Probabilistic Time Series Modeling with Decomposable Denoising Diffusion Model

Tijin Yan [1] **Hengheng Gong** [1] **Yongping He** [1] **Yufeng Zhan** [1] **Yuanqing Xia** [2 1]

Abstract

Probabilistic time series modeling based on generative models has attracted lots of attention because of its wide applications and excellent performance. However, existing state-of-the-art models, based on stochastic differential equation, not only struggle to determine the drift and diffusion coefficients during the design process but also have slow generation speed. To tackle this challenge, we firstly propose decomposable denoising diffusion model (D^3M) and prove it is a general framework unifying denoising diffusion models and continuous flow models. Based on the new framework, we propose some simple but efficient probability paths with high generation speed. Furthermore, we design a module that combines a special state space model with linear gated attention modules for sequence modeling. It preserves inductive bias and simultaneously models both local and global dependencies. Experimental results on 8 real-world datasets show that D^3M reduces RMSE and CRPS by up to 4.6% and 4.3% compared with state-of-the-arts on imputation tasks, and achieves comparable results with state-of-the-arts on forecasting tasks with only 10 steps.

1. Introduction

Time series modeling is very important because of its wide applications in various domains such as intelligent system monitoring, user behavior analysis and smart healthcare. With the increasing complexity of modern systems, traditional sequence modeling methods like ARIMA (Shumway et al., 2017) can not satisfy the requirements. Recently, deep learning-based methods have made significant progress with the development of hardware. Among these methods, probabilistic time series modeling methods have attracted lots of attention because they can explicitly model the data distribution and noise.

Generative models learn data distribution by constructing a probability path (Shaul et al., 2023) between predefined target distribution and data distribution. ScoreSDE (Song et al., 2020) proposes a general framework for describing the probability path based on the stochastic differential equation (SDE). However, in order to obtain the SDE corresponding to a certain probability path, we need to determine the drift and diffusion coefficients through the expectation and variance of the SDE, which may be difficult to calculate. In addition, the generative speed of ScoreSDE is very slow because it requires thousands of steps for simulating reverse SDE for high sampling quality. Many training-free methods based on high-order ODE solvers (Bao et al., 2022; Lu et al., 2022) have been proposed for fast simulation of the reverse SDE. However, few of them consider constructing new probability paths that combine high generation speed and sampling quality. Therefore, designing generative models that combines accurately data distribution modeling and high generation speed remains challenging.

Modeling of both local and global dependencies within a sequence is of great importance for generative models-based time series modeling tasks. Linear state space models (SSMs) (Gu et al., 2021), which introduce strong inductive bias on local dependencies, have been proved very effective for long-term sequence modeling. CSDI (Tashiro et al., 2021) uses self-attention modules to extract global information by modeling interactions at each step. However, the quadratic complexity and the weak inductive bias of attention make the training process expensive and slow to converge. In addition, above methods cannot simultaneously model local and global dependencies. Therefore, designing sequence modeling modules that combine low computational complexity, well inductive bias and dependencies modeling is challenging.

In order to address the issues mentioned above, we firstly propose decomposable denoising diffusion model (D^3M). It's a general framework that directly builds generative models based on the explicit solutions of the linear SDE. Com-

[1]School of Automation, Beijing Institute of Technology, Beijing, China. [2]Zhongyuan University of Technology, Zhengzhou, Henan, China. Correspondence to: Yufeng Zhan <yu-feng.zhan@bit.edu.cn>, Yuanqing Xia <xia_yuanqing@bit.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

pared with ScoreSDE (Song et al., 2020), the connection between D^{3}M and the probability path is more clear. It can be decomposed and learned by two separate processes: signal dissipation process and noise injection process. Secondly, D^{3}M unifies existing two large classes of generative models: denoising diffusion models (DDMs) and continuous flow-based generative models. Based on the framework, we design some new generative models with high generation speed and sampling quality. Thirdly, we combine exponential moving average (EMA), a special state space model with linear gated attention modules for sequence modeling. It preserves inductive bias, low complexity and simultaneously models both local and global dependencies. Fourthly, we evaluate D^{3}M on probabilistic time series imputation and forecasting tasks. The experimental results show that D^{3}M reduces RMSE and CRPS by up to 4.6% and 4.3% compared with state-of-the-arts on imputation tasks and achieves comparable results with state-of-the-art methods on forecasting tasks with only 10 steps.

2. Related works

2.1. Generative models

Probabilistic time series modeling methods have achieved great performance for various time series modeling tasks recently. The integration of variational autoencoders (VAEs) and Kalman filters (Krishnan et al., 2015), Gaussian process (Fortuin et al., 2020) and RNN (Fraccaro et al., 2016) have made great progress in various tasks. Flow-based generative models (Kobyzev et al., 2020) are also adopted by many methods (Rasul et al., 2020; De Brouwer et al., 2019). Recently, denoising diffusion models (Ho et al., 2020) have attracted lots of attention. They directly estimate the probability density with an unknown normalizing constant (Song & Kingma, 2021), also called *energy function*. Song et al. (2020) propose using linear SDEs to model the probability path between data and target distribution. DDM (Huang et al., 2023) also proposes SDE-based diffusion models based on a specific noise schema. Recently, many diffusion models (Yang et al., 2023; Lin et al., 2023) have been proposed for further improvement of data distribution modeling and speed acceleration.

2.2. Time series modeling

Time series modeling has attracted lots of attention in recent years because of its wide applications in various domains such as network traffic analysis and intelligent system monitoring. Some methods (Chen et al., 2018; Salvi et al., 2022; Kidger et al., 2020; Li et al., 2020) combine neural networks and continuous differential equations to model temporal dependencies. The training process based on the adjoint sensitivity method is very slow. TCN (Yan et al., 2020; van den Oord et al., 2016) adopts multi-layer dilated convolution to solve the problem mentioned above. Recently, structured state space models (SSMs) (Gu et al., 2021; 2022) have achieved great success for long sequence modeling. TIDER (Liu et al., 2022) decomposes time-series data into trend, seasonality, and noise terms based on matrix factorization. In addition, attention-based methods (Zhou et al., 2021; Du et al., 2023) have been proved very effective especially on large sequence datasets. However, the quadratic complexity makes the training and inference process very expensive. Recently, many methods (Alcaraz & Strodthoff, 2022; Kollovieh et al., 2024) that combine diffusion models have achieved great performance.

3. Methodology

In this section, we firstly introduce D^{3}M, the general framework for building generative models. Then, we introduce the network architecture for sequence modeling.

3.1. Decomposable denoising diffusion model

Without loss of generality, given a general linear SDE as

$$d\boldsymbol{X}_t = f(t)\boldsymbol{X}_t dt + g(t)d\boldsymbol{W}_t, \tag{1}$$

where $t \in [0, 1]$, $\boldsymbol{W}_t$ is the standard Wiener process, $f(t)\boldsymbol{X}_t$ and $g(t)$ are drift and diffusion coefficients. Then the explicit solution of Eq. (1) is

$$\boldsymbol{X}_t = e^{\int_0^t f(s)ds}\boldsymbol{X}_0 + \int_0^t e^{\int_s^t f(r)dr} g(s)d\boldsymbol{W}_s, \tag{2}$$

where $\boldsymbol{X}_0$ is the initial state. Detailed derivation can be found in chapter 4 of Applied SDEs (Särkkä & Solin, 2019). It's obvious that the expectation of the second term in Eq. (2) is $\mathbf{0}$. If we want to build a probability path that evolves from data distribution to target Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, the first term and the variance of the second term in Eq. (2) should approach to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ when $t \to 1$. Actually, many existing proposed score-based generative models (SGMs) based on SDEs (Song et al., 2020; Karras et al., 2022) design generative models following the above rules. However, the design of new generative models based on the SDE in Eq. (1) is inconvenient and not intuitive because f and g that satisfy the constraints mentioned above may be difficult to calculate. Why not directly build generative models based on the explicit solution of linear SDEs? This is exactly the inspiration of this paper.

3.1.1. Forward and reverse process

For the simplicity of symbols, we can rewrite Eq. (2) as the general form of the explicit solution of linear SDEs,

$$\boldsymbol{X}_t = \underbrace{\boldsymbol{X}_0 + \int_0^t \boldsymbol{h}(s)ds}_{\text{Signal dissipation}} + \underbrace{\int_0^t \boldsymbol{l}(s, \boldsymbol{W}_s)d\boldsymbol{W}_s}_{\text{Noise injection}}. \tag{3}$$

Table 1. Possible selections for $\boldsymbol{h}(t)$ or $\boldsymbol{H}(t)$ and $\boldsymbol{l}(t, \boldsymbol{W}_t)$ or β_t for D^{3}M.

TYPE	$\boldsymbol{h}(t)$	$\boldsymbol{H}_t$	CONSTRAINT 1	CONSTRAINT 2
CONSTANT	$\boldsymbol{\mu} - \boldsymbol{X}_0$	$(\boldsymbol{\mu} - \boldsymbol{X}_0)t$	$\boldsymbol{H}_0 = \boldsymbol{0}$	$\boldsymbol{H}_1 = \boldsymbol{\mu} - \boldsymbol{X}_0$
LINEAR	$\boldsymbol{a}t + \boldsymbol{b}$	$\frac{\boldsymbol{a}t^2}{2} + \boldsymbol{b}t$	$\boldsymbol{H}_0 = \boldsymbol{0}$	$\frac{\boldsymbol{a}}{2} + \boldsymbol{b} = \boldsymbol{\mu} - \boldsymbol{X}_0$
TRIGONOMETRIC	$\boldsymbol{a} \sin \frac{\pi t}{2} \boldsymbol{X}_0 + \boldsymbol{b}$	$-\frac{2\boldsymbol{a}}{\pi} \cos \frac{\pi t}{2} + \boldsymbol{b}t + \boldsymbol{c}$	$-\frac{2\boldsymbol{a}}{\pi} + \boldsymbol{c} = \boldsymbol{0}$	$\boldsymbol{b} + \boldsymbol{c} = \boldsymbol{\mu} - \boldsymbol{X}_0$
EXPONENTIAL	$e^{\boldsymbol{a}t} + \boldsymbol{b}$	$e^{\boldsymbol{a}t}/\boldsymbol{a} + \boldsymbol{b}t + \boldsymbol{c}$	$1/\boldsymbol{a} + \boldsymbol{c} = \boldsymbol{0}$	$e^{\boldsymbol{a}}/\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{c} = \boldsymbol{\mu} - \boldsymbol{X}_0$
TYPE	$\boldsymbol{l}(t, \boldsymbol{W}_t)$	β_t	CONSTRAINT1	CONSTRAINT2
SQRT	$\boldsymbol{L}$	$\sqrt{t}$	$\boldsymbol{\Sigma}_0 = \boldsymbol{0}$	$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$
LINEAR	$\sqrt{2}\boldsymbol{L}\boldsymbol{W}_t$	t	$\boldsymbol{\Sigma}_0 = \boldsymbol{0}$	$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$
POLY.	$\sqrt{3}\boldsymbol{L}t$	$\sqrt{t^3}$	$\boldsymbol{\Sigma}_0 = \boldsymbol{0}$	$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$
TRIGONOMETRIC	–	$\sin \frac{\pi t}{2}$	$\boldsymbol{\Sigma}_0 = \boldsymbol{0}$	$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$
EXPONENTIAL	$\boldsymbol{L}(e^{\boldsymbol{W}_t - 0.5t})/\sqrt{e-1}$	$\sqrt{(e^t - 1)/(e-1)}$	$\boldsymbol{\Sigma}_0 = \boldsymbol{0}$	$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}$

where $\boldsymbol{h}$ and $\boldsymbol{l}$ are two vector functions. The first two terms called *signal dissipation* process correspond to the first term in Eq. (2). The last term called *noise injection* process corresponds to the second term in Eq. (2). Different from DDM, the diffusion term can be any functions. Denote $\boldsymbol{H}_t = \int_0^t \boldsymbol{h}(t)dt, \int_0^t \boldsymbol{l}(s, \boldsymbol{W}_s)d\boldsymbol{W}_s \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_t)$, then it's easy to obtain the probability path of Eq. (3) is

$$q(\boldsymbol{X}_t|\boldsymbol{X}_0) \sim \mathcal{N}(\boldsymbol{X}_0 + \boldsymbol{H}_t, \boldsymbol{\Sigma}_t). \tag{4}$$

In this paper, we set the target distribution as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is positive definite and can be factorized as $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T$. In addition, assume that $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}\beta_t^2$ for simplicity, where β_t is a scalar function. In order to build generative models, we should select proper $\boldsymbol{h}$ and $\boldsymbol{l}$ such that $\lim_{t\to 0} \boldsymbol{H}_t = \boldsymbol{0}, \lim_{t\to 1} \boldsymbol{H}_t = \boldsymbol{\mu} - \boldsymbol{X}_0, \lim_{t\to 1} \beta_t = 1, \lim_{t\to 0} \beta_t = 0$.

