Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

*Review:*

# Diffusion models for time-series applications: a survey

Lequan LIN‡[1], Zhengkun LI[2], Ruikun LI[1], Xuliang LI[1], Junbin GAO[1]

[1]*Discipline of Business Analytics, The University of Sydney Business School, Camperdown, NSW 2006, Australia*

[2]*Postdoctoral Programme of Zhongtai Securities Co., Ltd., Jinan 250000, China*

E-mail: lequan.lin@sydney.edu.au; lizk@zts.com.cn; ruikun.li@sydney.edu.au;
xuli3128@uni.sydney.edu.au; junbin.gao@sydney.edu.au

Received Apr. 29, 2023; Revision accepted Sept. 18, 2023; Crosschecked Nov. 28, 2023; Published online Dec. 28, 2023

**Abstract:** Diffusion models, a family of generative models based on deep learning, have become increasingly prominent in cutting-edge machine learning research. With distinguished performance in generating samples that resemble the observed data, diffusion models are widely used in image, video, and text synthesis nowadays. In recent years, the concept of diffusion has been extended to time-series applications, and many powerful models have been developed. Considering the deficiency of a methodical summary and discourse on these models, we provide this survey as an elementary resource for new researchers in this area and to provide inspiration to motivate future research. For better understanding, we include an introduction about the basics of diffusion models. Except for this, we primarily focus on diffusion-based methods for time-series forecasting, imputation, and generation, and present them, separately, in three individual sections. We also compare different methods for the same application and highlight their connections if applicable. Finally, we conclude with the common limitation of diffusion-based methods and highlight potential future research directions.

**Key words:** Diffusion models; Time-series forecasting; Time-series imputation; Denoising diffusion probabilistic models; Score-based generative models; Stochastic differential equations

https://doi.org/10.1631/FITEE.2300310 **CLC number:** TP181

## 1 Introduction

Diffusion models, a family of deep-learning-based generative models, have risen to prominence in the machine learning community in recent years (Croitoru et al., 2023; Yang L et al., 2023). With exceptional performance in various real-world applications such as image synthesis (Austin et al., 2021; Dhariwal and Nichol, 2021; Ho et al., 2022a), video generation (Harvey et al., 2022; Ho et al., 2022b; Yang RH et al., 2022), natural language processing (Li XL et al., 2022; Nikolay et al., 2022; Yu et al., 2022), and time-series prediction (Rasul et al., 2021; Li Y et al., 2022; Alcaraz and Strodthoff, 2023), diffusion models have demonstrated their power over many existing generative techniques.

Given some observed data $\boldsymbol{x}$ from a target distribution $q(\boldsymbol{x})$, the objective of a generative model is to learn a generative process that produces new samples from $q(\boldsymbol{x})$ (Luo C, 2022). To learn such a generative process, most diffusion models begin by progressively disturbing the observed data by injecting Gaussian noise, and then applying a reversed process with a learnable transition kernel to recover the data (Sohl-Dickstein et al., 2015; Ho et al., 2020; Luo C, 2022). Typical diffusion models assume that after a certain number of noise injection steps, the observed data will become standard Gaussian noise. So, if we can find the probabilistic process that recovers the original data from standard Gaussian noise, then we can generate similar samples using the same probabilistic process with any random standard Gaussian noise as the starting point.

The recent three years have witnessed the extension of diffusion models to time-series-related

‡ Corresponding author
ORCID: Lequan LIN, https://orcid.org/0009-0006-4677-7327

applications, including time-series forecasting (Rasul et al., 2021; Li Y et al., 2022; Biloš et al., 2023), time-series imputation (Tashiro et al., 2021; Alcaraz and Strodthoff, 2023; Liu MZ et al., 2023), and time-series generation (Lim et al., 2023). Given observed historical time series, we often try to predict future time series. This process is known as time-series forecasting. Because observed time series are sometimes incomplete due to reasons such as data collection failures and human errors, time-series imputation is implemented to fill in the missing values. Different from time-series forecasting and imputation, time-series generation or synthesis aims to produce more time-series samples with characteristics similar to the observed period.

Basically, diffusion-based methods for time-series applications are developed from three fundamental formulations, including denoising diffusion probabilistic models (DDPMs), score-based generative models (SGMs), and stochastic differential equations (SDEs). The target distributions learned by the diffusion components in different methods often involve the condition on previous time steps. Nevertheless, the design of the diffusion and denoising processes varies with different task objectives. Hence, a comprehensive and self-contained summary of relevant literature will be an inspiring beacon for new researchers who are just entering this new-born area and experienced researchers who seek future directions. Accordingly, this survey aims to summarize the literature, compare different approaches, and identify potential limitations.

In this paper, we review diffusion-based models for time-series applications (refer to Table 1 for a quick summary). For better understanding, we include a brief introduction about three predominant formulations of diffusion models in Section 2. Next, we categorize the existing models based on their major functions. Specifically, we discuss the models primarily for time-series forecasting, time-series imputation, and time-series generation in Sections 3, 4, and 5, respectively. In each section, we have a separate subsection for problem formulation, which helps clarify the objective, training, and forecasting settings of each specific application. We highlight if a model can serve multiple purposes and articulate the linkage when one model is related to or slightly different from another. In Section 6, we compare all the models mentioned in this review on some specific aspects such as their diffusion formulation and sampling processes. In Section 7, we briefly introduce some practical application domains in the real world while providing good sources of publicly available datasets. Furthermore, we discuss some current limitations and challenges faced by researchers and practitioners to inspire future research in Section 8. Eventually, we conclude this survey in Section 9.

