similar to the other approaches, we used a bidirectional layer with layer normalization, no dropout, and an internal state dimension of $N = 64$.

Table 11: S4 hyperparameters.

Hyperparameter	**Value**
Layers	1
State N dimensions	64
Bidirectional	Yes
Layer normalization	Yes
Drop-out	0.0
Maximum length	as required

S4 In Table 11 we present the hyperparameter settings for the state space S4 model Gu et al. (2022a) used in this work. Overall, we utilize a single S4 layer with a bidirectional setting for time dependencies learning of series in both directions. Similarly, we applied a layer normalization and utilize a internal state of dimensionality $N = 64$ as used in prior work Gu et al. (2022a).

Hyperparameter tuning In this paragraph, we briefly describe the strategy that led to the hyperparameter selections discussed in the previous paragraphs: As a first remark, we focus on SSSD^{S4} in this section. These settings were applied to SSSD^{SA} without further experimentation. We only adapted the number of residual layers to obtain a model with comparable computational complexity as SSSD^{S4}. For CSDI^{S4}, we left the hyperparameters as close to the original default hyperparameters as possible. As a general remark, we point out that the hyperparameter optimization was carried out based on validation set scores on the PTB-XL data set at 248 time steps. We did not adjust the hyperparameters for other data sets, which can be seen as a hint for the robustness of the proposed method. Below, we briefly comment on specific aspects of the hyperparameter selection.

Diffusion hyperparameters: We implemented a linear schedule with a minimum noise level B_0 of 0.001 and a maximum noise level of B_1 as 0.5, based on trial and error. We found that the fewer diffusion steps, the faster the network converges during training, however, at the cost of less accurate results. In this respect,

using 200 diffusion steps represented a reasonable compromise.
Three-layer diffusion embedding: From the beginning of our experiments, we implemented a three-layer embedding with 128, 512, and 512 hidden units, respectively. While reducing the residual layers in some experiments, we also experimented with reducing those dimensionalities to 16, 32, 32, or 64, 64, 256. However, we found that the former choice led to better results.
Residual channels and skip channels: we introduced a large number of channels (256) to avoid the degradation problem.
Residual layers: We set the number of residual layers to 36 after the experimental phase of our ablation study. In our previous experiments with a single S4 layer, we worked with 48 layers but reduced it to 36 to increase the speed of convergence during training and to reduce the computational complexity of the overall model, which in variant C requires two S4 layers.

A.2 Data sets description

Table 12: PTB-XL data set details.

Description / Setting	PTB-XL 248	PTB-XL 1000
Train size	69,764	17,441
Validation size	8,772	2,193
Test size	8,812	2,203
Training batch	32	4
Sample length	248	1000
Sample features	12	12
Conditional values	198	800
Target values	52	200

PTB-XL data set Table 12 contains the PTB-XL data set details Wagner et al. (2020a;b); Goldberger et al. (2000). The PTB-XL ECG data set consists of 21837 clinical 12-lead ECGs, each lasting 10 seconds, from 18885 patients. The data set was collected and preprocessed as in the Physionet repository. We collected for all experiments the ECG signals at a sampling rate of 100 Hz. For the three imputation and forecasting scenarios, we utilize 20% as target values. In all of the settings, the number of input channels is 12 as it is a 12-lead electrocardiogram. For the 248 time steps setting, the data set was preprocessed on crops, which corresponds to 69,764 training and 8,812 test samples.

Table 13: Electricity data set details.

Description	Value
Train size	817
Test size	921
Training batch	43
Sample length	100
Data set features	370
Sample features	37
Conditional values	90, 70, 50
Target values	10, 30, 50

Electricity data set Table 13 contains details on the electricity data set, which was used for RM imputation. The electricity data set from the UCI repository Dua & Graff (2017) contains electricity usage data (in kWh) gathered from 370 clients which represent 370 features every 15 minutes. The data set was collected and preprocessed as in Du et al. (2022). As the data set not contain missing values, we collected the complete data set and in our experiments and randomly dropped the values for the computation of targets according to the RM scenario. The data is already normalized and we present results in this setting. The first 10 months of data (2011/01 - 2011/10) are the test set, the following 10 months of data (2011/11 - 2012/08) the validation set and the left (2012/09 - 2014/12) the training set. We directly utilize the training and test set leaving the validation set out. We consider this a challenging task due to the fact that the test set contains many clients that were not present in the training set. The data set contains 817 samples of a length of 100 time steps with the 370 mentioned features. However, we observed faster convergence when applying feature sampling, specifically, splitting the 370 channels into 10 batches of 37 features each, then, passing to the network mini-batches of 43 samples each with 37 features and its respective length of 100 to ensure that we do not drop any data during training.

Table 14: MuJoCo data set details.

Description	**Value**
Train size	8000
Test size	2000
Training batch	50
Sample length	100
Data set features	14
Conditional values	50, 30, 20, 10
Target values	50, 70, 80, 90

MuJoCo data set Table 14 contains details of the MuJoCo data set, which is a data set for physical simulation, created by the authors at Rubanova et al. (2019) using the "Hopper" model from the Deepmind Control Suite. The hopper's initial placements and speeds are randomly sampled in such a way that the hopper rotates in the air before crashing to the earth. There are 10,000 sequences of 100 regularly sampled time points for each trajectory in the 14-dimensional data set. By convention, there is a 80/20 random split for training and testing. Both data sets were already preprocessed by the NRTSI authors in Shan et al. (2021) to ensure a fair comparison.

Table 15: PEMS-Bay data set details.