Compared with previous models based on SDEs (Song et al., 2020) and ODEs (Lipman et al., 2022; Albergo & Eric, 2023), building generative models based on Eq. (3) is more convenient because it's easy to design $\boldsymbol{H}_t$ and $\boldsymbol{l}$ that satisfy the above constraints. We list some possible selections for these terms in Table 1. Note that we only need to know one of $\boldsymbol{h}(t)$ and $\boldsymbol{H}_t$ for signal dissipation process and one of $\boldsymbol{l}(t, \boldsymbol{W}_t)$ and $(\beta_t, \boldsymbol{\Sigma})$ for noise injection process to build generative models.

For the reverse process, given the forward probability path in Eq. (4) and time interval Δt, we can calculate the conditional distribution $q(\boldsymbol{X}_t|\boldsymbol{X}_{t-\Delta t})$ as

$$q(\boldsymbol{X}_t|\boldsymbol{X}_{t-\Delta t}) \sim \mathcal{N}(\boldsymbol{X}_{t-\Delta t} + \boldsymbol{H}_t - \boldsymbol{H}_{t-\Delta t}, \boldsymbol{\Sigma}(\beta_t^2 - \beta_{t-\Delta t}^2)). \tag{5}$$

Then according to the Bayesian formula, it's easy to obtain that $q(\boldsymbol{X}_{t-\Delta t}|\boldsymbol{X}_t, \boldsymbol{X}_0)$ also follows a Gaussian distribution $\mathcal{N}(\boldsymbol{M}, \boldsymbol{P})$, where

$$\boldsymbol{P} = \frac{\beta_{t-\Delta t}^2(\beta_t^2 - \beta_{t-\Delta t}^2)}{\beta_t^2}\boldsymbol{\Sigma}, \tag{6}$$

$$\boldsymbol{M} = \boldsymbol{X}_t - \boldsymbol{H}_t + \boldsymbol{H}_{t-\Delta t} - \frac{\beta_t^2 - \beta_{t-\Delta t}^2}{\beta_t}\boldsymbol{L}\boldsymbol{\varepsilon}, \tag{7}$$

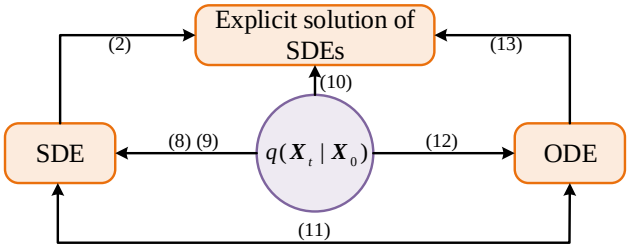


Figure 1. Connections between D^{3}M and existing models.

$\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Detailed derivation process can be found in Appendix A.1.

3.1.2. UNIFIED FRAMEWORK FOR CONTINUOUS FLOWS & DIFFUSION MODELS

When $\boldsymbol{l}(t, \boldsymbol{W}_t) = \boldsymbol{0}$, Eq. (3) is exactly the solution of ODEs. Many ODE-based generative models can be implemented under such conditions. Actually, D^{3}M is a general framework that unifies two large classes of generative models: ODE-based flow models (Lipman et al., 2022; Liu et al., 2023; Albergo & Eric, 2023; Heitz et al., 2023) and SDE-based diffusion models (Song et al., 2020; Lee et al., 2021).

Theorem 3.1. *If the probability path of a certain generative model is $q(\boldsymbol{X}_t|\boldsymbol{X}_0) \sim \mathcal{N}(\alpha_t\boldsymbol{X}_0, \beta_t^2\boldsymbol{I})$, where $\lim_{t\to 0} \alpha_t = 1, \lim_{t\to 0} \beta_t = 0$, then D^{3}M is equivalent to existing ODE-based flow models and SDE-based denoising diffusion models with proper settings of $\boldsymbol{H}(t)$ and $\boldsymbol{l}(s, \boldsymbol{W}_s)$.*

Proof: For the following probability path: $q(\boldsymbol{X}_t|\boldsymbol{X}_0) \sim \mathcal{N}(\alpha_t\boldsymbol{X}_0, \beta_t^2\boldsymbol{I})$, where $t \in [0, 1], \lim_{t\to 0} \alpha_t = 1, \lim_{t\to 0} \beta_t = 0$, it has been proved by VDM (Kingma et al., 2021) that $q(\boldsymbol{X}_t|\boldsymbol{X}_0)$ is governed by the SDE

$$d\boldsymbol{X}_t = f(t)\boldsymbol{X}_t dt + g(t)d\boldsymbol{W}_t, \tag{8}$$

where

$$f(t) = \frac{d\log\alpha_t}{dt}, \quad g^2(t) = \frac{d\beta_t^2}{dt} - 2\frac{d\log\alpha_t}{dt}\beta_t^2. \tag{9}$$

Table 2. The settings that make D^{3}M share the same probability path with original methods.

NAME	$\boldsymbol{H}_t$	β_t^2	TARGET DIST.	PARAMETERS
VP SDE (SONG ET AL., 2020)	$(e^{-\frac{1}{2}\int_0^t \beta(s)ds}-1)\boldsymbol{X}_0$	$1-e^{-\int_0^t \beta(s)ds}$	$\mathcal{N}(\mathbf{0},\boldsymbol{I})$	$\beta(t)=0.01+19.9t$
PRIORGRAD (LEE ET AL., 2021)	$(e^{-\frac{1}{2}\int_0^t \beta(s)ds}-1)(\boldsymbol{X}_0-\boldsymbol{\mu})-\boldsymbol{\mu}$	$1-e^{-\int_0^t \beta(s)ds}$	$\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma})$	$\boldsymbol{\mu},\boldsymbol{\Sigma}=f(\boldsymbol{X}_0)$
VE SDE (SONG ET AL., 2020)	$\mathbf{0}$	$\sigma_t^2-\sigma_0^2$	$\mathcal{N}(\mathbf{0},\sigma_1^2\boldsymbol{I})$	$\sigma_t=0.01*5000^t$
FLOWMAT. (LIPMAN ET AL., 2022)	$-t\boldsymbol{X}_0$	$(\sigma+(1-\sigma)t)^2$	$\mathcal{N}(\mathbf{0},\boldsymbol{I})$	$\sigma=1e-4$
RECFLOW (LIU ET AL., 2023)	$-t\boldsymbol{X}_0$	t^2	$\mathcal{N}(\mathbf{0},\boldsymbol{I})$	–
STO. INT. (ALBERGO & ERIC, 2023)	$(\cos\frac{\pi}{2}t-1)\boldsymbol{X}_0$	$\sin^2\frac{\pi t}{2}$	$\mathcal{N}(\mathbf{0},\boldsymbol{I})$	–

Substitute Eq. (8) and Eq. (9) into Eq. (2), we can get the coefficients in D^{3}M are

$$\boldsymbol{H}_t=(\frac{\alpha_t}{\alpha_0}-\boldsymbol{I})\boldsymbol{X}_0,\qquad \boldsymbol{l}(s,\boldsymbol{W}_s)=\frac{\alpha_t g(s)}{\alpha_s}. \tag{10}$$

In addition, ScoreSDE (Song et al., 2020) also proposes probability flow ODE that shares the same probability path with Eq. (8), that is

$$\frac{d\boldsymbol{X}_t}{dt}=f(t)\boldsymbol{X}_t-\frac{1}{2}g^2(t)\nabla_{\boldsymbol{X}_t}\log q_t(\boldsymbol{X}_t). \tag{11}$$

Substitute Eq. (9) into this equation and we can get

$$\begin{aligned}\frac{d\boldsymbol{X}_t}{dt}&=\frac{d\log\alpha_t}{dt}\boldsymbol{X}_t+\frac{1}{2}[\frac{d\beta_t^2}{dt}-2\frac{d\log\alpha_t}{dt}\beta_t^2]\frac{\boldsymbol{X}_t-\alpha_t\boldsymbol{X}_0}{\beta_t^2}\\&=\frac{\dot\beta_t}{\beta_t}(\boldsymbol{X}_t-\alpha_t\boldsymbol{X}_0)+\dot\alpha_t\boldsymbol{X}_0.\end{aligned} \tag{12}$$

Compare Eq. (12) and Eq. (3), it's easy to obtain that

$$\boldsymbol{H}_t=\int_0^t\frac{\dot\beta_t}{\beta_t}(\boldsymbol{X}_t-\alpha_t\boldsymbol{X}_0)+\dot\alpha_t\boldsymbol{X}_0dt,\quad \boldsymbol{l}(s,\boldsymbol{W}_s)=\mathbf{0}. \tag{13}$$

In the end, the connections between D^{3}M and existing generative models built on SDEs and ODEs are shown in Fig. 1. D^{3}M provides a new perspective to build various generative models. It indicates that existing generative models based on SDEs and ODEs can be transferred to D^{3}M with proper settings of $\boldsymbol{H}_t$ and $\boldsymbol{l}(s,\boldsymbol{W}_s)$ from the arrows pointing to D^{3}M in Fig. 1. Furthermore, the generative models constructed based on SDEs, ODEs or explicit solutions of SDEs are equivalent to each other as long as their probability paths are the same. Besides, the variance of target distribution of above models is limited to diagonal matrix, the target distribution of D^{3}M is more general and can be any multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$.

Based on Theorem 3.1, we list the settings that make D^{3}M share the same probability path with original methods in Table 2. Note that some target distributions in Table 2 have been approximated. For example, the target distribution of VE SDE (Song et al., 2020) should be $\mathcal{N}(\boldsymbol{X}_0,\boldsymbol{\sigma}_1^2\boldsymbol{I})$. Since $\boldsymbol{\sigma}_1$ is big enough, it can be approximated by $\mathcal{N}(\mathbf{0},\boldsymbol{\sigma}_1^2\boldsymbol{I})$, which is consistent with the original paper. It's obvious that D^{3}M is a general framework for existing SDE-based denoising diffusion models and ODE-based flow models.

Algorithm 1 Unconditional training procedure of D^{3}M.

Input: Dataset $\mathcal{D}$, target distribution $\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$.
Select $\boldsymbol{h}(t)$ and $\boldsymbol{l}(t,\boldsymbol{W}_t)$ or β_t.
Set $\boldsymbol{\psi}=$[coefficients of $\boldsymbol{h}(t)$]
while not converged **do**
  Sample $\boldsymbol{X}_0$ from $\mathcal{D}$.
  $s\sim\text{Uniform}(0,1),\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$.
  Calculate $\boldsymbol{H}_s$ and β_s, obtain $\boldsymbol{L}$ with $\boldsymbol{\Sigma}=\boldsymbol{L}^T\boldsymbol{L}$.
  $\boldsymbol{X}_s=\boldsymbol{X}_0+\boldsymbol{H}_s+\beta_s\boldsymbol{L}\boldsymbol{\varepsilon}$
  $\boldsymbol{\psi}_\theta,\boldsymbol{\varepsilon}_\theta=\text{Net}_\theta(\boldsymbol{X}_s,s)$.
  Calculate loss $\mathcal{L}(\theta)$ according to Eq. (14).
  Take gradient descent step on $\nabla_\theta\mathcal{L}_\theta$
end while

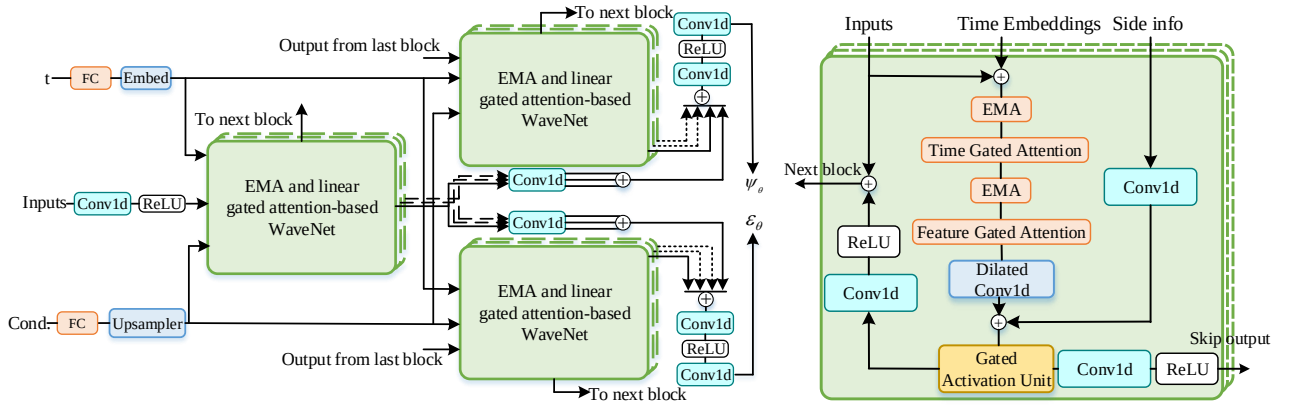
Algorithm 2 Unconditional sampling procedure of D^{3}M.