## 2 Basics of diffusion models

The underlying principle of diffusion models is to progressively perturb the observed data with a forward diffusion process and then recover the original data through a backward reverse process. The forward process involves multiple steps of noise injection, where the noise level changes at each step. The backward process, in contrast, consists of multiple denoising steps that aim to remove the injected noise gradually. Normally, the backward process is parameterized by a neural network. Once the backward process has been learned, it can generate new samples from almost arbitrary initial data. Stemming from this basic idea, diffusion models are predominantly formulated in three ways: DDPMs, SGMs, and SDEs.

In this section, we discuss the theoretical background of diffusion models before introducing the above-mentioned formulations. By doing this, we aim to lead readers who are not deeply familiar with this area from traditional generative concepts to diffusion models.

### 2.1 Background of diffusion models

Generative models are designed to generate samples from the same distribution of the observed data $\boldsymbol{x}$ by learning the approximation of the true distribution $q(\boldsymbol{x})$. To achieve this, many methods assume that the observed data can be generated from some invisible data or the so-called latent variable. The learning task then becomes minimizing the variational lower bound of the negative log-likelihood of the target distribution with some learnable parameters associated with the distribution of the latent variable. Variational autoencoder (VAE) is one of the most widely known methods built on this assumption (Kingma and Welling, 2013). While the VAE is based on a single latent variable, the hierarchical variational autoencoder (HVAE) generalizes

**Table 1 A summary of diffusion-based methods for time-series applications**

| Application | Data type | Method | Source |
|---|---|---|---|
| Time-series forecasting | Multivariate time series | TimeGrad | Rasul et al., 2021 |
| | | ScoreGrad | Yan et al., 2021 |
| | | $D^3$VAE | Li Y et al., 2022 |
| | | DSPD/CSPD | Biloš et al., 2023 |
| | | TimeDiff | Shen and Kwok, 2023 |
| | Spatio-temporal graphs | DiffSTG | Wen et al., 2023 |
| | | GCRDD | Li RK et al., 2023 |
| Time-series imputation | Multivariate time series | CSDI | Tashiro et al., 2021 |
| | | DSPD/CSPD | Biloš et al., 2023 |
| | | SSSD | Alcaraz and Strodthoff, 2023 |
| | Spatio-temporal graphs | PriSTI | Liu MZ et al., 2023 |
| Time-series generation | Multivariate time series | TSGM | Lim et al., 2023 |
| | | DiffTime | Coletta et al., 2023 |
| | | Loss-DiffTime | Coletta et al., 2023 |
| | | Guided-DiffTime | Coletta et al., 2023 |

the idea to multiple latent variables with hierarchies, enhancing the flexibility of the underlying generative assumptions (Kingma et al., 2016; Sønderby et al., 2016).

Diffusion models were originally introduced in Sohl-Dickstein et al. (2015). Then some improvements proposed by Ho et al. (2020) endowed diffusion models with remarkable practical value, contributing to their conspicuous popularity nowadays. As a matter of fact, diffusion models can be seen as an extension of the HVAE, constrained by the following restrictions: (1) the latent variables are under a Markov assumption, effectively leading to a Markov chain based generative process; (2) the observed data and latent variables share the same dimension; (3) the Markov chain is linked by Gaussian transition kernels, and the last latent variable of the chain is from the standard Gaussian distribution. For better understanding, we illustrate the connections and differences between traditional generative models and diffusion models in Fig. 1.

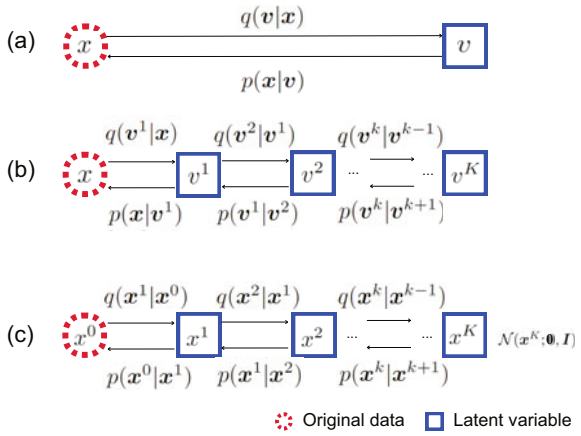


**Fig. 1 Comparison of traditional generative models and the diffusion model: (a) variational autoencoder; (b) Markov hierarchical variational autoencoder; (c) diffusion model**

## 2.2 Denoising diffusion probabilistic models

DDPMs implement the forward and backward processes through two Markov chains (Sohl-Dickstein et al., 2015; Ho et al., 2020). Let the original observed data be $\boldsymbol{x}^0$, where 0 indicates that the data are free from the noises injected in the diffusion process.