Description	**Value**
Train size	1200
Test size	50
Training batch	40
Sample length	200
Data set features	325
Sample features	65
Conditional values	150
Target values	50

PEMS-BAY data set Table 15 contains details on the PEMS-Bay data set Huang et al. (2019), which was compiled for The Performance Measurement System (PeMS) of the California Transportation Agencies (CalTrans). It represents a network of 325 traffic sensors in the California Bay Area. It contains traffic readings every five minutes for six months, from January 1st 2017 to May 31st 2017. The data set was collected from Cini et al. (2022). The methodology of Cini et al. (2022) required the creation of an adjacency matrix from the original data set due to the use of a graph neural network. We work directly with the data set, which has 52,116 time steps and 325 features. However, in order to obtain samples for training, we considered 200 time steps over the first 50,000 to get 250 samples, we set the first 240 for training and the 10 rest for testing. Then, we fit a standard scaler on the training set and transform both the training and the test sets. Finally, we feature sampled to obtain batches by a factor of 5, where for training set we obtained 5 batches of 240 samples with 200 time steps and 65 channels each. Finally, we iterate over the second dimension to pass batches of 40 for training.

Table 16: METR-LA data set details.

Description	Value
Train size	750
Test size	100
Training batch	50
Sample length	200
Data set features	207
Sample features	40
Conditional values	150
Target values	50

METR-LA data set Table 16 contains details on the METR-LA data set, which represents the traffic data from a road network consisting of 207 loop detectors in a period between the 1st of March 2012 to the 30th of June 2012. Similar to the preprocessing of the PEMS-Bay data set, we selected from the original set the first 34,000 time steps and the first 200 features, we obtained 170 samples of 200 time steps each. We used the first 150 as training set and the remaining 20 for testing. Finally, we sampled features by a factor of 5, obtaining for training 5 batches of 150 samples with 200 time steps and 40 features each. For training we iterate over the first dimension to obtain batches of 50 samples.

Table 17: Solar data set details.

Description	Value
Train size	130
Test size	16
Training batch	65
Sample length	192
Data set features	128
Sample features	64
Conditional values	168
Target values	24

Solar data set Table 17 presents details on the solar data set. The original data set contains the solar power production records in the year 2006, sampled every 10 minutes from 137 photovoltaic power plants in Alabama State Lai et al. (2018). However, as a conventional benchmark, we collected the data set from GluonTS Alexandrov et al. (2020) which represents hourly sampled data. The task is to condition on 168 time steps to forecast the following 24. The whole data set contains 73 samples of 192 time steps and 128 features each in a chronological manner, we use the first 65 as training set and the remaining 8 as the set. Similar to the preprocessing applied to other data sets described above, we feature sample this data set by a factor of 2, where for the training set, for example, we obtained 2 batches of 65 samples with 192 time steps and 64 features each. We standard scale the train and test sets for training using training set statistics.

ETTm1 data set Table 18 contains the ETTm1 data set details. This data set was created to investigate the amount of detail required for long-time series forecasting based on the Electricity Transformer Temperature (ETT), which is an important measure for the long-term deployment of electric power. The data set contains

Table 18: ETTm1 data set details.

Description	Value
Train size	33,865, 34,417, 34,000, 33,600, 33,200
Test size	11,490, 10,000, 11,420, 10,000, 10,000
Training batch	65, 127, 17, 14, 4
Sample length	120, 96, 480, 576, 1,052
Data set features	7
Conditional values	96, 48, 384, 288, 384
Target values	24, 48, 96, 288, 672

information from a compilation of 2-year data from two distinct Chinese counties. Here, we work with ETTm1 which covers data at a 15-minute level. The data is composed of the target value oil temperature and six power load features. We collected and preprocessed the data directly from Zhou et al. (2021), we forecast in each of the benchmarking horizons, utilizing train and test sets, coming from the original split of train/val/test set, which was for 12/4/4 months, respectively. As seen in Table 18, there are five different preprocessing settings implemented with regard to the forecasting horizon, the first table contains information on three and the second for the remaining two, where the main difference is the number of values used to condition on, and the targets to forecast. Similarly, there are some differences with respect to the batch size used during training. As the samples get longer, we utilize smaller batch sizes. Finally, for samples generation, as the test sets are very large with more than 10,000 samples each, we subset the test set in order to obtain batches.

A.3 Additional content

A.3.1 S4 Model

$$A_{nk} = -\begin{cases} (2n+1)^1/2(2k+1)1/2 & \text{if}\, n > k \\ n+1 & \text{if}\, n = k \\ 0 & \text{if}\, n < k \end{cases} \tag{5}$$

S4 is a deep structured state space model, which is built with four ideal components for time series analysis, specifically, for the handling of long sequences. First of all, it contains the continuous representation of SSMs previously introduced in Eq. 4, for which with the help of HiPPO matrices Gu et al. (2020), it provides an importance score for each past time step through an online compression of signals and a novel and robust updating system (HiPPO-LegS) that scales for long sequences. In a nutshell, HiPPO aims to overcome the issue of time series of different lengths and the vanishing gradient problem. HiPPO is referenced in 5, which compresses the scaled Legendre measure (LegS) operator for a uniform system update on the SMM vector A.

$$\begin{aligned} \overline{A} &= (I - \Delta/2 \cdot A)^{-1}(I + \Delta/2 \cdot A) \\ \overline{B} &= (I - \Delta/2 \cdot A)^{-1}\Delta B \\ \overline{C} &= C \end{aligned} \tag{6}$$

Additionally, the HiPPO architecture provides the advantage of being able to handle irregularly sampled data with the help of a recurrent discretization procedure by a step size Δ using the bilinear method. The discretization procedure in Eq. 6 creates a sequence-to-sequence mapping that yields a recurrent state space; here, the SSMs can be computed like an RNN. Concretely, it can be viewed as a hidden state with a transition matrix $\overline{A}$.