Initialization: $\boldsymbol{X}_1\sim\mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma}),\boldsymbol{\Sigma}=\boldsymbol{L}^T\boldsymbol{L},t=1,\Delta t$
while $t>0$ **do**
  $\boldsymbol{\psi}_\theta,\boldsymbol{\varepsilon}_\theta=\text{Net}_\theta(\boldsymbol{X}_t,t)$.
  Calculate $\boldsymbol{H}_t^\theta=\int_0^t h(s)dt$ based on $\boldsymbol{\psi}_\theta$.
  Calculate the mean $\boldsymbol{M}$ and variance $\boldsymbol{P}$ according to Eq. (7) and Eq. (6).
  Denote the standard deviation as $\boldsymbol{p}=\boldsymbol{P}^{1/2}$.
  $\boldsymbol{X}_t=\boldsymbol{M}+\boldsymbol{\varepsilon}\boldsymbol{p},\boldsymbol{\varepsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})$.
  $t=t-\Delta t$.
end while
Output: $\boldsymbol{X}_0$

3.1.3. LOSSES, TRAINING AND SAMPLING

Given Eq. (7), we need to determine $\boldsymbol{H}_t$ and $\boldsymbol{\varepsilon}$ at each step in the reverse process. Different from Flow Matching (Lipman et al., 2022), we do not directly learn the vector field $\boldsymbol{h}(t)$. Denote the list of coefficients of $\boldsymbol{h}(t)$ as $\boldsymbol{\psi}$. For example, $\boldsymbol{\psi}=[\boldsymbol{a},\boldsymbol{b}]$ when the type of $\boldsymbol{h}(t)$ is *Linear* in Table 1. Then we directly construct networks $\boldsymbol{\psi}_\theta$ to match $\boldsymbol{\psi}$, that is $\mathcal{L}_{SD}=\|\boldsymbol{\psi}_\theta-\boldsymbol{\psi}\|^2$.

As for the noise $\boldsymbol{\varepsilon}$ in Eq. (7), we construct a network $\boldsymbol{\varepsilon}_\theta$ to learn $\boldsymbol{\varepsilon}$, that is $\mathcal{L}_{NM}=\|\boldsymbol{\varepsilon}_\theta(\boldsymbol{X}_t)-\boldsymbol{\varepsilon}\|^2$. It is similar with the loss function in DDPM (Ho et al., 2020). We can also use score matching loss (Hyvärinen & Dayan, 2005) or denoising score matching loss (Vincent, 2011) for the learning of the noise injection process. Actually, we can prove that



(a) Overall architecture (b) EMA and linear gated attention-based WaveNet

Figure 2. (a) shows the overall architecture of Net_θ when used for conditional generation tasks like time series forecasting and imputation. (b) shows the architecture of a single block of EMA and linear gated attention-based WaveNet.

Theorem 3.2. *If the probability path of the forward process follows Eq. (4) and the learned $\psi_\theta = \psi$, then the losses mentioned above (Hyvärinen & Dayan, 2005; Vincent, 2011; Ho et al., 2020) are equivalent to each other with proper settings of weighting functions.*

We defer the proof in Appendix A.2. In this paper, we set the loss of D^{3}M as the weighted sum of $\mathcal{L}_{NM}$ and $\mathcal{L}_{SD}$,

$$\mathcal{L}_\theta = \omega_1 \mathcal{L}_{NM} + \omega_2 \mathcal{L}_{SD}, \tag{14}$$

where ω_1 and ω_2 are weighting functions. In this paper, we use a single neural network Net_θ with two output branches to simultaneously learn ψ_θ and ε_θ. The design of Net_θ for sequence modeling can be found in Section 3.2.

The general training and sampling process of D^{3}M are shown in Algorithm 1 and Algorithm 2. In addition, we also design a weighted incremental sampler when the type of $\boldsymbol{h}(t)$ is *Constant*. Besides, we find that $\boldsymbol{L}_{SD}$ can not guarantee that the learned coefficients satisfy the constraints in Table 1. In order to solve this problem, we propose some modified samplers to ensure the constraints are satisfied in Appendix A.3.

3.2. Network architecture of Net_θ

Various sequence modeling tasks like forecasting and imputation can be viewed as the conditional generation process of D^{3}M. Therefore, we design Net_θ as Fig. 2. It comprises two branches, with the outputs of these branches ψ_θ and ε_θ separately. Furthermore, these two branches share shallow features extracted from EMA and linear gated attention-based WaveNet. As shown in Fig. 2 (b), the backbone of the network is WaveNet (van den Oord et al., 2016). Different from existing architecture, we firstly use a special state space model, EMA, to model the temporal and local dependencies. Then we use time-oriented and feature-oriented linear gated attention modules to further extract information across time and features from different channels. This ensemble of components collectively equips the network with adaptability, flexibility, and high-performance capabilities, enabling it to effectively support a variety of tasks.

3.2.1. EMA

Exponential moving average strategy is one of the common methods for sequence modeling. Actually, it can be viewed as a special state space model (SSM) (Ma et al., 2022). Firstly, set the input of the SSM as the linear transformation of $\boldsymbol{x}_k \in \mathbb{R}^d$, that is $\boldsymbol{u}_k^j = \boldsymbol{\xi}_j \boldsymbol{x}_k^j$, $\boldsymbol{\xi}_j \in \mathbb{R}^h, \boldsymbol{u}_k^j \in \mathbb{R}^h$, $j \in \{1, 2, \cdots, d\}$. Then, consider the following SSM,

$$\boldsymbol{h}_k^j = \boldsymbol{\lambda}_j \odot \boldsymbol{u}_k^j + (1 - \boldsymbol{\lambda}_j \odot \boldsymbol{\delta}_j) \odot \boldsymbol{h}_k^{j-1}, \quad \boldsymbol{y}_k^j = \boldsymbol{\eta}_j^T \boldsymbol{h}_k^j, \tag{15}$$

where $\boldsymbol{\eta}_j, \boldsymbol{\lambda}_j, \boldsymbol{\delta}_j \in \mathbb{R}^h$. $\boldsymbol{\lambda}$ is the weighting factor that controls the weights of previous and current observations. $\boldsymbol{\delta}$ is the damping factor where each element is between 0 and 1. $\boldsymbol{\eta}$ is a matrix that transforms the h-dimensional states to single-dimensional output. $\odot$ represents Hadamard product. Denote $\boldsymbol{m} = (\boldsymbol{1} - \boldsymbol{\lambda} \odot \boldsymbol{\delta})$, then combine the SSM from all d channels and with proper derivation, we can get $\boldsymbol{y}_{1:k} = \mathcal{K} * \boldsymbol{x}_{1:k} + \boldsymbol{\eta}^T \boldsymbol{m}^T \odot \boldsymbol{h}_0$, where $*$ represents convolution operator, $\mathcal{K} = (\boldsymbol{\eta}^T(\boldsymbol{\lambda} \odot \boldsymbol{\xi}), \boldsymbol{\eta}^T(\boldsymbol{m} \odot \boldsymbol{\lambda} \odot \boldsymbol{\xi}), \cdots, \boldsymbol{\eta}^T(\boldsymbol{m}^k \odot \boldsymbol{\lambda} \odot \boldsymbol{\xi}))$. $\boldsymbol{K}$ can be easily computed by the Vandermonde product and the convolution can be efficiently computed according to the convolution theorem. Furthermore, compared with S4, the damped EMA does not need the Hippo framework (Gu et al., 2020) for the initialization of the state matrices.

3.2.2. Linear Gated Attention

In this paper, we use time and feature-oriented linear gated attention (Hua et al., 2022) to model temporal and feature dependencies. The z-dimensional shared representations from sequential input are computed as $\boldsymbol{Z} = f_{silu}(\text{FC}_1(\boldsymbol{y}_{1:k}))$.

f_{silu} is the self-gated activation function, FC_1 is a fully connected layer. Then the query, key and value can be calculated as $\boldsymbol{Q} = \boldsymbol{W}_Q \odot \boldsymbol{Z} + \boldsymbol{b}_Q, \boldsymbol{K} = \boldsymbol{W}_K \odot \boldsymbol{Z} + \boldsymbol{b}_K, \boldsymbol{V} = f_{silu}(\text{FC}_2(\boldsymbol{x}_{1:k}))$, where $\boldsymbol{W}_Q, \boldsymbol{W}_K, \boldsymbol{b}_Q, \boldsymbol{b}_K$ are learnable parameters. $\boldsymbol{Q}, \boldsymbol{K} \in \mathbb{R}^{k\times z}, \boldsymbol{V} \in \mathbb{R}^{k\times v}$.

Then the output of the attention is $\boldsymbol{O} = f^2_{relu}(\mathbf{Q}\mathbf{K}^T + \boldsymbol{b}_{pos})\boldsymbol{V}$, where f_{relu} is ReLU function, $\boldsymbol{b}_{pos}$ is the position embedding. The output of the gated attention is sent to a gated unit, that is $\boldsymbol{\gamma} = f_{silu}(\text{FC}_3(\boldsymbol{y}_{1:k}))$, $\boldsymbol{r} = f_{sigmoid}(\text{FC}_4(\boldsymbol{y}_{1:k}))$, $\hat{\boldsymbol{H}} = f_{silu}(\text{FC}_5(\boldsymbol{y}_{1:k}) + \text{FC}_6(\boldsymbol{\gamma} \odot \boldsymbol{O}))$, $\boldsymbol{H} = \boldsymbol{r} \odot \hat{\boldsymbol{H}} + (1 - \boldsymbol{r}) \odot \boldsymbol{x}_{1:k}$. FC_* are all fully connected layers. $f_{sigmoid}$ is the sigmoid activation function.

In this paper, we use single-head attention because it has been proved $\boldsymbol{\gamma} \odot \boldsymbol{O}$ is equivalent to multi-head attention under proper conditions (Ma et al., 2022). Then, we can split the queries, keys and values into $n = k/c$ chunks, where c is a small number that represents the chunk length. With this method, the complexity of the gated attention will reduce to linear, that is $O(nc^2) = O(kc)$.

4. Experiments

We use probabilistic time series imputation and forecasting tasks to evaluate the proposed model.

4.1. Probabilistic time series imputation

4.1.1. Datasets and experimental settings

For time series imputation tasks, we use the PhysioNet Challenge 2012 and Air quality for evaluation. Detailed description of these datasets can be found in Appendix B.1.

In addition, we denote the samples from data distribution as $\boldsymbol{X}_0$. We use the mask matrix $\boldsymbol{M}$ to indicate whether each element in the time series is observable. Then we randomly select any time t in $[0, 1]$ and calculate $\boldsymbol{X}_t$ based on the masked initial value $\boldsymbol{X}_0 \odot \boldsymbol{M}$. We set the input for Net_θ as $[\boldsymbol{X}_0 \odot \boldsymbol{M}, \boldsymbol{X}_t, \boldsymbol{M}]$. The condition term is the collection of some extra covariates like embeddings of observed time and embeddings of features. The number of steps for inference is set as 10 for all experiments. We use the first two types of $\boldsymbol{h}(t)$ and β_t in Table 1 for evaluation because their probability paths are very simple. We set $\boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = \boldsymbol{I}$ for simplicity. For D^3M with *Linear* type for $\boldsymbol{h}(t)$, we set $\boldsymbol{a} = -\boldsymbol{X}_0, \boldsymbol{b} = -\frac{\boldsymbol{X}_0}{2}$. The settings for weighting function in the loss can be found in Appendix C.1. We use the grid-search method for the hyper-parameters of EMA and linear gated attention module. Specifically, we set candidates of h as $\{8, 12, 16\}$, candidates of z as $\{32, 64, 96\}$, candidates of v as $\{128, 160, 256\}$. The batch size and epochs are set as 16 and 300, separately. In addition, we use a multi-step learning rate scheduler which decays the learning rate at 75% and 90% of all epochs. In the inference process, we set the number of generated samples as 100 for all datasets.

Table 3. Comparing RMSE for probabilistic time series imputation tasks (lower if better). The mean and standard error are reported by three runs.