The forward Markov chain transforms $\boldsymbol{x}^0$ to a sequence of disturbed data $\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^K$ with a diffusion transition kernel:

$$q(\boldsymbol{x}^k|\boldsymbol{x}^{k-1}) = \mathcal{N}\left(\boldsymbol{x}^k; \sqrt{\alpha_k}\boldsymbol{x}^{k-1}, (1-\alpha_k)\boldsymbol{I}\right), \quad (1)$$

where $\alpha_k \in (0,1)$ for $k = 1, 2, ..., K$ are hyperparameters indicating the changing variance of the noise level at each step, and $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the general notation for the Gaussian distribution of $\boldsymbol{x}$ with the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$. A nice property of this Gaussian transition kernel is that we may obtain

$\boldsymbol{x}^k$ directly from $\boldsymbol{x}^0$ by

$$q(\boldsymbol{x}^k|\boldsymbol{x}^0)=\mathcal{N}\left(\boldsymbol{x}^k;\sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0,(1-\tilde{\alpha}_k)\boldsymbol{I}\right), \quad (2)$$

where $\tilde{\alpha}_k \coloneqq \prod_{i=1}^{k}\alpha_i$. Therefore, $\boldsymbol{x}^k=\sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. This property can be derived by applying the reparameterization trick recursively to each backward step from $\boldsymbol{x}_t$ to $\boldsymbol{x}_0$, and we refer readers to page 11 in Luo C (2022) for the detailed proof. Normally, we design $\tilde{\alpha}_K\approx 0$ such that $q(\boldsymbol{x}^K)\coloneqq\int q(\boldsymbol{x}^K|\boldsymbol{x}^0)q(\boldsymbol{x}^0)\mathrm{d}\boldsymbol{x}^0\approx\mathcal{N}(\boldsymbol{x}^K;\boldsymbol{0},\boldsymbol{I})$, which means the starting point of the backward chain can be any standard Gaussian noise. This matches the third restriction of diffusion models mentioned in Section 2.1.

Assuming that the backward process can also be realized with a Gaussian transition kernel, the reverse transition kernel is modeled by a parameterized neural network:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k)=\mathcal{N}\left(\boldsymbol{x}^{k-1};\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k),\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k)\right), \quad (3)$$

where $\boldsymbol{\theta}$ denotes learnable parameters. Now, the remaining problem is how to estimate $\boldsymbol{\theta}$. Basically, the objective is to maximize the likelihood objective function so that the probability of observing the training sample $\boldsymbol{x}^0$ estimated by $p_{\boldsymbol{\theta}}(\boldsymbol{x}^0)$ is maximized. This task is accomplished by minimizing the variational lower bound of the estimated negative log-likelihood $-\log q(\boldsymbol{x}^0)$, that is,

$$\mathbb{E}_{q(\boldsymbol{x}^{0:K})}\left[-\log p(\boldsymbol{x}^K)-\sum_{k=1}^{K}\log\frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}^{k-1}|\boldsymbol{x}^k)}{q(\boldsymbol{x}^k|\boldsymbol{x}^{k-1})}\right], \quad (4)$$

where $\boldsymbol{x}^{0:K}$ denotes the sequence $\boldsymbol{x}^0,\boldsymbol{x}^1,...,\boldsymbol{x}^K$.

Ho et al. (2020) proposed that we could simplify the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k)$ in Eq. (3) as a constant-dependent matrix $\sigma_k^2\boldsymbol{I}$, where $\sigma_k^2$ controls the noise level and may vary at different diffusion steps. In addition, they rewrote the mean as a function of a learnable noise term as

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k)=\frac{1}{\sqrt{\alpha_k}}\left(\boldsymbol{x}^k-\zeta(k)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k)\right), \quad (5)$$

where $\zeta(k)=\frac{1-\alpha_k}{\sqrt{1-\tilde{\alpha}_k}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is a noise-matching network that predicts $\boldsymbol{\epsilon}$ corresponding to inputs $\boldsymbol{x}^k$ and $k$. With the property in Eq. (2), Ho et al. (2020) further simplified the objective function to

$$\mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},k\right)\right\|^2\right], \quad (6)$$

where $\delta(k)=\frac{(1-\alpha_k)^2}{2\sigma_k^2\alpha_k(1-\tilde{\alpha}_k)}$ is a positive-valued weight that can be discarded to produce better performance in practice. This simplified objective function can be understood as approximating the noise injected at each diffusion step in the forward process, which makes the mean of the reverse transition kernel attainable.

Eventually, samples are generated by eliminating the noises in $\boldsymbol{x}^K\sim\mathcal{N}(\boldsymbol{x}^K;\boldsymbol{0},\boldsymbol{I})$. Specifically, for $k=K-1,K-2,...,0$,

$$\boldsymbol{x}^k\leftarrow\frac{\left(\boldsymbol{x}^{k+1}-\zeta(k+1)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^{k+1},k+1)\right)}{\sqrt{\alpha_{k+1}}}+\sigma_k\boldsymbol{z},$$

where $\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ for $k=K-1,K-2,...,1$, and $\boldsymbol{z}=0$ for $k=0$. It is not hard to see that the sampling process above follows the reverse transition kernel in Eq. (3) with the mean and variance represented by the first and second terms on the right-hand side respectively.

### 2.3 Score-based generative models

SGMs consist of two modules, including score matching and annealed Langevin dynamics (ALD). ALD is a sampling algorithm that generates samples with an iterative process by applying Langevin Monte Carlo at each update step (Song Y and Ermon, 2019). Stein score is an essential component of ALD. The Stein score of a density function $q(\boldsymbol{x})$ is defined as $\nabla_{\boldsymbol{x}}\log q(\boldsymbol{x})$, which can be understood as a vector field pointing in the direction in which the log-likelihood of the target distribution grows most quickly. Because the true probabilistic distribution $q(\boldsymbol{x})$ is usually unknown, score matching (Hyvärinen, 2005) is implemented to approximate the Stein score with a score-matching network. Here we primarily focus on denoising score matching (Vincent, 2011) because it is empirically more efficient, but other methods such as sliced score matching (Song Y et al., 2020) are also commonly mentioned in the literature.