$$y_k = \overline{CAB}^k u_0 + \overline{CAB}^{k-1} u_1 + \ldots + \overline{CAB} u_{k-1} + \overline{CB} u_k \quad y = \overline{K} * u \tag{7}$$

$$\overline{k} \in \mathbb{R}^L := K_L(\overline{A}, \overline{B}, \overline{C}) := (\overline{CAB}^i)_{i\in[L]} = (\overline{CB}, \overline{CAB}, ..., \overline{CAB}^{L-1}) \tag{8}$$

The RNN described previously can be transformed into a CNN by unrolling in the discrete convolution step, which is shown in 7, where a single convolution can be computed efficiently with fast Fourier transform given the SSM convolution kernel $\overline{K}$ represented in Eq. 8. We strongly encourage the reader to refer to Gu et al. (2022a) and Gu et al. (2020) for further investigation of the S4 model and its components. From an external perspective, however, the S4 layer that is composed of multiple S4 blocks can be considered as a drop-in replacement for a one-dimensional (potentially dilated) convolutional layer, a RNN, or a transformer layer, which is particularly adapted to the requirements of time series analysis.

A.3.2 Performance Metrics

This section describes the performance metrics used throughout this work. We use a wide range of performance metrics for the computation of errors between targets and imputed values ($e_t = y - \hat{y}$), where for y and $\hat{y}$ we apply the respective conditional masking for metrics computation ($m_{\text{eval}} = m_{\text{mvi}} \odot (1 - m_{\text{imp}})$).

$$MAE = \frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{K} |(y - \hat{y}) \odot m_{\text{eval}}|_{t,k} \tag{9}$$

$$MSE = \frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{K} ((y - \hat{y}) \odot m_{\text{eval}})^2_{t,k} \tag{10}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{K} ((y - \hat{y}) \odot m_{\text{eval}})^2_{t,k}} \tag{11}$$

$$MRE = \frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{K} m_{\text{eval}\,t,k} \frac{|(y - \hat{y})|_{t,k}}{y_{t,k}} \tag{12}$$

Firstly, from a deterministic perspective, specifically for the computation of absolute errors, we implemented mean absolute error (MAE) equation 9 which is determined by dividing the total absolute errors by the total sample size of observations n. For the account of larger error sensitivity we implemented mean squared error (MSE) equation 10 which is the sum $\sum_{t=1}^{n}$ of squared errors e_t^2 divided by the total sample size of observations n, similarly, we implemented root mean squared error (RMSE) equation 11 to account for larger errors while preserving the error ranges in proportion to the observed mean, and to measure the precision of our imputations mean relative error (MRE) which as a percentage representation computes the mean of differences between the absolute errors $|y - \hat{y}|$ divided by their target values y.

A.4 Additional results

A.4.1 Ablation study

Table 19 contains the results obtained within an ablation study done aiming to find the best configuration for an S4 layer in our proposed model. We present three different variants with capital letter notations. (A) represents our proposed DiffWave imputer with dilated convolutional layers. (B) represents a DiffWave imputer with a S4 layer in replacement of the dilated convolutional layer. (C) represents setting B and an extra S4 layer after the conditional data addition (SSSD$^{\text{S4}}$). Lastly, setting (D) represents the testing of our proposed SSSD$^{\text{S4}}$ under the same 20% BM setting conditions, however, with an unconditional training at 150,000 iterations, following the procedure proposed in Lugmayr et al. (2022b) for image inpainting.

Table 19: Results for the ablation study.

Setting	MAE	RMSE
A: Diffwave	0.0435±3e-3	0.1167±1e-2
B: SSSD^{S4} (single S4 layer)	0.0367±2e-3	0.0929±2e-2
C: SSSD^{S4}	**0.0324±3e-3**	**0.0832±8e-3**
D: RePaint (unconditional training)	0.1235±4e-3	0.2131±8e-3

The experiment was carried out in a BM scenario at 20% of missing values on the PTB-XL data set. Metrics were obtained at the 150,000 training iterations. We report an averaged MAE, and RMSE, over three trials for 10 samples generated for each sample of the test set. The results observed in Table 19 shows the metrics obtained for four different variants in our ablation study. The replacement of the dilated convolutions by S4 layers leads to a significant error reduction (setting B vs. setting A). The introduction of a second S4 layer again leads to a smaller but still consistent improvement over the setup with a single S4 layer (setting C vs. setting B). The unconditional training procedure proposed in Lugmayr et al. (2022b) is clearly inferior to the proposed conditional training (setting D vs. setting C), see also Figure 5 for a qualitative impression of the generated samples. This leads us to propose SSSD^{S4} (setting C) as default variant for further experiments.