Method	Physionet			Air quality
	10% missing	50% missing	90% missing	
V-RIN	0.637±0.021	0.712 ±0.018	0.934±0.012	40.02±1.03
BRITS	0.619±0.018	0.701±0.021	0.847±0.021	24.28±0.65
SSGAN	0.607±0.034	0.758 ±0.025	0.830±0.009	–
RDIS	0.635±0.018	0.747 ±0.013	0.922±0.018	37.25±0.31
CSDI	0.531±0.009	0.668±0.007	0.834±0.006	19.21±0.13
CSBI	0.547±0.019	0.649 ±0.009	0.837±0.012	19.07±0.18
SSSD	0.459±0.001	0.632±0.004	0.824±0.003	18.77±0.08
TS-Diff	0.523±0.016	0.679±0.009	0.845±0.007	19.06±0.14
SAITS	0.461±0.009	0.636±0.005	0.819±0.002	18.68±0.13
TIDER	0.486±0.006	0.659±0.009	0.833±0.005	18.94±0.21
D^3M(Constant-Sqrt)	**0.438±0.003**	**0.615±0.012**	0.814±0.002	18.19±0.18
D^3M(Constant-Linear)	0.441±0.002	0.618±0.007	**0.803±0.003**	**18.13±0.23**
D^3M(Linear-Sqrt)	0.481±0.004	0.645±0.006	0.815±0.001	20.79±0.17
D^3M(Linear-Linear)	0.447±0.002	0.646±0.007	0.810±0.002	20.85±0.14

Table 4. Comparing CRPS for probabilistic time series imputation tasks (lower is better). The mean and standard error are reported by three runs.

Method	Physionet			Air quality
	10% missing	50% missing	90% missing	
GP-VAE	0.582±0.003	0.796±0.004	0.998±0.001	0.402±0.009
V-RIN	0.814±0.004	0.845±0.002	0.932±0.001	0.534±0.013
CSDI	0.242±0.001	0.336±0.002	0.528±0.003	0.108±0.001
CSBI	0.247±0.003	0.332 ±0.003	0.527±0.006	0.110±0.002
SSSD	0.233±0.001	0.331±0.002	0.522±0.002	0.107±0.001
TS-Diff	0.249±0.002	0.348±0.004	0.541±0.006	0.118±0.003
D^3M(Constant-Sqrt)	**0.223±0.001**	0.327±0.003	0.520±0.001	0.106±0.002
D^3M(Constant-Linear)	0.229±0.003	**0.325±0.002**	0.514±0.003	**0.102±0.001**
D^3M(Linear-Sqrt)	0.236±0.002	0.328±0.001	0.516±0.002	0.128±0.003
D^3M(Linear-Linear)	0.231±0.003	0.332±0.002	**0.511±0.003**	0.129±0.001

4.1.2. Baselines and metrics

We divide existing time series imputation methods into two categories. The first category is deterministic imputation methods which can only generate deterministic outputs for certain inputs: BRITS (Cao et al., 2018), RDIS (Choi et al., 2023), SSGAN (Miao et al., 2021), TIDER (Liu et al., 2022), and SAITS (Du et al., 2023). The second category of methods are probabilistic imputation methods: GP-VAE (Fortuin et al., 2020), V-RIN (Mulyadi et al., 2021), CSDI (Tashiro et al., 2021), CSBI (Chen et al., 2023), SSSD (Alcaraz & Strodthoff, 2022), and recently proposed TS-Diff (Kollovieh et al., 2024). For deterministic methods, we adopt root mean square error (RMSE) for comparison. For probabilistic imputation methods, we use RMSE and continuous ranked probability score (CRPS) for evaluation, which is a common metric used that assess the congruence between the estimated probability distribution and observed values.

4.1.3. Results

The comparison of D^3M and the methods mentioned above are shown in Table 3 and Table 4. We report the mean and standard values of these methods and all the results

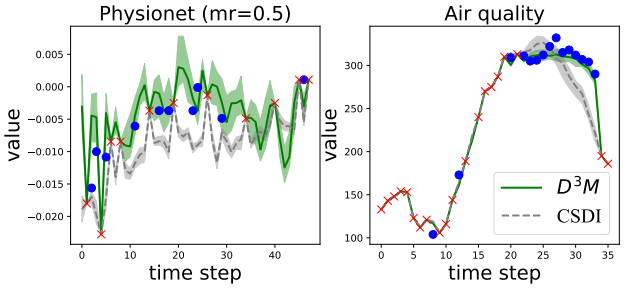


Figure 3. Examples of probabilistic imputation for Physionet with missing rate 50% and Air quality. The red points represent observed data, the blue points represent ground truth values to be imputed. The median values of imputations are shown in line. 50% of distribution intervals are shown in the shade.

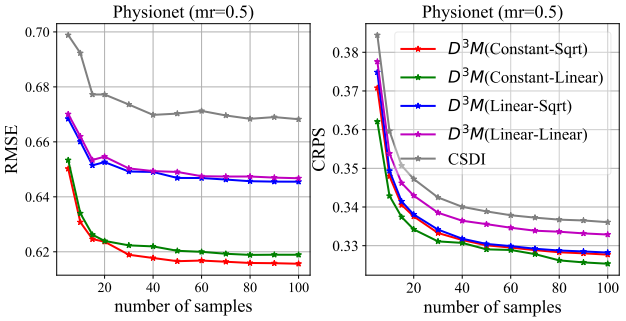


Figure 4. The effect of number of samples for prediction (Lower is better).

are obtained by three runs. We give 4 variants of D^{3}M for time series imputation tasks. Note that *D^3M(type1-type2)* represents we use *type1* for $\boldsymbol{h}(t)$ and *type2* for β_t. The best results for each dataset are shown in bold while the second best are marked with underline.

Compared with V-RIN and BRITS that adopt recurrent units for time series imputation, our method incorporate EMA into linear gated attention modules so that it can simultaneously model local dependencies and global dependencies. Compared with GP-VAE, V-RIN and SSGAN, CSDI significantly improves the imputation performance. In addition, CSBI adopts Schrödinger bridge algorithm for sequence modeling. SSSD and TS-Diff are also diffusion models for imputaion. It indicates the superiority of DDMs over other generative models. However, CSDI requires 50 to 100 steps for inference, CSBI requires 100 steps for inference. In addition, the quadratic complexity of attention used in CSDI and CSBI makes the training speed slow. Compared with CSDI and CSBI, our method requires only 10 steps for the generative process. In addition, the EMA module used in our methods introduces well inductive bias and the complexity of EMA and gated attention mechanism are all linear. In our experiments, D^{3}M reduces the inference time by up to 87.7% compared with CSDI. Under the settings of missing ratios 10%, 50%, 90% on Physionet and Air Quality, it reduces RMSE by 4.6%, 2.7%, 2.0% and 2.9% separately and it reduces CRPS by 4.3%, 1.9%, 2.1% and

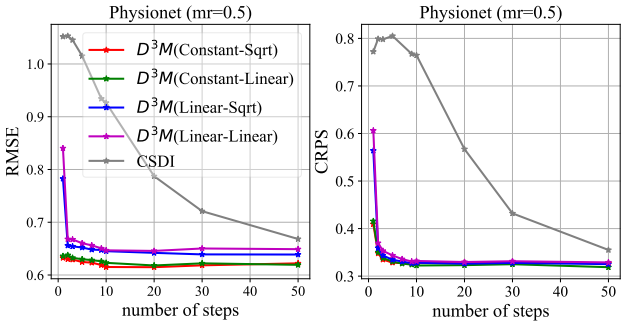


Figure 5. The effect of number of steps for prediction (Lower is better).

Table 5. Ablation experiments of D^{3}M (Constant-Sqrt) on Physionet with missing rate 0.5.

EMA	TIME GATED ATT.	FEAT. GATED ATT.	RMSE	CRPS
			1.055	0.740
		✔	0.915	0.586
	✔		0.714	0.434
	✔	✔	0.636	0.330
✔			0.649	0.348
✔		✔	0.628	0.332
✔	✔		<u>0.621</u>	<u>0.329</u>
✔	✔	✔	**0.615**	**0.327**

4.6% separately. It suggests that D^{3}M can estimate the probability distribution more accurately than other probabilistic methods. Among the four variants, we can observe that the choice of $\boldsymbol{h}(t)$ has a greater impact on performance compared with the choice of β_t. In addition, we should select type *Constant* for $\boldsymbol{h}(t)$ for settings with missing rates of 10% and 50%.

As shown in Fig. 3, we compare the imputation results of CSDI and D^{3}M (Constant-Linear) for different settings. There are two typical missing patterns shown in these two figures: random missing and block missing. It's obvious that our method can provide more accurate estimation of the values to be imputed in these two scenarios compared with CSDI. More results can be found in Appendix C.2.

4.1.4. ABLATION EXPERIMENTS

In addition, we also explore the impact of some important hyperparameters on the performance. Fig. 4 shows the effect of the number of samples on imputation performance. With the increasing of the number of samples, the imputation performance becomes better. All the four variants of D^{3}M are better than CSDI. In addition, D^{3}M which selects *Constant* type for $\boldsymbol{h}(t)$ performs better than *Linear*, which is consistent with the results in Table 3 and 4. Fig. 5 shows the effect of the number of sampling steps on imputation performance. It's obvious that our methods already obtain reasonable imputation results with only 2 steps while CSDI requires at least 30 steps. Furthermore, all of our methods perform better than CSDI. The RMSE obtained by D^{3}M based on *Constant* type for $\boldsymbol{h}(t)$ is significantly better than

Table 6. Comparion of $CRPS_{sum}$ of the methods on six real-world datasets (lower is better). All the values are obtained by 10 runs.

Method	Exchange	Solar	Electricity	Traffic	Taxi	Wikipedia
GP scaling	0.010±0.000	0.364±0.009	0.024±0.001	0.075±0.001	0.176±0.026	1.413±0.962
GP Copula	0.007±0.001	0.341±0.022	0.023±0.004	0.079±0.003	0.213±0.013	0.089±0.002
Transformer-MAF	**0.005±0.003**	0.308±0.011	0.0209±0.003	0.058±0.002	0.182±0.001	0.065±0.002
TimeGrad	0.006±0.001	0.287±0.020	0.0206±0.001	0.049±0.006	0.114±0.02	0.050±0.002
TLAE	0.007±0.001	0.302±0.028	0.037±0.001	0.071±0.006	0.128±0.007	0.217±0.005
TimeLAR	0.006±0.001	0.318±0.002	0.033±0.002	0.052±0.002	0.123±0.009	0.221±0.003
ScoreGrad(VP SDE)	0.006±0.001	0.268±0.021	**0.0192±0.001**	0.043±0.004	0.102±0.006	**0.041±0.003**
ScoreGrad(sub-VP SDE)	0.006±0.001	0.256±0.015	0.0194±0.001	0.041±0.004	0.101±0.004	0.043±0.002
ScoreGrad(VE SDE)	0.007±0.001	0.277±0.011	0.0199±0.001	0.037±0.003	0.104±0.009	0.046±0.002
CSDI	0.007±0.001	0.304±0.012	0.0214±0.008	0.039±0.006	0.124±0.002	0.049±0.002
CSBI	0.009±0.002	0.287±0.021	0.0219±0.007	0.049±0.002	0.119±0.003	0.062±0.007
SSSD	0.006±0.001	0.241± 0.014	0.0196±0.001	0.037±0.002	0.103±0.004	0.043±0.003
TS-Diff	0.009±0.002	0.257±0.009	0.0223±0.006	0.053±0.003	0.127±0.006	0.067±0.008
D^3M(Constant-Sqrt)	**0.005±0.001**	0.252±0.013	0.0198±0.002	**0.030±0.002**	0.106±0.001	0.051±0.005
D^3M(Constant-Linear)	0.006±0.001	**0.208±0.018**	**0.0192±0.004**	0.034±0.001	**0.097±0.001**	0.047±0.003

the others. However, the CRPS obtained by these 4 variants are similar. More experimental results can be found in Appendix C.2.

We also do some ablation experiments to analyze the effect of EMA, time and feature-oriented linear gated attention modules on the performance and the results are shown in Table 5. When only one module is used for sequence modeling, D^3M based on EMA performs best. It indicates that the inductive bias induced by the special state space model is reasonable and effective. In addition, all the models with time-oriented gated attention modules obtain better imputation results than those without this module. It suggests the importance of global dependencies modeling for sequence modeling. Furthermore, the feature-oriented gated attention module also improves the imputation performance, which indicates the importance of modeling data correlation across different channels. The best result is obtained when all three modules are integrated together. It implies that these modules are different and useful for imputation tasks.

4.2. Probabilistic time series forecasting

4.2.1. Datasets and experimental settings

We also evaluate D^3M on autoregressive probabilistic time series forecasting tasks. We use six real-world datasets for evaluation: Exchange, Solar, Electricity, Traffic, Taxi and Wikipedia. All of these datasets can be obtained from GluonTS (Alexandrov et al., 2020). All the covariates are known for all the periods of prediction. Detailed properties of these datasets can be found in Appendix B.2.

We denote the future data as $\boldsymbol{X}_0$. The condition term for Net_θ are features extracted from historical data. Similar to previous studies (Rasul et al., 2021), a two-layer GRU is used for feature extraction because we mainly focus on the comparison of DDMs. In addition, due to the high dimensionality of features, we remove attention blocks for memory usage saving. The batch size and epochs are set as 64 and 300 separately.