The underlying principle of denoising score matching is to process the observed data with the forward transition kernel $q(\boldsymbol{x}^k|\boldsymbol{x}^0)=\mathcal{N}(\boldsymbol{x}^k;\boldsymbol{x}^0,\sigma_k^2\boldsymbol{I})$, with $\sigma_k^2$ being a set of increasing noise levels for $k=1,2,...,K$, and then to jointly estimate the Stein scores for the noise density distributions $q_{\sigma_1}(\boldsymbol{x}),q_{\sigma_2}(\boldsymbol{x}),...,q_{\sigma_k}(\boldsymbol{x})$ (Song Y and Ermon, 2019). The Stein score for noise density function $q_{\sigma_k}(\boldsymbol{x})$ is defined as $\nabla_{\boldsymbol{x}}\log q_{\sigma_k}(\boldsymbol{x})$. Once the Stein scores are

found, samples can be generated with ALD by gradually pushing random white noises towards high-density regions of the target distribution along the direction pointed by these scores.

Usually, the Stein score is approximated by a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, \sigma_k)$, where $\boldsymbol{\theta}$ contains learnable parameters. Accordingly, the initial objective function is given as

$$\mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{x}^k}\left[\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,\sigma_k)-\nabla_{\boldsymbol{x}^k}\log q_{\sigma_k}(\boldsymbol{x}^k)\right\|\right]. \quad (7)$$

With the Gaussian assumption of the forward transition kernel, a tractable version of the objective function can be found as

$$\mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{x}^k}\left[\delta(k)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,\sigma_k)+\frac{\boldsymbol{x}^k-\boldsymbol{x}^0}{\sigma_k^2}\right\|^2\right], \quad (8)$$

where $\delta(k)$ is a positive-valued weight depending on the noise scale $\sigma_k$.

After the score-matching network $\boldsymbol{s}_{\boldsymbol{\theta}}$ is learned, the ALD algorithm will be implemented for sampling. The algorithm is initialized with a sequence of increasing noise levels $\sigma_1, \sigma_2, ..., \sigma_K$ and a starting point $\boldsymbol{x}^{K,0} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. For $k = K, K-1, ..., 0$, $\boldsymbol{x}^k$ will be updated with $N$ iterations that compute

$$\boldsymbol{x}^{k,n} \leftarrow \boldsymbol{x}^{k,n-1}+\frac{1}{2}\eta_k \boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{k,n-1},\sigma_k\right)+\sqrt{\eta_k}\boldsymbol{z},$$

where $n = 1, 2, ..., N$, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\eta_k$ represents the update step. Note that after each $N$ iterations, the last output $\boldsymbol{x}^{k,N}$ will be assigned as the starting point of the next $N$ iterations, that is, $\boldsymbol{x}^{k-1,1}$. $\boldsymbol{x}^{0,N}$ will be the final sample. The role of $\boldsymbol{z}$ in this sampling process is to add slight uncertainty, such that the algorithm will not end up with almost identical samples.

### 2.4 Stochastic differential equations

DDPMs and SGMs implement the forward pass as a discrete process, which means we should carefully design the diffusion steps. To overcome this limitation, one may consider the diffusion process as continuous such that it becomes the solution of an SDE (Yang S et al., 2021). This formulation can be thought of as a generalization of the previous two formulations, because both DDPMs and SGMs are discrete forms of SDEs. The backward process is modeled as a time-reverse SDE, and samples can be generated by solving this time-reverse SDE. Let $\boldsymbol{w}$ and $\tilde{\boldsymbol{w}}$ be a standard Wiener process and its time-reverse version, respectively, and consider a continuous diffusion time $k \in [0, K]$. A general expression of SDE is

$$\mathrm{d}\boldsymbol{x} = f(\boldsymbol{x}, k)\mathrm{d}k + g(k)\mathrm{d}\boldsymbol{w}, \quad (9)$$

and the time-reverse SDE, as shown by Anderson (1982), is

$$\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x},k)-g^2(k)\nabla_{\boldsymbol{x}}\log q_k(\boldsymbol{x})\right]\mathrm{d}k + g(k)\mathrm{d}\tilde{\boldsymbol{w}}. \quad (10)$$

In addition, Yang S et al. (2021) illustrated that sampling from the probability flow ordinary differential equation (ODE) as follows has the same distribution as the time-reverse SDE:

$$\mathrm{d}\boldsymbol{x} = \left[f(\boldsymbol{x},k)-\frac{1}{2}g^2(k)\nabla_{\boldsymbol{x}}\log q_k(\boldsymbol{x})\right]\mathrm{d}k. \quad (11)$$

Here $f(\boldsymbol{x}, k)$ and $g(k)$ separately compute the drift coefficient and the diffusion coefficient for the diffusion process. $\nabla_{\boldsymbol{x}}\log q_k(\boldsymbol{x})$ is the Stein score corresponding to the marginal distribution of $\boldsymbol{x}^k$, which is unknown but can be learned with a similar method as in SGMs with the objective function

$$\mathbb{E}_{k,\boldsymbol{x}^0,\boldsymbol{x}^k}\left[\delta(k)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}^k,k)-\nabla_{\boldsymbol{x}^k}\log q_{0k}(\boldsymbol{x}^k|\boldsymbol{x}^0)\right\|^2\right]. \quad (12)$$

Therefore, the training objective is once again to find a neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}^k, k)$ to approximate the Stein scores. However, because the Stein scores here are instead based on a continuous process, the subtraction term cannot be simplified as before, and the SDE of the diffusion process is required before one can derive the Stein scores for approximation.