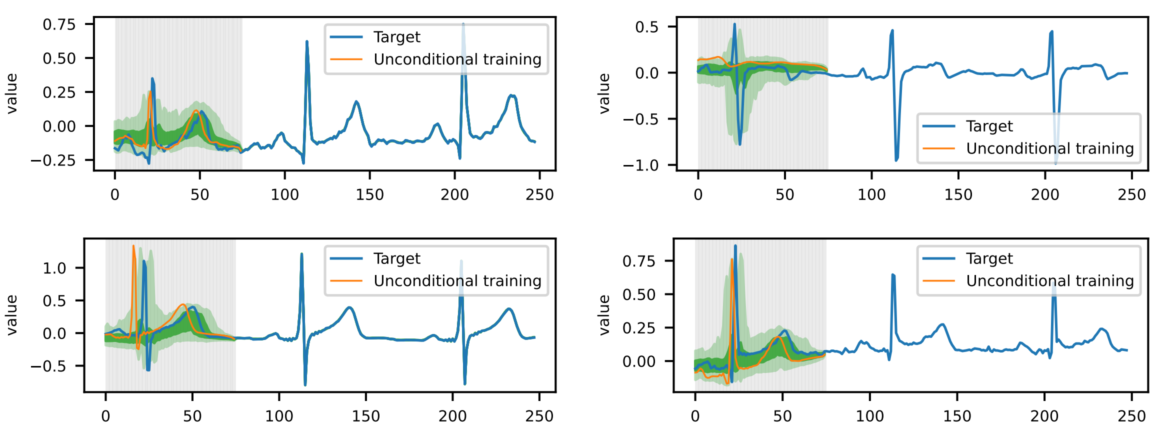

Figure 5: PTB-XL BM imputations for the leads I, V1, V4, and aVF of an ECG from a patient with sinus arrhythmia obtained from unconditional training (setting D).

A.4.2 Full imputation and forecasting results

In Table 20 to Table 24, we compile the full imputation and forecasting results including all available baseline results and missingness scenarios that were only presented in an excerpted fashion in the main text. In addition, Table 25 and Table 26 describe the number of training iterations used while training on each of the data sets.

Table 20: Imputations for RM, MNR, and BM scenarios on the PTB-XL data set.

Model	**MAE**	**RMSE**
	20% RM on PTB-XL	
Median	0.1040	0.2071
LAMC	0.0678	0.1309
DiffWaveD_0	0.0047±2e-5	0.0175±3e-4
DiffWaveD_1	0.0043±4e-4	0.0177±4e-4
CSDI	0.0038±2e-6	0.0189±5e-5
CSDI$^{\mathbf{S4}}$	**0.0031±1e-7**	0.0171±6e-4
SSSD$^{\text{SA}}D_0$	0.0052±3e-5	0.0229±4e-6
SSSD$^{\text{SA}}D_1$	0.0045±3e-7	0.0181±4e-6
SSSD$^{\text{S4}}D_0$	0.0044±2e-5	0.0137±1e-4
SSSD$^{\mathbf{S4}}\mathbf{D_1}$	0.0034±4e-6	**0.0119±1e-4**
	20% MNR on PTB-XL	
Median	0.1074	0.2157
LAMC	0.0759	0.1498
DiffWaveD_0	0.0482±1e-3	0.1209±8e-3
DiffWaveD_1	0.0250±1e-3	0.0808±5e-3
CSDI	0.0186±1e-5	0.0435±2e-4
CSDI$^{\text{S4}}$	0.0222±2e-5	0.0573±1e-3
SSSD$^{\text{SA}}D_0$	0.0202±1e-3	0.0612±1e-2
SSSD$^{\text{SA}}D_1$	0.0170±1e-4	0.0492±1e-2
SSSD$^{\text{S4}}D_0$	0.0116±2e-4	0.0251±7e-4
SSSD$^{\mathbf{S4}}\mathbf{D_1}$	**0.0103±3e-3**	**0.0226±9e-4**
	20% RM on PTB-XL	
Median	0.1252	0.2347
LAMC	0.0840	0.1171
DiffWaveD_0	0.0492±1e-3	0.1405±8e-3
DiffWaveD_1	0.0451±7e-4	0.1378±5e-3
CSDI	0.1054±4e-5	0.2254±7e-5
CSDI$^{\text{S4}}$	0.0792±2e-4	0.1879±1e-4
SSSD$^{\text{SA}}D_0$	0.0493±1e-3	0.1192±7e-3
SSSD$^{\text{SA}}D_1$	0.0435±3e-3	0.1167±1e-2
SSSD$^{\text{S4}}D_0$	0.0415±1e-3	0.1073±5e-3
SSSD$^{\mathbf{S4}}\mathbf{D_1}$	**0.0324±3e-3**	**0.0832±8e-3**

Table 21: RM Imputation results for the Electricity data set. Overall, CSDI^{S4} outperformed all the RM scenarios' baselines metrics even at high levels of missing data. All baseline results were collected from Du et al. (2022).

Model	**MAE**	**RMSE**	**MAE**
	10% RM on Electricity		
Median	2.056	2.732	110%
M-RNN	1.244	1.867	66.6%
GP-VAE	1.094	1.565	58.6%
BRITS	0.847	1.322	45.3%
Transformer	0.823	1.301	44.0%
SAITS	0.735	1.162	39.4%
SSSD^{S4}	**0.345±1e-4**	**0.554±5e-5**	**18.4%±5e-5**
	30% RM on Electricity		
Median	2.055	2.732	110%
M-RNN	1.258	1.876	67.3%
GP-VAE	1.057	1.571	56.6%
BRITS	0.943	1.435	50.4%
Transformer	0.846	1.321	45.3%
SAITS	0.790	1.223	42.3%
SSSD^{S4}	**0.407±5e-4**	**0.625±1e-4**	**21.8±0%**
	50% RM on Electricity		
Median	2.053	2.728	109%
M-RNN	1.283	1.902	68.7%
GP-VAE	1.097	1.572	58.8%
BRITS	1.037	1.538	55.5%
Transformer	0.895	1.410	47.9%
SAITS	0.876	1.377	46.9%
SSSD^{S4}	**0.532±1e-4**	**0.821±1e-4**	**28.5%±1e-4**

Table 22: MAE, MSE, MRE for PEMS-BAY and METR-LA data sets. All baseline results were collected from Cini et al. (2022).