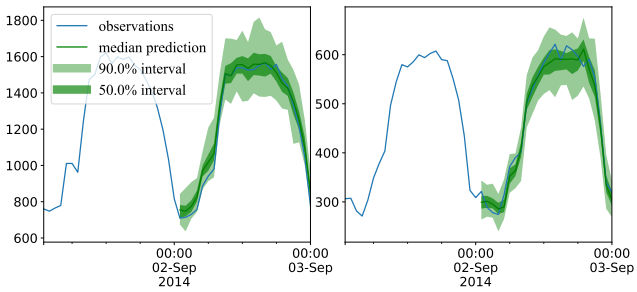


Figure 6. Examples of probabilistic forecasting for Electricity based on D^3M (Constant-Linear).

We select following competitive methods for comparison: GP Scaling, GP Copula (Salinas et al., 2019), Transformer-MAF (Rasul et al., 2020), TLAE (Nguyen & Quanz, 2021), TimeLAR (Zhang & Dai, 2022), TimeGrad (Rasul et al., 2021), ScoreGrad (Yan et al., 2021), CSDI, CSBI, SSSD, and TS-Diff. Two variants of D^3M are used for comparison. Similar to previous studies, we use $CRPS_{sum}$ to evaluate the compatibility between estimated distribution and observed future values.

4.2.2. Results

The comparison between D^3M and baselines are shown in Table 6. TimeGrad adopts denoising diffusion probabilistic models and obtains good results. ScoreGrad adopts SDE-based DDMs and achieves impressive results on Electricity, Taxi and Wikipedia. SSSD combines S4 and DDPM and obtain impressive results on Solar and Wikipedia. D^3M achieve 13.6% and 18.9% improvement on Solar and Traffic. It obtains comparable results with ScoreGrad on Electricity, Taxi and Exchange with only 10 steps. Furthermore, compared with ScoreGrad, D^3M reduces the inference time by

approximately 90%. It indicates D^3M can combine high generation speed and prediction accuracy. In addition, we also compare the $NRMSE_{sum}$ of the methods based on diffusion models in Appendix C.3.

As shown in Fig. 6, we give some examples of probabilistic forecasting on Electricity based on D^3M (Constant-Linear). The blue and green lines represent the observations and median predictions. We also give 50% and 90% prediction intervals. It shows that most observed values are within 50% prediction intervals, which indicates the effectiveness of our method. More experimental results can be found in Appendix C.3.

5. Conclusion

In this paper, we propose D^3M, which directly builds generative models based on explicit solutions of SDEs. It can be decomposed and learned by two separate processes: signal dissipation process and noise injection process. It's a general framework that unifies two existing large classes of generative models: denoising diffusion models and continuous flow-based models. Then we design some new generative models that combine high generation speed and high sampling quality. In addition, we adopt a module that combines EMA with linear gated attention modules for sequence modeling. It preserves well inductive bias, low computational complexity and local and global dependencies modeling capabilities. Our method reduces RMSE and CRPS by up to 4.6% and 4.3% compared with state-of-the-arts on imputation tasks and achieves comparable results with state-of-the-arts on forecasting tasks with only 10 steps.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 61836001) and the National Natural Science Foundation of China (Grant No. 62102022).

Impact Statement

We have established a general framework called D^3M that unifies existing SDE-based denoising diffusion models and ODE-based flow models. In addition, we combine a special state space model and linear gated attention for sequence modeling. Experimental results on probabilistic time series imputation and forecasting tasks from various domains indicate that the proposed method combines high accuracy and inference speed. As a tool, its impact depends on users. We hope and expect to see a positive impact on various sequence modeling tasks like intelligent system monitoring, web traffic forecasting and anomaly detection.

References

Albergo, Michael Samuel, V.-E. and Eric. Building normalizing flows with stochastic interpolants. In *The Eleventh International Conference on Learning Representations*, 2023.

Alcaraz, J. L. and Strodthoff, N. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2022.

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., et al. Gluonts: Probabilistic and neural time series modeling in python. *The Journal of Machine Learning Research*, 21 (1):4629–4634, 2020.

Bao, F., Li, C., Zhu, J., and Zhang, B. Analytic-dpm: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2022.

Bishop, C. M. and Nasrabadi, N. M. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Cao, W., Wang, D., Li, J., Zhou, H., Li, L., and Li, Y. Brits: Bidirectional recurrent imputation for time series. *Advances in neural information processing systems*, 31, 2018.

Che, Z., Purushotham, S., Cho, K., Sontag, D., and Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085, 2018.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Chen, Y., Deng, W., Fang, S., Li, F., Yang, T. N., Zhang, Y., Rasul, K., Zhe, S., Schneider, A., and Nevmyvaka, Y. Provably convergent schrödinger bridge with applications to probabilistic time series imputation. *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Choi, T.-M., Kang, J.-S., and Kim, J.-H. Rdis: Random drop imputation with self-training for incomplete time series data. *IEEE Access*, 2023.

De Brouwer, E., Simm, J., Arany, A., and Moreau, Y. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

Du, W., Côté, D., and Liu, Y. Saits: Self-attention-based imputation for time series. *Expert Systems with Applications*, 219:119619, 2023.

Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. Gp-vae: Deep probabilistic time series imputation. In *International conference on artificial intelligence and statistics*, pp. 1651–1661. PMLR, 2020.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential neural models with stochastic layers. *Advances in neural information processing systems*, 29, 2016.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Gu, A., Goel, K., and Re, C. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35: 35971–35983, 2022.

Heitz, E., Belcour, L., and Chambon, T. Iterative α-(de) blending: a minimalist deterministic diffusion model. *arXiv preprint arXiv:2305.03486*, 2023.

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

Hua, W., Dai, Z., Liu, H., and Le, Q. Transformer quality in linear time. In *International Conference on Machine Learning*, pp. 9099–9117. PMLR, 2022.

Huang, Y., Qin, Z., Liu, X., and Xu, K. Decoupled diffusion models with explicit transition probability. *arXiv preprint arXiv:2306.13720*, 2023.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35: 26565–26577, 2022.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33: 6696–6707, 2020.

Kingma, D., Salimans, T., Poole, B., and Ho, J. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Kobyzev, I., Prince, S. J., and Brubaker, M. A. Normalizing flows: An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):3964–3979, 2020.

Kollovieh, M., Ansari, A. F., Bohlke-Schneider, M., Zschiegner, J., Wang, H., and Wang, Y. B. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

Krishnan, R. G., Shalit, U., and Sontag, D. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Lee, S.-g., Kim, H., Shin, C., Tan, X., Liu, C., Meng, Q., Qin, T., Chen, W., Yoon, S., and Liu, T.-Y. Priorgrad: Improving conditional denoising diffusion models with data-dependent adaptive prior. In *International Conference on Learning Representations*, 2021.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Lin, L., Li, Z., Li, R., Li, X., and Gao, J. Diffusion models for time-series applications: a survey. *Frontiers of Information Technology & Electronic Engineering*, pp. 1–23, 2023.

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

Liu, S., Li, X., Cong, G., Chen, Y., and Jiang, Y. Multivariate time-series imputation with disentangled temporal representations. In *The Eleventh International Conference on Learning Representations*, 2022.

Liu, X., Gong, C., et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, 2022.

Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., May, J., and Zettlemoyer, L. Mega: Moving average equipped gated attention. In *The Eleventh International Conference on Learning Representations*, 2022.

Miao, X., Wu, Y., Wang, J., Gao, Y., Mao, X., and Yin, J. Generative semi-supervised learning for multivariate

time series imputation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8983–8991, 2021.

Mulyadi, A. W., Jun, E., and Suk, H.-I. Uncertainty-aware variational-recurrent imputation network for clinical time series. *IEEE Transactions on Cybernetics*, 52(9):9684–9694, 2021.

Nguyen, N. and Quanz, B. Temporal latent auto-encoder: A method for probabilistic multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9117–9125, 2021.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U. M., and Vollgraf, R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *International Conference on Learning Representations*, 2020.

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

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank gaussian copula processes. *Advances in neural information processing systems*, 32, 2019.

Salvi, C., Lemercier, M., and Gerasimovics, A. Neural stochastic pdes: Resolution-invariant learning of continuous spatiotemporal dynamics. *Advances in Neural Information Processing Systems*, 35:1333–1344, 2022.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Shaul, N., Chen, R. T., Nickel, M., Le, M., and Lipman, Y. On kinetic optimal probability paths for generative models. In *International Conference on Machine Learning*, pp. 30883–30907. PMLR, 2023.

Shumway, R. H., Stoffer, D. S., Shumway, R. H., and Stoffer, D. S. Arima models. *Time series analysis and its applications: with R examples*, pp. 75–163, 2017.

Silva, I., Moody, G., Scott, D. J., Celi, L. A., and Mark, R. G. Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. In *2012 Computing in Cardiology*, pp. 245–248. IEEE, 2012.

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

Song, Y. and Kingma, D. P. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020.

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

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pp. 125–125, 2016.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Yan, J., Mu, L., Wang, L., Ranjan, R., and Zomaya, A. Y. Temporal convolutional networks for the advance prediction of enso. *Scientific reports*, 10(1):8055, 2020.

Yan, T., Zhang, H., Zhou, T., Zhan, Y., and Xia, Y. Scoregrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models. *arXiv preprint arXiv:2106.10121*, 2021.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, 56(4):1–39, 2023.

Yi, X., Zheng, Y., Zhang, J., and Li, T. St-mvl: filling missing values in geo-sensory time series data. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016.

Zhang, J. and Dai, Q. Latent adversarial regularized autoencoder for high-dimensional probabilistic time series prediction. *Neural Networks*, 155:383–397, 2022.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 11106–11115, 2021.

A. Proofs and derivations

A.1. Derivation of the reverse process

Firstly, we verify the forward transition probability. The covariance matrix can be factorized as $\boldsymbol{\Sigma}_t = \boldsymbol{L}\boldsymbol{L}^T\beta_t^2$, then

$$\begin{aligned}\boldsymbol{X}_t &= \boldsymbol{X}_0 + \boldsymbol{H}_t + \boldsymbol{L}\beta_t\boldsymbol{\varepsilon} \\ &:= \boldsymbol{X}_0 + \boldsymbol{H}_{t-\Delta t} + \boldsymbol{H}_t - \boldsymbol{H}_{t-\Delta t} + \boldsymbol{L}\beta_{t-\Delta t}\boldsymbol{\varepsilon}_1 + \boldsymbol{L}\sqrt{\beta_t^2 - \beta_{t-\Delta t}^2}\boldsymbol{\varepsilon}_2 \\ &= \boldsymbol{X}_{t-\Delta t} + \boldsymbol{H}_t - \boldsymbol{H}_{t-\Delta t} + \boldsymbol{L}\sqrt{\beta_t^2 - \beta_{t-\Delta t}^2}\boldsymbol{\varepsilon}_2\end{aligned} \tag{16}$$

where $\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. Therefore, $q(\boldsymbol{X}_t|\boldsymbol{X}_{t-\Delta t}) \sim \mathcal{N}(\boldsymbol{X}_{t-\Delta t} + \boldsymbol{H}_t - \boldsymbol{H}_{t-\Delta t}, \boldsymbol{\Sigma}_t - \boldsymbol{\Sigma}_{t-\Delta t})$. We can calculate the reverse process according to the Bayesian formulas, that is

$$q(\boldsymbol{X}_{t-\Delta t}|\boldsymbol{X}_t, \boldsymbol{X}_0) = \frac{q(\boldsymbol{X}_{t-\Delta t}|\boldsymbol{X}_0)q(\boldsymbol{X}_t|\boldsymbol{X}_{t-\Delta t})}{q(\boldsymbol{X}_t|\boldsymbol{X}_0)} \sim \mathcal{N}(\boldsymbol{M}, \boldsymbol{P}), \tag{17}$$

Then, according to formulas (2.113) to (2.117) in *pattern recognition and machine learning* (Bishop & Nasrabadi, 2006), substitute Eq. (16) into Eq. (17), it's easy to obtain the covariance matrix $\boldsymbol{P}$ of $q(\boldsymbol{X}_{t-\Delta t}|\boldsymbol{X}_t, \boldsymbol{X}_0)$

$$\begin{aligned}\boldsymbol{P} &= (\boldsymbol{\Sigma}^{-1}/\beta_{t-\Delta t}^2 + \boldsymbol{\Sigma}^{-1}/(\beta_t^2 - \beta_{t-\Delta t}^2))^{-1} \\ &= \frac{\beta_{t-\Delta t}^2(\beta_t^2 - \beta_{t-\Delta t}^2)}{\beta_t^2}\boldsymbol{\Sigma},\end{aligned} \tag{18}$$

the expectation $\boldsymbol{M}$ of $q(\boldsymbol{X}_{t-\Delta t}|\boldsymbol{X}_t, \boldsymbol{X}_0)$ is

$$\begin{aligned}\boldsymbol{M} =& \frac{\beta_{t-\Delta t}^2(\beta_t^2 - \beta_{t-\Delta t}^2)\boldsymbol{\Sigma}}{\beta_t^2}\frac{\boldsymbol{\Sigma}^{-1}}{\beta_t^2 - \beta_{t-\Delta t}^2}(\boldsymbol{X}_t - \boldsymbol{H}_t + \boldsymbol{H}_{t-\Delta t}) + \frac{\beta_{t-\Delta t}^2(\beta_t^2 - \beta_{t-\Delta t}^2)\boldsymbol{\Sigma}}{\beta_t^2}\frac{\boldsymbol{\Sigma}^{-1}}{\beta_{t-\Delta t}^2}(\boldsymbol{X}_0 + \boldsymbol{H}_{t-\Delta t}) \\ =& \frac{\beta_{t-\Delta t}^2}{\beta_t^2}(\boldsymbol{X}_t - \boldsymbol{H}_t + \boldsymbol{H}_{t-\Delta t}) + \frac{(\beta_t^2 - \beta_{t-\Delta t}^2)}{\beta_t^2}(\boldsymbol{X}_0 + \boldsymbol{H}_{t-\Delta t}),\end{aligned} \tag{19}$$