Now, how to write the diffusion processes of DDPMs and SGMs as SDEs? Recall that $\alpha_k$ is a defined parameter in DDPMs and that $\sigma_k^2$ denotes the noise level in SGMs. The SDE corresponding to DDPMs is known as a variance preserving (VP) SDE, and is defined as

$$\mathrm{d}\boldsymbol{x} = -\frac{1}{2}\alpha(k)\boldsymbol{x}\mathrm{d}k + \sqrt{\alpha(k)}\mathrm{d}\boldsymbol{w}, \quad (13)$$

where $\alpha(\cdot)$ is a continuous function, and $\alpha\left(\frac{k}{K}\right) = K(1-\alpha_k)$ as $K \to \infty$. For the forward pass of SGMs, the associated SDE is known as a variance exploding (VE) SDE, defined as

$$\mathrm{d}\boldsymbol{x} = \sqrt{\frac{\mathrm{d}\left[\sigma^2(k)\right]}{\mathrm{d}k}}\mathrm{d}\boldsymbol{w}, \quad (14)$$

where $\sigma(\cdot)$ is a continuous function, and $\sigma(\frac{k}{K}) = \sigma_k$ as $K \to \infty$ (Yang S et al., 2021). Inspired by the VP SDE, Yang S et al. (2021) designed another SDE called the sub-VP SDE, which performs especially well on likelihoods, and is given by

$$\mathrm{d}\boldsymbol{x} = -\frac{1}{2}\alpha(k)\boldsymbol{x}\mathrm{d}k + \sqrt{\alpha(k)\left(1-\mathrm{e}^{-2\int_0^k \alpha(s)\mathrm{d}s}\right)}\mathrm{d}\boldsymbol{w}. \tag{15}$$

The objective function involves a perturbation distribution $q_{0k}(\boldsymbol{x}^k|\boldsymbol{x}^0)$ that varies for different SDEs. For the three aforementioned SDEs, their corresponding perturbation distributions are derived as

$$q_{0k}(\boldsymbol{x}^k|\boldsymbol{x}^0) = \begin{cases} \mathcal{N}(x^k; x^0, [\sigma(k)^2 - \sigma(0)^2]\boldsymbol{I}), \\ \qquad\qquad \text{(VP SDE)} \\ \mathcal{N}(x^k; x^0\mathrm{e}^{-\frac{1}{2}\int_0^k \alpha(s)\mathrm{d}s}, \left[1- \right. \\ \left.\mathrm{e}^{-\int_0^k \alpha(s)\mathrm{d}s}\right]\boldsymbol{I}), \quad \text{(VE SDE)} \\ \mathcal{N}(x^k; x^0\mathrm{e}^{-\frac{1}{2}\int_0^k \alpha(s)\mathrm{d}s}, \left[1- \right. \\ \left.\mathrm{e}^{-\int_0^k \alpha(s)\mathrm{d}s}\right]^2\boldsymbol{I}). \quad \text{(sub-VP SDE)} \end{cases} \tag{16}$$

After successfully learning $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}, k)$, samples are produced by deriving the solutions to the time-reverse SDE or the probability flow ODE with techniques such as ALD.

## 3 Time-series forecasting

Multivariate time-series forecasting is a crucial area of study in machine learning research, with wide-ranging applications across a variety of industries. Different from the univariate time series, which tracks only one feature over time, the multivariate time series involves the historical observations of multiple features that interact with each other and evolve with time (Fig. 2). Consequently, multivariate time series provide a more comprehensive understanding of complex systems and realize more reliable predictions of future trends and behaviours.

In recent years, generative models have been implemented for multivariate time-series forecasting tasks. For example, WaveNet is a generative model with dilated causal convolutions that encode long-term dependencies for sequence prediction (van den Oord et al., 2016). As another example, Kashif et al. (2021) modeled a multivariate time series with an autoregressive deep learning model, in which the data distribution is expressed by a conditional normalizing flow. Nevertheless, the common shortcoming of these models is that the functional structure

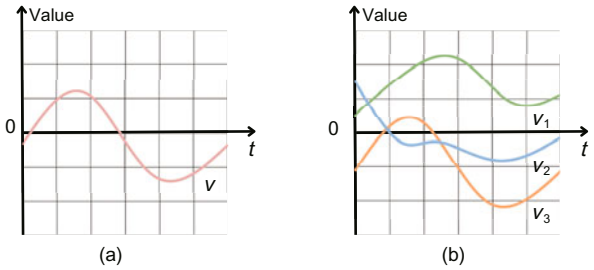


**Fig. 2 Examples of a univariate time series (a) and a multivariate time series (b)**

of their target distributions is strictly constrained. Diffusion-based methods, on the other hand, can provide a less restrictive solution. In this section, we will discuss five diffusion-based approaches. We also discuss two models designed specifically for spatio-temporal graphs (i.e., spatially related entities with multivariate time series) to highlight the extension of diffusion theories to more complicated problem settings. Because relevant literature mostly focuses on multivariate time-series forecasting, "forecasting" refers to multivariate time-series forecasting in the rest of this survey unless otherwise stated.