Model	**MAE**	**MSE**	**MRE**
	25% RM on PEMS-Bay		
Mean	5.42±0.00	86.59±0.00	8.67±0.00
KNN	4.30±0.00	49.80±0.00	6.88±0.00
MF	3.29±0.01	51.39±0.64	5.27±0.02
MICE	3.09±0.02	31.43±0.41	4.95±0.02
VAR	1.30±0.00	6.52±0.01	2.07±0.01
rGAIN	1.88±0.02	10.37±0.20	3.01±0.04
BRITS	1.47±0.00	7.94±0.03	2.36±0.00
MPGRU	1.11±0.00	7.59±0.02	1.77±0.00
GRIN	**0.67±0.00**	**1.55±0.01**	**1.08±0.00%**
$SSSD^{S4}$	0.97±0.01	2.98±0.03	1.42±0.01
	25% RM on PEMS-Bay		
Mean	7.56±0.00	142.22±0.00	13.10±0.00
KNN	7.88±0.00	129.29±0.00	13.65±0.00
MF	5.56±0.03	113.46±1.08	9.62±0.05
MICE	4.42±0.07	55.07±1.46	7.65±0.12
VAR	2.69±0.00	21.10±0.02	4.66±0.00
rGAIN	2.83±0.01	20.03±0.09	4.91±0.01
BRITS	2.34±0.00	16.46±0.05	4.05±0.00
MPGRU	2.44±0.00	22.17±0.03	4.22±0.00
GRIN	**1.91±0.00**	**10.41±0.03**	**3.30±0.00**
$SSSD^{S4}$	2.83±0.02	21.95±0.14	5.59±0.08

Table 23: Forecasting results for the PTB-XL data set.

Model	**MAE**	**RMSE**
Median	0.134	0.273
CSDI	0.165±0.0009	0.302±0.0004
$CSDI^{S4}$	0.120±0.0002	0.246±0.0001
$SSSD^{SA}$	**0.087±0.008**	0.220±0.012
$SSSD^{S4}$	0.090±0.003	**0.219±0.006**

Table 24: Time series forecasting MAE and MSE on the ETTm1 data set.

Model	24	48	96	288	672
			TF on ETTm1 (MAE)		
LSTNet	1.170	1.215	1.542	2.076	2.941
LSTMa	0.629	0.939	0.913	1.124	1.555
Reformer	0.607	0.777	0.945	1.094	1.232
LogTrans	0.412	0.583	0.792	1.320	1.461
Informer†	0.371	0.470	0.612	0.879	1.103
Informer	0.369	0.503	0.614	0.786	0.926
Autoformer	0.403	**0.453**	**0.463**	**0.528**	**0.542**
SSSD$^{\mathbf{S4}}$	**0.361(6)**	0.479(8)	0.547(12)	0.648(10)	0.783(66)
			TF on ETTm1 (MSE)		
LSTNet	1.968	1.999	2.762	1.257	1.917
LSTMa	0.621	1.392	1.339	1.740	2.736
Reformer	0.724	1.098	1.433	1.820	2.187
LogTrans	0.419	0.507	0.768	1.462	1.669
Informer†	**0.306**	0.465	0.681	1.162	1.231
Informer	0.323	0.494	0.678	1.056	1.192
Autoformer	0.383	**0.454**	**0.481**	**0.634**	**0.606**
SSSD^{S4}	0.351(9)	0.612(2)	0.538(13)	0.797(5)	0.804(45)

Table 25: Training on the proposed PTB-XL data sets. training epochs (e) and iterations (i) on the proposed PTB-XL data set.

Model	PTB 248	PTB 1000
CSDI	200 (e)	200 (e)
CSDI^{S4}	200 (e)	200 (e)
DiffWave	150,000 (i)	150,000 (i)
SSSD^{SA}	150,000 (i)	150,000 (i)
SSSD^{S4}	150,000 (i)	150,000 (i)

Table 26: SSSD^{S4} training on benchmarking data sets. contains the SSSD^{S4} training iterations (i) on the benchmarking data sets across diverse baselines.

data set	**Setting**	**Iterations**
Electricity	Imputation 10%	150,000 (i)
Electricity	Imputation 30%	150,000 (i)
Electricity	Imputation 50%	150,000 (i)
PEMS-BAY	Imputation 25%	350,000 (i)
METR-LA	Imputation 25%	250,000 (i)
MuJoCo	Imputation 70%	232,000 (i)
MuJoCo	Imputation 80%	160,000 (i)
MuJoCo	Imputation 90%	150,000 (i)
Solar	Forecast 24	100,000 (i)
ETTm1	Forecast 24	212,000 (i)
ETTm1	Forecast 48	150,000 (i)
ETTm1	Forecast 96	250,000 (i)
ETTm1	Forecast 288	250,000 (i)
ETTm1	Forecast 672	250,000 (i)