Substitute $\boldsymbol{X}_0 = \boldsymbol{X}_t - \boldsymbol{H}_t - \boldsymbol{L}\beta_t\boldsymbol{\varepsilon}$ into Eq. (19), we can get

$$\boldsymbol{M} = \boldsymbol{X}_0 - \boldsymbol{H}_t + \boldsymbol{H}_{t-\Delta t} + \frac{(\beta_t^2 - \beta_{t-\Delta t}^2)\boldsymbol{L}}{\beta_t}\boldsymbol{\varepsilon}, \tag{20}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

A.2. Proof of theorem 3.2

If the learned $\boldsymbol{\psi}_\theta = \boldsymbol{\psi}$, it means that we can obtain $\boldsymbol{H}_t$ accurately. Then we firstly rewrite the noise matching loss for learning the noise injection process as

$$\mathcal{L}_{NM} = \left\|\boldsymbol{\varepsilon}_\theta(\boldsymbol{X}_t) - \boldsymbol{\varepsilon}\right\|^2, \tag{21}$$

where $\boldsymbol{X}_t = \boldsymbol{X}_0 + \boldsymbol{H}_t + \beta_t\boldsymbol{L}\boldsymbol{\varepsilon}$, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, $\boldsymbol{\varepsilon}_\theta$ is parameterized by a neural network. Given the probability path in Eq. (4), it's easy to calculate the score function as

$$\nabla_{\boldsymbol{X}_t}\log q(\boldsymbol{X}_t|\boldsymbol{X}_0) = \frac{\nabla_{\boldsymbol{X}_t}q(\boldsymbol{X}_t|\boldsymbol{X}_0)}{q(\boldsymbol{X}_t|\boldsymbol{X}_0)} = -\frac{\boldsymbol{\Sigma}^{-1}}{\beta_t^2}(\boldsymbol{X}_t - \boldsymbol{X}_0 - \boldsymbol{H}_t). \tag{22}$$

In addition, we can get $\boldsymbol{\varepsilon} = \boldsymbol{L}^{-1}(\boldsymbol{X}_t - \boldsymbol{X}_0 - \boldsymbol{H}_t)/\beta_t$ according to Eq. (4). Then Eq. (22) can be simplified as $\nabla_{\boldsymbol{X}_t}\log q(\boldsymbol{X}_t|\boldsymbol{X}_0) = -\frac{\boldsymbol{L}^{-1}\boldsymbol{\varepsilon}}{\beta_t}$. Similarly, we can parameterize the learned score function $\boldsymbol{s}_\theta$ in a similar form, that is $s_\theta = -\frac{\boldsymbol{L}^{-1}\boldsymbol{\varepsilon}_\theta}{\beta_t}$. Then according to Eq. (22), the score matching loss proposed in (Hyvärinen & Dayan, 2005) can be written as

$$\begin{aligned}\mathcal{L}_{SM} &= \left\|s_\theta - \nabla_{\boldsymbol{X}_t}\log q(\boldsymbol{X}_t|\boldsymbol{X}_0)\right\|^2 \\ &= \left\|s_\theta(\boldsymbol{X}_t) + \boldsymbol{\Sigma}^{-1}/\beta_t^2(\boldsymbol{X}_t - \boldsymbol{X}_0 - \boldsymbol{H}_t)\right\|^2 \\ &= \frac{\|\boldsymbol{\Sigma}^{-1}\|}{\beta_t^2}\left\|\boldsymbol{\varepsilon}_\theta - \boldsymbol{\varepsilon}\right\|^2 = \frac{\|\boldsymbol{\Sigma}^{-1}\|}{\beta_t^2}\mathcal{L}_{NM},\end{aligned} \tag{23}$$

Algorithm 3 Unconditional sampling procedure of D^{3}M when $\boldsymbol{h}_t = \boldsymbol{\mu} - \boldsymbol{X}_0$.

1: Initialization: $\boldsymbol{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \boldsymbol{\Sigma} = \boldsymbol{L}^T\boldsymbol{L}, t = 1, \Delta t$
2: **while** $t > 0$ **do**
3: $\boldsymbol{\psi}_\theta, \boldsymbol{\varepsilon}_\theta = \text{Net}_\theta(\boldsymbol{X}_t, t)$.
4: $\tilde{\boldsymbol{X}}_0 = \boldsymbol{X}_t - \boldsymbol{H}_t - \beta_t\boldsymbol{L}\boldsymbol{\varepsilon}_\theta$.
5: $\boldsymbol{\psi}_\theta^{new} = [w_t\boldsymbol{\psi}_\theta[0] + (1 - w_t)(\boldsymbol{\mu} - \tilde{\boldsymbol{X}}_0),]$.
6: Calculate $\boldsymbol{H}_t^\theta = \int_0^t h(s)ds$ based on $\boldsymbol{\psi}_\theta^{new}$.
7: Calculate the mean $\boldsymbol{M}$ and variance $\boldsymbol{P}$ according to Eq. (7) and Eq. (6).
8: Denote the standard deviation as $\boldsymbol{p} = \boldsymbol{P}^{1/2}$.
9: $\boldsymbol{X}_t = \boldsymbol{M} + \boldsymbol{\varepsilon}\boldsymbol{p}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.
10: $t = t - \Delta t$.
11: **end while**
Output: $\boldsymbol{X}_0$

For the denoising score matching loss (Vincent, 2011), denote $\boldsymbol{Y}_t = \frac{\alpha_0\boldsymbol{X}_t}{\alpha_t}$. Then according to Eq. (10), we have

$$\boldsymbol{Y}_t = \boldsymbol{X}_0 + \frac{\beta_t\alpha_0}{\alpha_t}\boldsymbol{L}\boldsymbol{\varepsilon} \tag{24}$$

Then the denosing score matching loss can be simplified as

$$\begin{aligned}\mathcal{L}_{DSM} &= \left\| \boldsymbol{\phi}_\theta(\boldsymbol{Y}_t) - \frac{\partial \log q(\boldsymbol{Y}_t|\boldsymbol{X}_0)}{\partial \boldsymbol{Y}_t} \right\|^2 \\ &= \left\| \boldsymbol{\phi}_\theta(\boldsymbol{Y}_t) - \frac{\boldsymbol{\Sigma}^{-1}(\boldsymbol{X}_0 - \boldsymbol{Y}_t)}{\beta_t^2\alpha_0^2/\alpha_t^2} \right\|^2\end{aligned} \tag{25}$$

If we parameterize $\boldsymbol{\phi}_\theta$ in a similar manner with $\frac{\partial \log q(\boldsymbol{Y}_t|\boldsymbol{X}_0)}{\partial \boldsymbol{Y}_t}$, then $\mathcal{L}_{DSM}$ can be simplified as

$$\mathcal{L}_{DSM} = \left\| \frac{\boldsymbol{L}^{-1}\boldsymbol{\varepsilon}_\theta(\boldsymbol{X}_t)}{\beta_t\alpha_0/\alpha_t} - \frac{\boldsymbol{L}^{-1}\boldsymbol{\varepsilon}}{\beta_t\alpha_0/\alpha_t} \right\|^2 = \frac{\|\boldsymbol{\Sigma}^{-1}\|\alpha_t^2}{\beta_t^2\alpha_0^2}\mathcal{L}_{NM}, \tag{26}$$

Therefore, under the above conditons, score matching loss (Hyvärinen & Dayan, 2005), denoising score matching loss (Sohl-Dickstein et al., 2015) and noise matching loss (Ho et al., 2020) are equivalent to each other with proper weighting functions.

A.3. Derivation of modified samplers

When $\boldsymbol{h}(t) = \boldsymbol{\mu} - \boldsymbol{X}_0, \boldsymbol{\psi} = [\boldsymbol{\mu} - \boldsymbol{X}_0]$. We can firstly use $\boldsymbol{X}_t$ to estimate $\tilde{\boldsymbol{X}}_0$ firstly. Then we can use the weighted sum of $\tilde{\boldsymbol{X}}_0$ and $\boldsymbol{\psi}_\theta[0]$ (the first element of $\boldsymbol{\psi}_\theta$) to generate $\boldsymbol{h}(t)$. The modified sampler is shown in Algorithm 3. The steps marked in blue are the modifications compared with Algorithm 2. The w_t in line 5 is a monotonically decreasing function from 1 to 0.

When the type of $\boldsymbol{h}(t) = \boldsymbol{a}t + \boldsymbol{b}$ is "Linear", there are two parameters to estimate. However, simultaneously learning these two parameters can not guarantee that the constraint 2 in Table 1 can always be satisfied. Instead, we only learn the parameter $\boldsymbol{a}$ in the training process. Then, we use Constraint 2 in Table 1 to calculate $\boldsymbol{b}$.

For the forward process, we have

$$\begin{aligned}\boldsymbol{X}_t &= \boldsymbol{X}_0 + \frac{\boldsymbol{a}}{2}t^2 + \boldsymbol{b}t + \beta_t\boldsymbol{L}\boldsymbol{\varepsilon} \\ &= \boldsymbol{X}_0 + \frac{\boldsymbol{a}}{2}t^2 + (\boldsymbol{\mu} - \boldsymbol{X}_0 - \frac{\boldsymbol{a}}{2})t + \beta_t\boldsymbol{L}\boldsymbol{\varepsilon} \\ &= (1-t)\boldsymbol{X}_0 + \frac{\boldsymbol{a}}{2}t^2 + (\boldsymbol{\mu} - \frac{\boldsymbol{a}}{2})t + \beta_t\boldsymbol{L}\boldsymbol{\varepsilon},\end{aligned} \tag{27}$$

Algorithm 4 Unconditional sampling procedure of D^{3}M when $\boldsymbol{h}_t = \boldsymbol{a}t + \boldsymbol{b}$.

Initialization: $\boldsymbol{X}_1 \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \boldsymbol{\Sigma} = \boldsymbol{L}^T\boldsymbol{L}, t = 1, \Delta t$
while $t > 0$ **do**
  $\boldsymbol{a}_\theta, \boldsymbol{\varepsilon}_\theta = \text{Net}_\theta(\boldsymbol{X}_t, t)$.
  Get $\tilde{\boldsymbol{X}}_0$ according to Eq. (28).
  Clamp $\tilde{\boldsymbol{X}}_0$ to proper range.
  $\boldsymbol{b}_\theta = \boldsymbol{\mu} - \tilde{\boldsymbol{X}}_0 - \frac{\boldsymbol{a}_\theta}{2}$.
  $\boldsymbol{\psi}_\theta = [\boldsymbol{a}_\theta, \boldsymbol{b}_\theta]$.
  Calculate $\boldsymbol{H}_t^\theta = \int_0^t h(s)ds$ based on $\boldsymbol{\psi}_\theta$.
  Calculate the mean $\boldsymbol{M}$ and variance $\boldsymbol{P}$ according to Eq. (7) and Eq. (6).
  Denote the standard deviation as $\boldsymbol{p} = \boldsymbol{P}^{1/2}$.
  $\boldsymbol{X}_t = \boldsymbol{M} + \boldsymbol{\varepsilon}\boldsymbol{p}, \boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.
  $t = t - \Delta t$.
end while
Output: $\boldsymbol{X}_0$

Table 7. Properties of the six datasets for probabilistic time series forecasting.