### 3.1 Problem formulation

Consider a multivariate time series $\boldsymbol{X}^0 = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, ..., \boldsymbol{x}_T^0 | \boldsymbol{x}_i^0 \in \mathbb{R}^D\}$, where 0 indicates that the data are free from the perturbation in the diffusion process. The forecasting task is to predict $\boldsymbol{X}_{\mathrm{p}}^0 = \{\boldsymbol{x}_{t_0}^0, \boldsymbol{x}_{t_0+1}^0, ..., \boldsymbol{x}_T^0\}$ given the historical information $\boldsymbol{X}_{\mathrm{c}}^0 = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, ..., \boldsymbol{x}_{t_0-1}^0\}$. $\boldsymbol{X}_{\mathrm{c}}^0$ is known as the context window, while $\boldsymbol{X}_{\mathrm{p}}^0$ is known as the prediction interval. In diffusion-based models, the problem is formulated as learning the joint probabilistic distribution of data in the prediction interval:

$$q\left(\boldsymbol{x}_{t_0:T}^0 | \boldsymbol{x}_{1:t_0-1}^0\right) = \prod_{t=t_0}^{T} q\left(\boldsymbol{x}_t^0 | \boldsymbol{x}_{1:t_0-1}^0\right). \tag{17}$$

Some literature also considers the role of covariates in forecasting, such as Rasul et al. (2021) and Yan et al. (2021). Covariates are additional information that may impact the behavior of variables over time, such as seasonal fluctuations and weather changes. Incorporating covariates in forecasting often helps strengthen the identification of factors that drive temporal trends and patterns in data. The forecasting problem with covariates is

formulated as

$$q\left(\boldsymbol{x}_{t_0:T}^0|\boldsymbol{x}_{1:t_0-1}^0,\boldsymbol{c}_{1:T}\right)=\prod_{t=t_0}^{T}q\left(\boldsymbol{x}_t^0|\boldsymbol{x}_{1:t_0-1}^0,\boldsymbol{c}_{1:T}\right), \tag{18}$$

where $\boldsymbol{c}_{1:T}$ denotes the covariates for all time points and is assumed to be known for the whole period.

For the purpose of training, one may randomly sample the context window followed by the prediction window from the complete training data. This process can be seen as applying a moving window with size $T$ on the whole timeline. Then, the optimization of the objective function can be conducted with the samples. Forecasting a future time series is usually achieved by the generation process corresponding to the diffusion models.

### 3.2 TimeGrad

The first noticeable work on diffusion-based forecasting is TimeGrad proposed by Rasul et al. (2021). Developed from DDPMs, TimeGrad first injects noise to data at each predictive time point, and then gradually denoises through a backward transition kernel conditioned on historical time series. To encode historical information, TimeGrad approximates the conditional distribution in Eq. (18) by

$$\prod_{t=t_0}^{T}p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^0|\boldsymbol{h}_{t-1}), \tag{19}$$

where

$$\boldsymbol{h}_t=\mathrm{RNN}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^0,\boldsymbol{c}_t,\boldsymbol{h}_{t-1}) \tag{20}$$

is the hidden state calculated with a recurrent neural network (RNN) module such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or gated recurrent unit (GRU) (Chung et al., 2014) that can preserve historical temporal information, and $\boldsymbol{\theta}$ contains learnable parameters for the overall conditional distribution and its RNN component.

The objective function of TimeGrad is in the form of a negative log-likelihood, given as

$$\sum_{t=t_0}^{T}-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^0|\boldsymbol{h}_{t-1}), \tag{21}$$

where for each $t\in[t_0,T]$, $-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^0|\boldsymbol{h}_{t-1})$ is upper bounded by

$$\mathbb{E}_{k,\boldsymbol{x}_t^0,\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{x}_t^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{h}_{t-1},k\right)\right\|^2\right]. \tag{22}$$

The context window is used to generate the hidden state $\boldsymbol{h}_{t_0-1}$ for the starting point of the training process. It is not hard to see that Eq. (22) is very similar to Eq. (6) except for the inclusion of hidden states to represent the historical information.

In the training process, the parameter $\boldsymbol{\theta}$ is estimated by minimizing the negative log-likelihood objective function with stochastic sampling. Then, future time series are generated in a step-by-step manner. Suppose that the last time point of the complete time series is $\tilde{T}$. The first step is to derive the hidden state $\boldsymbol{h}_{\tilde{T}}$ based on the last available context window. Next, the observation for the next time point $\tilde{T}+1$ is predicted in a manner similar to DDPMs:

$$\boldsymbol{x}_{\tilde{T}+1}^k\leftarrow\frac{\left(\boldsymbol{x}_{\tilde{T}+1}^{k+1}-\zeta(k+1)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_{\tilde{T}+1}^{k+1},\boldsymbol{h}_{\tilde{T}},k+1)\right)}{\sqrt{\alpha_{k+1}}}+\sigma_{k+1}\boldsymbol{z}.$$

The predicted $\boldsymbol{x}_{\tilde{T}+1}^k$ should be fed back to the RNN module to obtain $\boldsymbol{h}_{\tilde{T}+1}$ before the prediction for the next time point. The sampling process will be repeated until the desired length of the future time series is reached.