B Multimedia Appendix

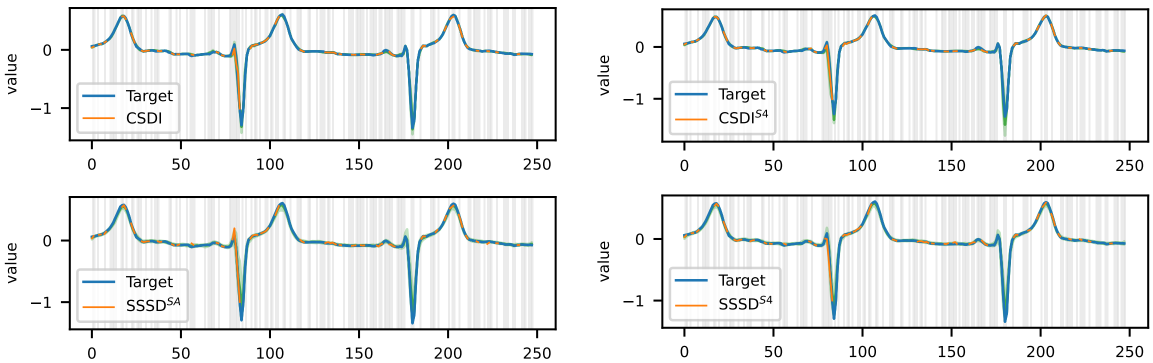

Figure 6: PTB-XL RM imputations of a V3 lead for an ECG from a patient with an anterior myocardial infarction (AMI). The figure shows 100 RM imputations on a single sample tested at 50% for four imputers on the PTB-XL data set. As clearly seen from the the plots above, all the diffusion models are highly capable of reconstructing time series, even when there is a high ratio of missing values in a RM scenario. Quantitatively, as seen in the main paper, we observed that CSDI^{S4} outperforms the rest of the models on this data set. The differences in terms of imputation quality can hardly be observed visually as the rest of the models present slight green shaded areas surrounding the orange imputed areas which represent the imputation quantiles.

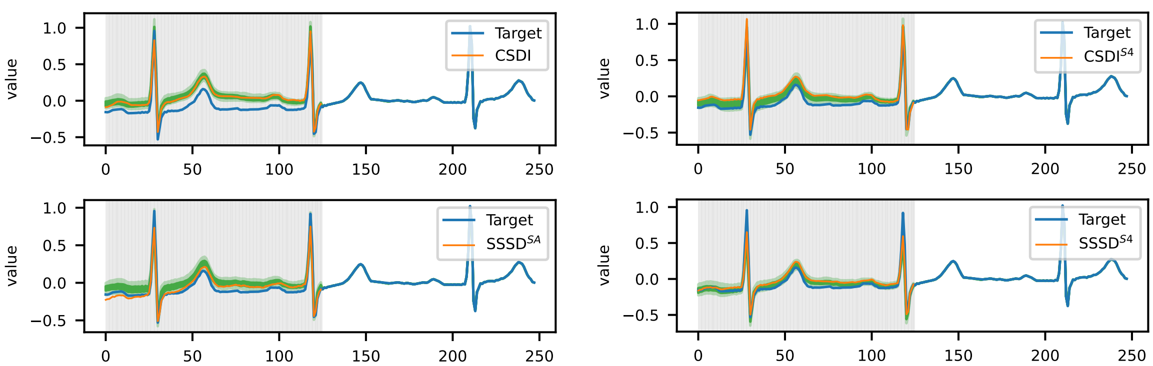

Figure 7: PTB-XL MNR imputations of a V4 lead for an ECG from a patient with an AV block (AVB). The figure shows 100 MNR imputations on a single sample tested at 50% for four imputers on the PTB-XL data set. As previously discussed, MNR assumes that the missing blocks are located at different time steps on each feature. Technically, in this setting, the diffusion models are capable of inferring the missing segments in a given channel from neighboring channels at the same time steps, which empirically seems to be a factor that enables an overall very good reconstruction across all models.

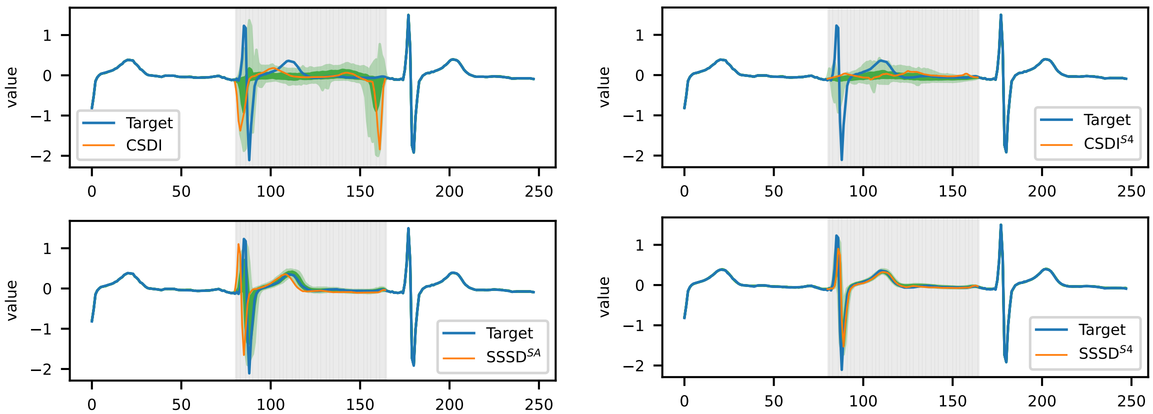

Figure 8: PTB-XL BM imputations for lead V2 for an ECG collected from a healthy patient. The figure shows four BM imputations tested at 30% on the PTB-XL data set reiterating the qualitative differences in imputation quality already demonstrated in the main text.