NAME	DIMENSION	RANGE	FREQUENCY	TIME STEPS	PREDICTION STEPS
EXCHANGE	8	$\mathbb{R}^+$	DAY	6071	30
SOLAR	137	$\mathbb{R}^+$	HOUR	7009	24
ELECTRICITY	370	$\mathbb{R}^+$	HOUR	5833	24
TRAFFIC	963	$(0, 1)$	HOUR	4001	24
TAXI	1214	$\mathbb{N}$	30-MIN	1488	24
WIKIPEDIA	2000	$\mathbb{N}$	DAY	792	30

We use the constraint $\frac{\boldsymbol{a}}{2} + \boldsymbol{b} = \boldsymbol{\mu} - \boldsymbol{X}_0$ in the second line of Eq. (27). Then we can obtain the estimated initial state as

$$\tilde{\boldsymbol{X}}_0 = \frac{\boldsymbol{X}_t - \frac{\boldsymbol{a}}{2}t^2 - (\boldsymbol{\mu} - \frac{\boldsymbol{a}}{2})t - \beta_t\boldsymbol{L}\boldsymbol{\varepsilon}_\theta}{1 - t + eps}, \tag{28}$$

where eps is a small enough positive constant. In the experiments, we set $eps = 1e - 6$ to avoid numerical problems. In addition, we also clamp the estimated $\tilde{\boldsymbol{X}}_0$ to proper range. The modified samplers are shown in Algorithm 4. The steps marked in blue are the modifications compared with Algorithm 2.

Similarly, for other $\boldsymbol{h}(t)$ with n parameters, we can choose $n - 1$ of these parameters to construct $\boldsymbol{\psi}$. Then in the sampling process, we can use the learned $n - 1$ parameters and the constraints in Table 1 to calculate the last parameter. This method can ensure the constraints are satisfied properly. We leave the derivation for the other cases in Table 2 as an exercise for readers.

B. Datasets

B.1. Probabilistic time series imputation

In this paper, we use the following two datasets that widely used for time series imputation for evaluation.

1. PhysioNet Challenge 2012 (Silva et al., 2012) is a 35-dimensional multivariate time series dataset that records irregular measurements for the first 48 hours in ICU. Similar to the processing method of (Cao et al., 2018; Che et al., 2018), we aggregate data to one point per hour. The missing rate of the dataset is about 80%. We randomly mask a certain proportion (10%, 50%, 90%) of the data as the test data set.

2. Air quality (Yi et al., 2016) is a 36-dimensional multivariate time series dataset which records hourly PM2.5 measurements for 12months. The missing rate of the dataset is about 13%. The ground truth values at the missing points are known and we use them as the test data set.

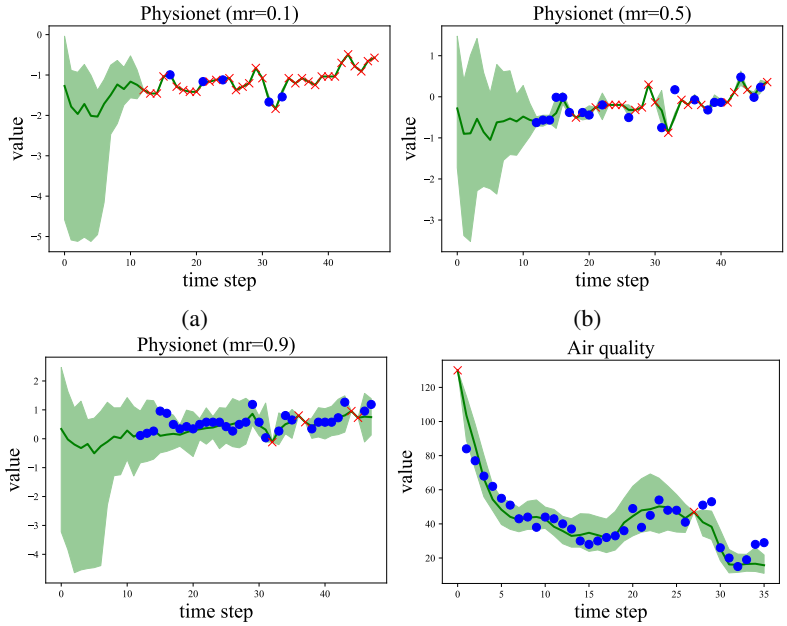


Figure 7. Examples of probabilistic imputation for Physionet with missing rate 10%, 50%, 90% and Air quality. The red points represent observed data, the blue points represent ground truth values, the green line represent the median values of imputations and green areas covers 50% of distribution intervals.

B.2. Probabilistic time series forecasting

The properties of the datasets used for probabilistic time series forecasting are shown in Table 7. It shows the dimension, range of value, frequency, total time steps and prediction steps for each dataset. These datasets have already been collected and preprocessed in GluonTS (Alexandrov et al., 2020). In the experiments, we firstly transform the value to the proper range by dividing the mean of historical data. Then we rescale the generated samples to the original value in the generative process.

C. Experiments and results

In this section, we give some extra experimental results. We analyze the influence of some important hyperparameters and give some imputation and forecasting examples on each dataset based on D^3M.

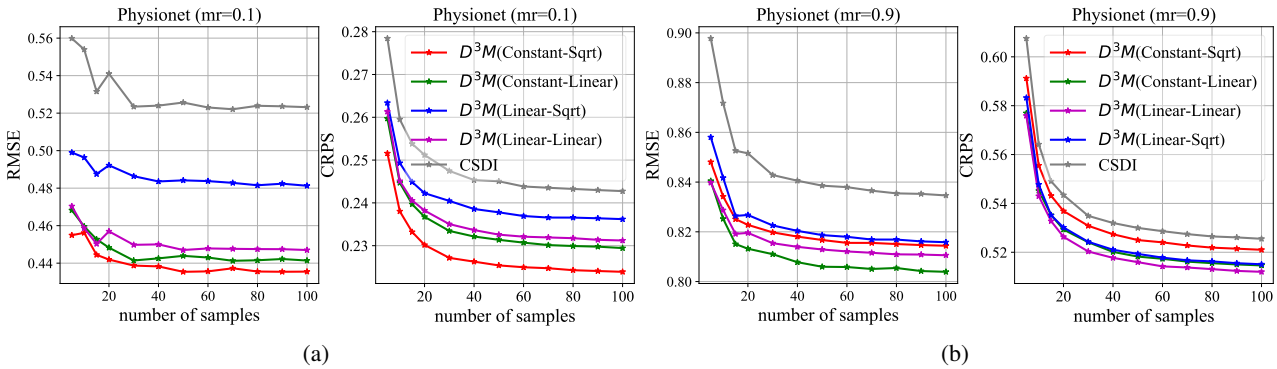
C.1. Settings of weighting functions of loss

There is a significant difference in the numerical values of each channel in time series data. Similar to previous methods, normalization is performed by dividing the original value by the mean term in probabilistic forecasting tasks. In order to balance the two terms in the loss, we set $\omega_1 = 1, \omega_2 = (\mathcal{L}_{NM}/\mathcal{L}_{SD}).detach()$ in most experiments.

C.2. Probabilistic imputations

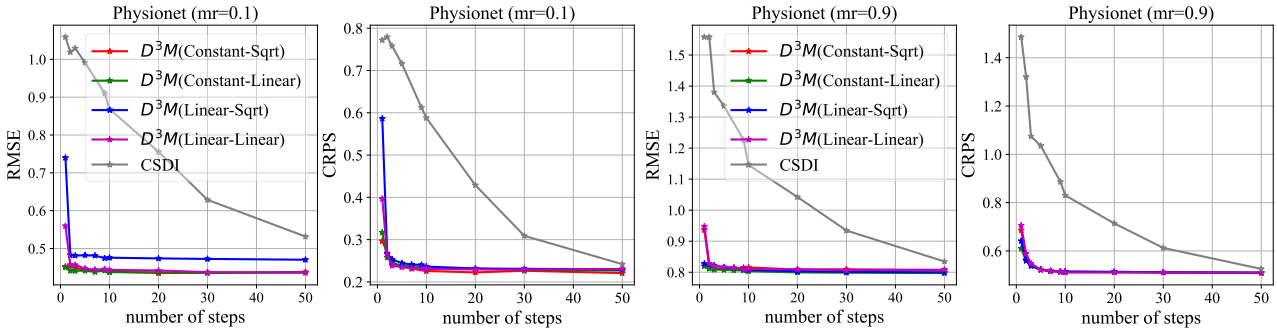
As shown in Fig. 7, we give some typical imputation examples for different experimental settings based on D^3M (Constant-Linear). The red points are observed data and the blue points are the values to be imputed. The first three figures shows the imputation results when the missing rate of the Physionet is different. It shows that our method can accurately estimates masked values when the missing rate gradually increases. In addition, as shown in the second rows of images, out method can still give a reasonable estimation of masked data when the missing rate is high. The median predictions indicate that D^3M can accurately interpolate the missing values for probabilistic time series imputation tasks and 50% prediction intervals can well cover the distribution of missing values.

In addition, we give the effect of the number of samples on performance with different experimental settings in Fig. 8. With the increasing of sampling numbers, the imputation performance get better. The effect of the number of steps for prediction with different settings are shown in Fig. 9. It's obvious that D^3M can already obtain competitive results with only 2 steps.



(a) (b)

Figure 8. The effect of number of samples for prediction(Lower is better).



(a) (b)

Figure 9. The effect of number of steps for prediction(Lower is better).

Therefore, our method can significantly reduce the inference time compared with existing sequence modeling methods based on denoising diffusion models.

C.3. Probabilistic forecasting

We also compare the $\text{NRMSE}_{\text{sum}}$ for probabilistic time series forecasting tasks and the results are shown in Table 8. Our method achieves 6.6%, 6.9% improvements on $\text{NRMSE}_{\text{sum}}$ on Solar and Traffic compared with existing methods. In addition, it obtains comparable results with ScoreGrad and SSSD on Electricity, Taxi and Exchange. Besides, TS-Diff, SSSD and ScoreGrad are probabilistic methods based on DDPM and ScoreSDE. They require hundreds of steps in the prediction process while D^3M (Constant-Sqrt/Linear) only requires 10 steps.

In addition, we give some forecasting examples of D^3M (Constant-Linear) on different datasets. The forecasting results are shown in Fig. 10, Fig. 11, Fig. 12, Fig. 13 and Fig. 14. The median prediction shows that our method can obtain accurate forecasting results on each datasets. In addition, the 90% prediction interval can accurately cover the possible range of data.

Table 8. Comparion of $\text{NRMSE}_{\text{sum}}$ of the methods on six real-world datasets (lower is better). All the values are obtained by 3 runs.

Method	Exchange	Solar	Electricity	Traffic	Taxi	Wikipedia
TimeGrad	0.011±0.001	0.611±0.018	0.038±0.002	0.069±0.002	0.209±0.004	0.076±0.001
ScoreGrad (VP SDE)	0.013±0.002	0.598±0.016	**0.033±0.004**	0.066±0.003	0.184±0.006	0.064±0.002
ScoreGrad (sub-VP SDE)	0.011±0.001	0.572±0.015	0.035±0.003	0.065±0.001	0.186±0.004	**0.062±0.003**
ScoreGrad (VE SDE)	0.013±0.001	0.597±0.019	0.040±0.003	0.060±0.002	0.189±0.008	0.069±0.002
CSDI	0.016±0.001	0.634±0.017	0.044±0.005	0.061±0.004	0.236±0.003	0.071±0.002
CSBI	0.018±0.002	0.617±0.013	0.049±0.003	0.070±0.003	0.227±0.004	0.089±0.005
SSSD	0.012±0.001	0.559±0.012	0.035±0.002	0.058±0.002	0.190±0.006	0.064±0.002
TS-Diff	0.017±0.002	0.568±0.007	0.046±0.004	0.082±0.003	0.232±0.006	0.087±0.003
D^3M(Constant-Sqrt)	**0.009±0.001**	0.576±0.010	0.039±0.002	**0.054±0.002**	0.194±0.004	0.073±0.004
D^3M(Constant-Linear)	0.012±0.002	**0.522±0.011**	0.034±0.003	0.059±0.001	**0.181±0.003**	0.068±0.003

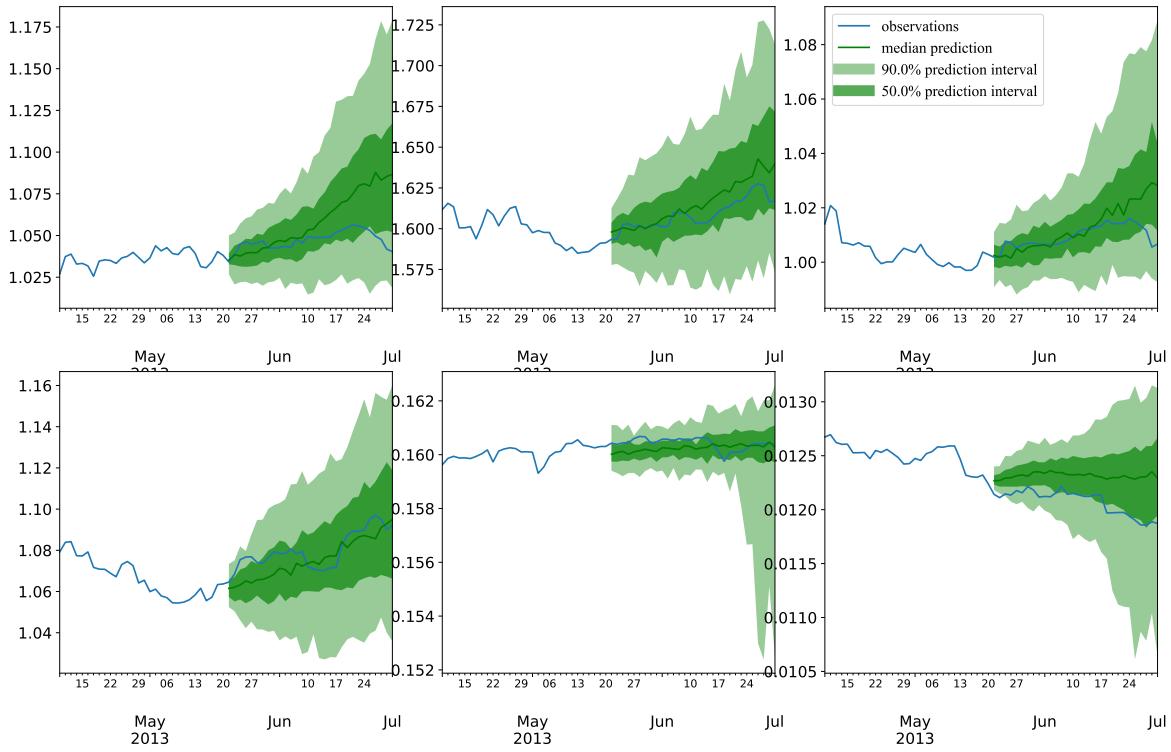


Figure 10. Forecasting results based on D^{3}M (Constant-Linear) on Exchange.