### 3.3 ScoreGrad

ScoreGrad shares the same target distribution as TimeGrad, but it is alternatively built upon SDEs, extending the diffusion process from discrete to continuous and replacing the number of diffusion steps with an interval of integration (Yan et al., 2021). ScoreGrad is composed of a feature extraction module and a conditional SDE-based score-matching module. The feature extraction module is almost identical to the computation of $\boldsymbol{h}_t$ in TimeGrad. However, Yan et al. (2021) discussed the potential of adopting other network structures to encode historical information, such as temporal convolutional networks (van den Oord et al., 2016) and attention-based networks (Vaswani et al., 2017). Here, we still focus on RNN as the default choice. In the conditional SDE-based score-matching module, the diffusion process is conducted through the same SDE as in Eq. (9) but its associated time-reverse SDE is

refined as

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, k) - g^2(k)\nabla_{\boldsymbol{x}_t}\log q_k(\boldsymbol{x}_t|\boldsymbol{h}_t)\right]\mathrm{d}k + g(k)\mathrm{d}\boldsymbol{w}, \tag{23}$$

where $k \in [0, K]$ represents the SDE integral time. As a common practice, the conditional score function $\nabla_{\boldsymbol{x}_t}\log q_k(\boldsymbol{x}_t|\boldsymbol{h}_t)$ is approximated with a parameterized neural network $\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^k, \boldsymbol{h}_t, k)$. Inspired by WaveNet (van den Oord et al., 2016) and DiffWave (Kong et al., 2021), the neural network is designed to have eight connected residual blocks, while each block contains a bidirectional dilated convolution module, a gated activation unit, a skip-connection process, and a one-dimension (1D) convolutional neural network for output.

The objective function of ScoreGrad is a conditional modification of Eq. (12), computed as

$$\sum_{t=t_0}^{T} L_t(\boldsymbol{\theta}), \tag{24}$$

with $L_t(\boldsymbol{\theta})$ being

$$\mathbb{E}_{k,\boldsymbol{x}_t^0,\boldsymbol{x}_t^k}\left[\delta(k)\left\|\boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{x}_t^k, \boldsymbol{h}_t, k) - \nabla_{\boldsymbol{x}_t}\log q_{0k}(\boldsymbol{x}_t|\boldsymbol{x}_t^0)\right\|^2\right]. \tag{25}$$

Up to this point, we use only the general expression of SDE for simple illustration. In the training process, one shall decide the specific type of SDE to use. Potential options include VE SDE, VP SDE, and sub-VP SDE (Yang S et al., 2021). The optimization varies depending on the chosen SDE, because different SDEs lead to different forward transition kernels $q(\boldsymbol{x}_t^k|\boldsymbol{x}_t^{k-1})$ and also different perturbation distributions $q_{0k}(\boldsymbol{x}_t|\boldsymbol{x}_t^0)$. Finally, for forecasting, ScoreGrad uses the predictor-corrector sampler as in Yang S et al. (2021) to sample from the time-reverse SDE.

### 3.4 D$^3$VAE

In practice, we may encounter the challenge of insufficient observations. If historical multivariate time series were recorded based on a short period, they will be prone to a significant level of noise due to measurement errors, sampling variability, and randomness from other sources. To address the problem of limited and noisy time series, D$^3$VAE, proposed by Li Y et al. (2022), employs a coupled diffusion process for data augmentation, and then uses a bidirectional variational autoencoder (BVAE) together with denoising score matching to clear the noise. In addition, D$^3$VAE considers disentangling latent variables by minimizing the overall correlation for better interpretability and stability of predictions. Moreover, the mean square error (MSE) between the prediction and actual observations in the prediction window is included in the objective function, further emphasizing the role of supervision.

Assuming that the prediction window can be generated from a set of latent variables $\boldsymbol{Z}$ that follows a Gaussian distribution $q(\boldsymbol{Z}|\boldsymbol{x}_{1:t_0-1}^0)$, the conditional distribution of $\boldsymbol{Z}$ is approximated with $p_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{x}_{1:t_0-1}^0)$, where $\boldsymbol{\phi}$ denotes learnable parameters. Then, the forecasting time series $\hat{\boldsymbol{x}}_{t_0:T}$ can be generated from the estimated target distribution, given by $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_0:T}^0|\boldsymbol{Z})$. It is not difficult to see that the prediction window is still predicted based on the context window, however, with latent variables $\boldsymbol{Z}$ as an intermediate.

In the coupled diffusion process, we inject noises separately into the context window and the prediction window. Different from TimeGrad which injects noise in the observation at each time point individually, the coupled diffusion process is applied to the whole period. For the context window, the same kernel as Eq. (2) is applied such that

$$\boldsymbol{x}_{1:t_0-1}^k = \sqrt{\tilde{\alpha}_k}\boldsymbol{x}_{1:t_0-1}^0 + \sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}, \tag{26}$$

where $\boldsymbol{\epsilon}$ denotes the standard Gaussian noise but with a matrix rather than a vector form.

The diffusion process is further applied to the prediction window with adjusted noise levels $\alpha_k' > \alpha_k$. Let $\tilde{\alpha}_k' \coloneqq \prod_{i=1}^k \alpha_i'$. Then we have

$$\boldsymbol{x}_{t_0:T}^k = \sqrt{\tilde{\alpha}_k'}\boldsymbol{x}_{t_0:T}^0 + \sqrt{1-\tilde{\alpha}_k'}\boldsymbol{\epsilon}. \tag{27}$$

This diffusion process simultaneously augments the context window and the prediction window, thus improving the generalization ability for short-time-series forecasting. In addition, it was proven by Li Y et al. (2022) that the uncertainty caused by the generative model and the inherent noises in the observed data can both be mitigated by the coupled diffusion process.