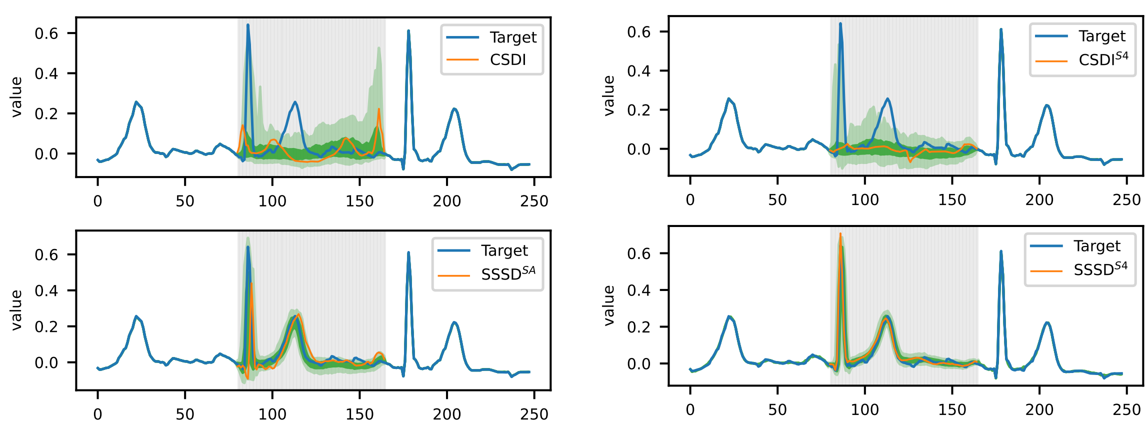

Figure 9: PTB-XL BM imputations for lead V6 for a normal heart condition. The figure shows four BM imputations tested at 30% on the PTB-XL data set.

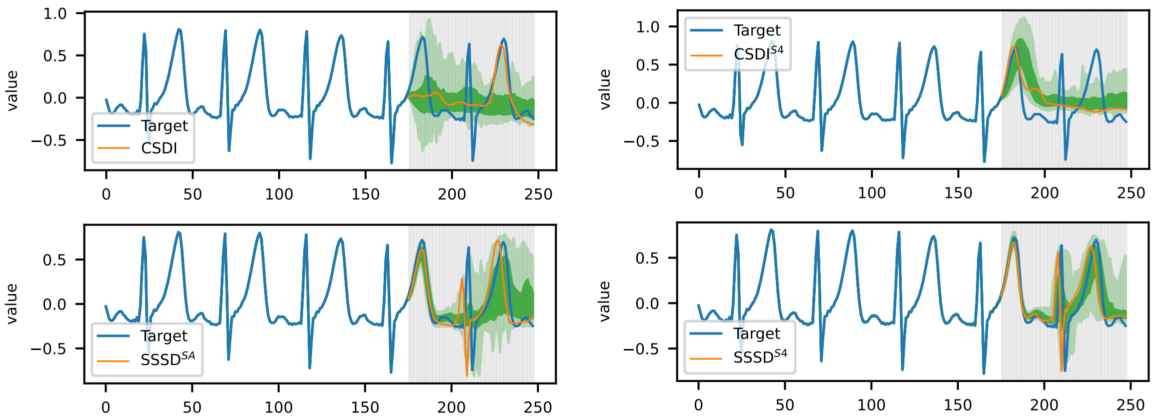

Figure 10: PTB-XL TF for lead V3 for an ECG from a patient with left ventricular hypertrophy. The figure shows 100 BM imputations on a single sample tested at 50% for four imputers on the PTB-XL data set. CSDI model is overall learning the trend that the series has, with a few correct feature generations as seen in the plot, nevertheless, we observe that the imputation falls outside the 0.05 and 0.95 quantile range, which is basically an outlier. CSDI$^{\mathrm{S4}}$ improves the quality of the generations, as its quantiles are less diverse and start to follow the signals patterns, however, it seems that after a certain number of steps, the learning decrease dramatically. On the contrary, SSSD$^{\mathrm{SA}}$ and SSSD$^{\mathrm{S4}}$ correctly capture the characteristics of the signal. In particular, SSSD$^{\mathrm{S4}}$ maintains a tight interquartile range even at longer forecasting horizons.

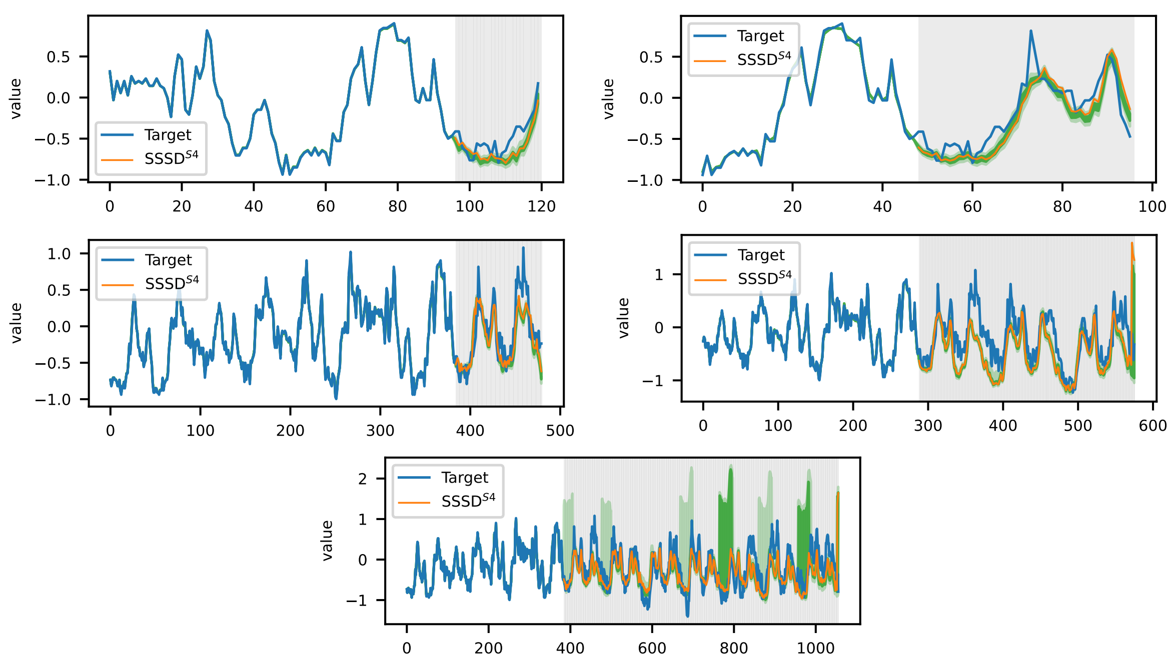

Figure 11: ETTm1 TF for the five different forecasting settings. From top to bottom 24, 48, 96, 288, 384 forecasting horizon targets. These plots display the complete sample including the conditional part.