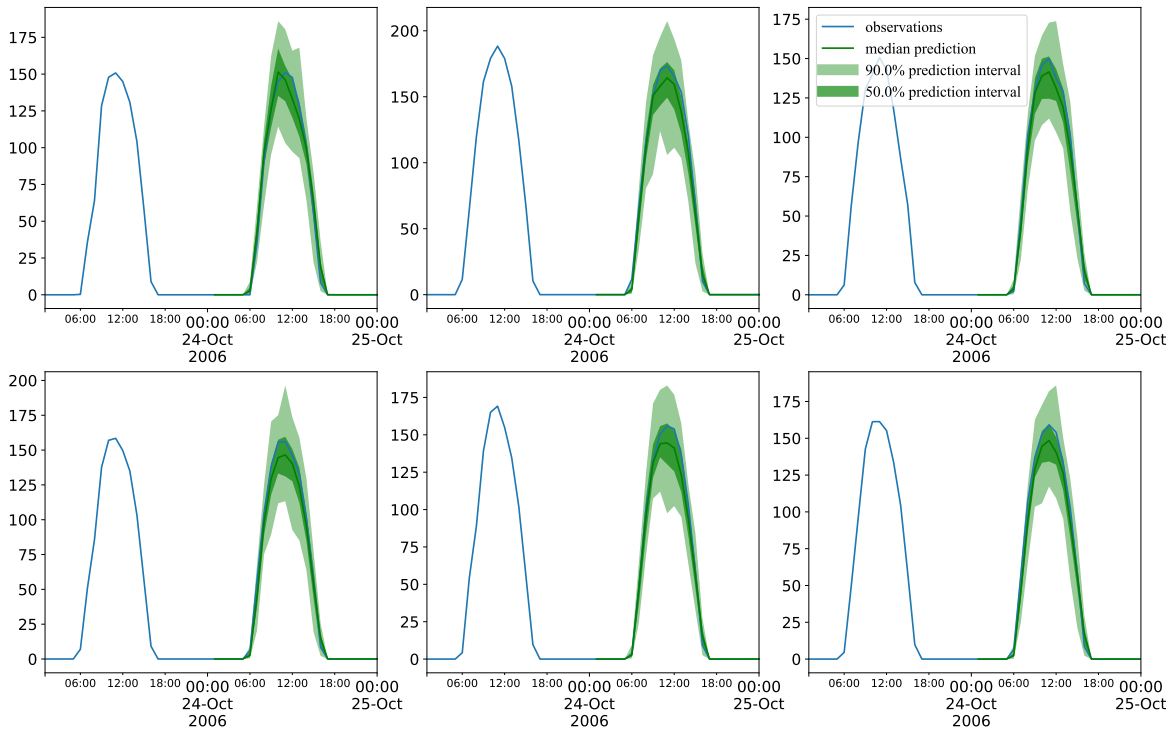


Figure 11. Forecasting results based on D^{3}M (Constant-Linear) on Solar.

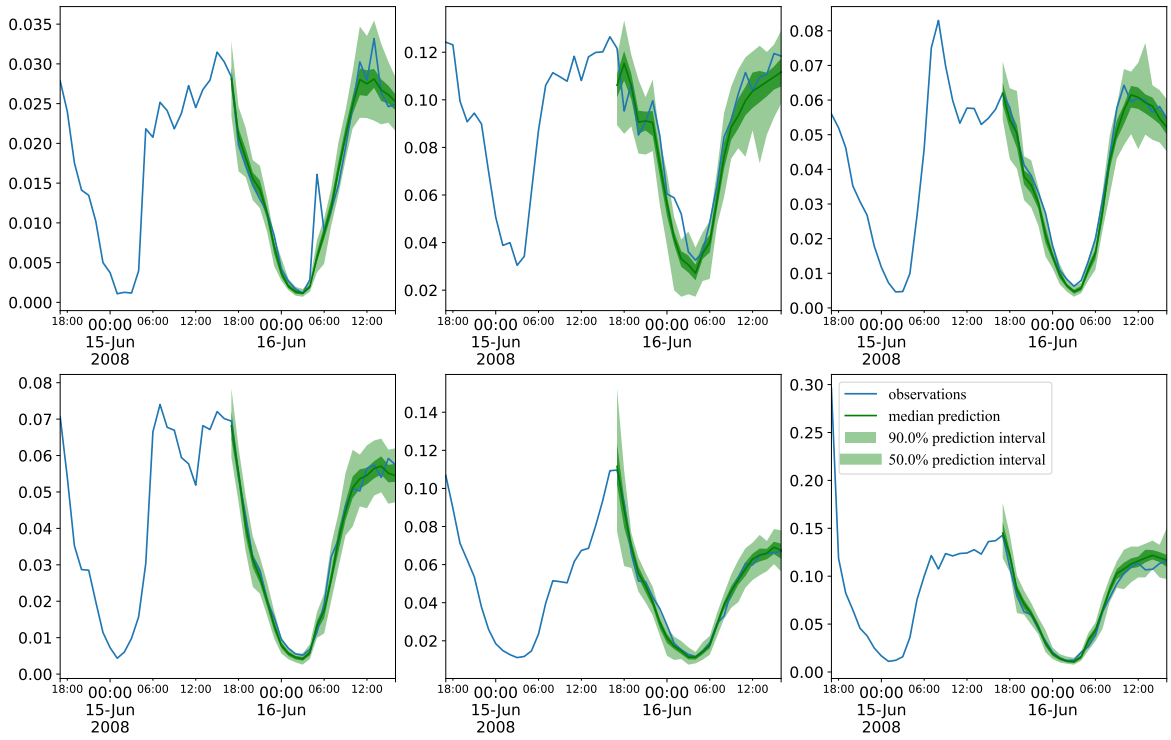


Figure 12. Forecasting results based on D^{3}M (Constant-Linear) on Traffic.

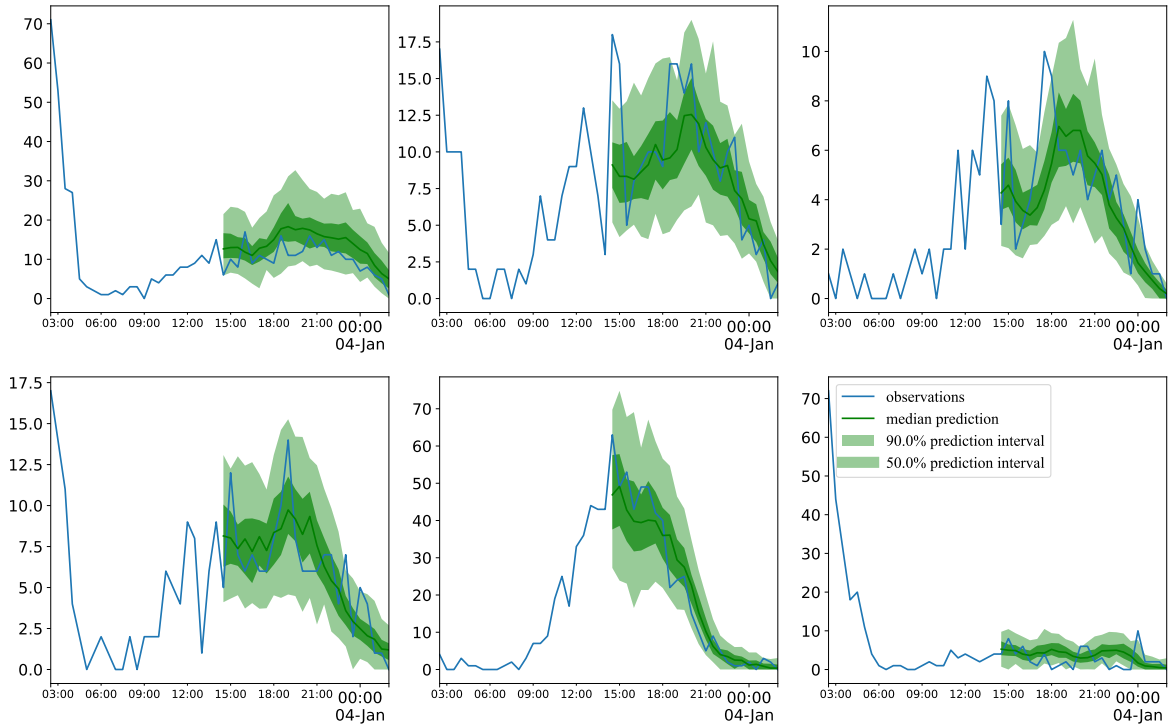


Figure 13. Forecasting results based on D^{3}M (Constant-Linear) on Taxi.

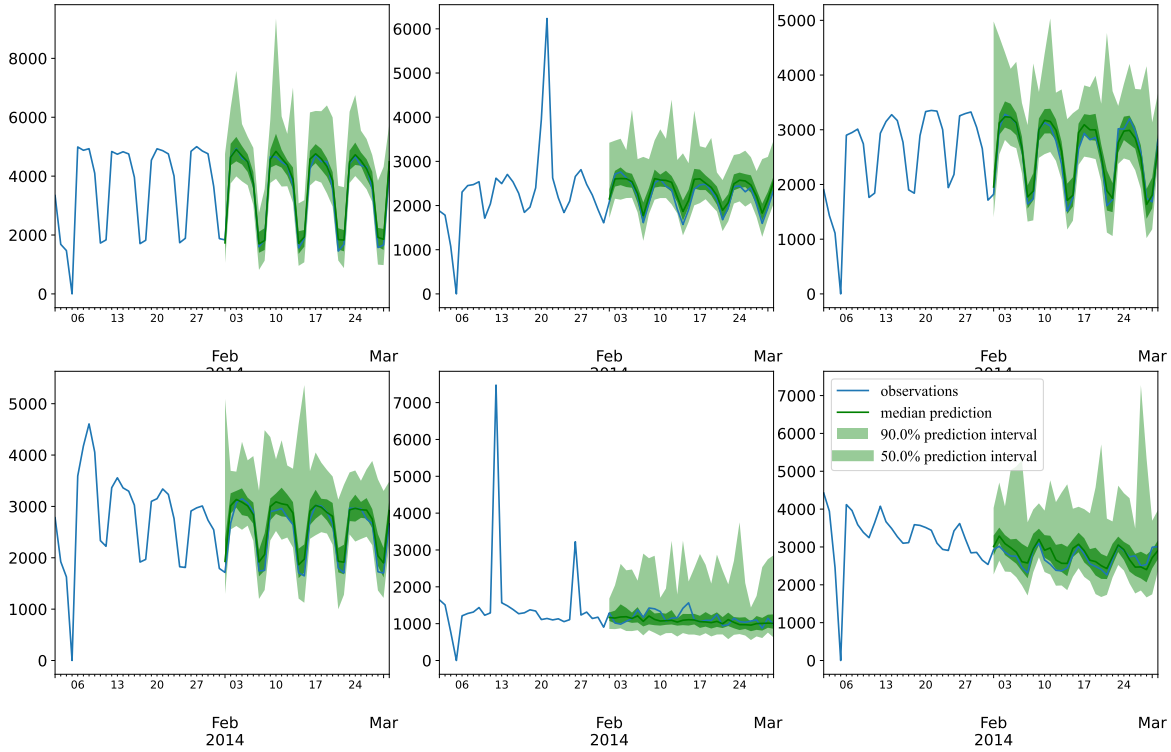


Figure 14. Forecasting results based on D^{3}M (Constant-Linear) on Wikipedia.