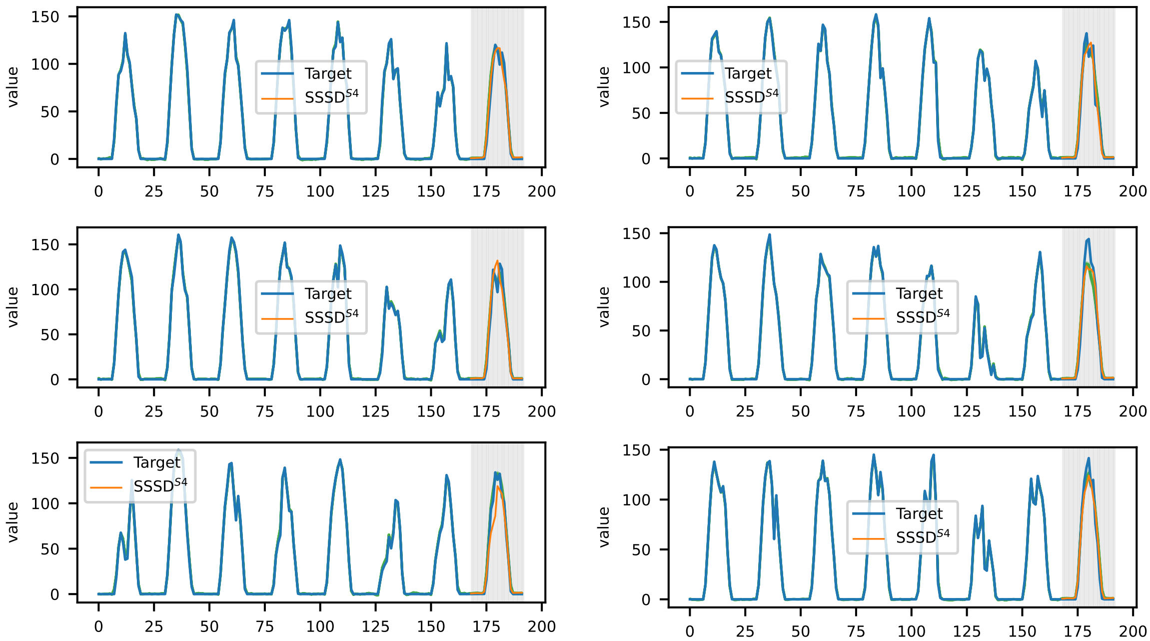

Figure 12: TF for six different channels of the Solar data set. These plots display the complete sample including the conditional part, see figure 13 for more detailed plots of the imputed area only.

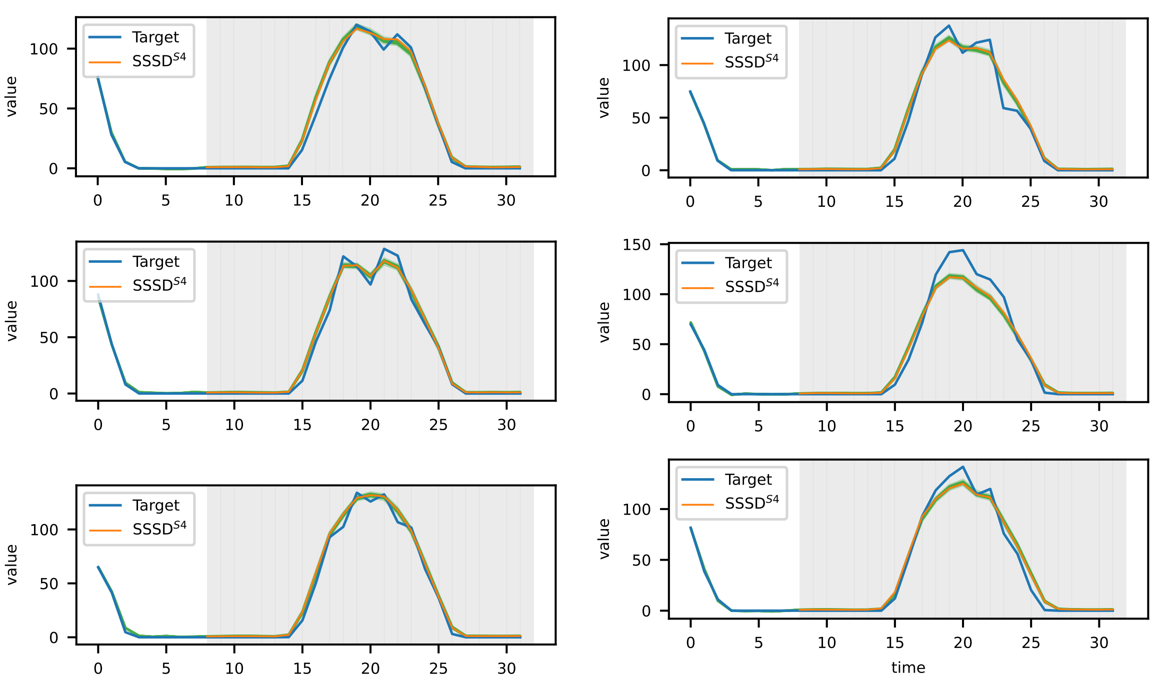

Figure 13: TF for six different channels of the Solar data set. These plots display only the imputed area.