The backward process is accomplished with two steps. The first step is to predict $\boldsymbol{x}_{t_0:T}^k$ with a BVAE as the one used in Vahdat and Kautz (2020), which is composed of an encoder and a decoder with multiple residual blocks and takes the disturbed context

window $\boldsymbol{x}^k_{1:t_0-1}$ as input. The latent variables in $\boldsymbol{Z}$ are gradually generated and fed into the model in a summation manner. The output of this process is the predicted disturbed prediction window $\hat{\boldsymbol{x}}^k_{t_0:T}$. The second step involves further cleaning of the predicted data with a denoising score-matching module. Specifically, the final prediction is obtained via a single-step gradient jump (Saremi and Hyvärinen, 2019):

$$\hat{\boldsymbol{x}}^0_{t_0:T} \leftarrow \hat{\boldsymbol{x}}^k_{t_0:T} - \sigma_0^2 \nabla_{\hat{\boldsymbol{x}}^k_{t_0:T}} E(\hat{\boldsymbol{x}}^k_{t_0:T}; \mathrm{e}),$$

where $\sigma_0$ is prescribed and $E(\hat{\boldsymbol{x}}^k_{t_0:T}; \mathrm{e})$ is the energy function.

Disentanglement of latent variables $\boldsymbol{Z}$ can efficiently enhance the model interpretability and reliability for prediction (Li YN et al., 2021). It is measured by the total correlation of the random latent variables $\boldsymbol{Z}$. Generally, a lower total correlation implies better disentanglement and is a signal of useful information. The computation of total correlation happens synchronously with the BVAE module.

The objective function of D$^3$VAE consists of four components. It can be written as

$$\begin{aligned} &w_1 D_{\mathrm{KL}}\big(q(\boldsymbol{x}^k_{t_0:T}) \| p_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^k_{t_0:T})\big) \\ &+w_2 \mathcal{L}_{\mathrm{DSM}} + w_3 \mathcal{L}_{\mathrm{TC}} + \mathcal{L}_{\mathrm{MSE}}, \end{aligned} \tag{28}$$

where $w_1, w_2$, and $w_3$ are trade-off parameters that assign significance levels to the components. The first component $D_{\mathrm{KL}}\big(q(\boldsymbol{x}^k_{t_0:T}) \| p_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}^k_{t_0:T})\big)$ matches the estimated target distribution with the true distribution of the prediction window. The last three components, $\mathcal{L}_{\mathrm{DSM}}$, $\mathcal{L}_{\mathrm{TC}}$, and $\mathcal{L}_{\mathrm{MSE}}$, correspond to the denoising score matching (DSM) module, disentanglement of latent variables, and MSE between the prediction and the truth, respectively. The parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are learned together in the training process. Eventually, forecasting samples are generated from the learned distribution $p_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{x}^0_{1:t_0-1})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}^0_{t_0:T}|\boldsymbol{Z})$.

### 3.5 DSPD

Multivariate time-series data can be considered as a record of value changes for multiple features of an entity of interest. Data are collected from the same entity, and the measuring tools normally stay unchanged during the whole observed time period. So, assuming that the change of variables over time is smooth, the time-series data can be modeled as values from an underlying continuous function (Biloš et al., 2023). In this case, the context window is expressed as $\boldsymbol{X}^0_{\mathrm{c}} = \{\boldsymbol{x}(1), \boldsymbol{x}(2), ..., \boldsymbol{x}(t_0-1)\}$ and the prediction window becomes $\boldsymbol{X}^0_{\mathrm{p}} = \{\boldsymbol{x}(t_0), \boldsymbol{x}(t_0+1), ..., \boldsymbol{x}(T)\}$, where $\boldsymbol{x}(\cdot)$ is a continuous function of the time point $t$.

Different from traditional diffusion models, the diffusion and reverse processes are no longer applied to vector observations at each time point. Alternatively, the target of interest is the continuous function $\boldsymbol{x}(\cdot)$, which means noises will be injected and removed from a function rather than a vector. Therefore, a continuous noise function $\boldsymbol{\epsilon}(\cdot)$ should take the place of the noise vector $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This function should be both continuous and tractable, such that it accounts for the correlation between measurements and enables training and sampling. These requirements are effectively satisfied by designing a Gaussian stochastic process $\boldsymbol{\epsilon}(\cdot) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{\Sigma})$ (Biloš et al., 2023).

Discrete stochastic process diffusion (DSPD) is built upon the DDPM formulation but with the stochastic process $\boldsymbol{\epsilon}(\cdot) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{\Sigma})$. It is a delight that DSPD is only slightly different from DDPM in terms of implementation. Specifically, DSPD simply replaces the commonly applied noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with the noise function $\boldsymbol{\epsilon}(\cdot)$ whose discretized form is $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$. Let $\boldsymbol{X}^0$ be an observed multivariate time series in a certain period of time $\boldsymbol{T}' = \{t'_1, t'_2, ..., t'_T\}$, which means $\boldsymbol{X}^0 = \{\boldsymbol{x}(t'_1), \boldsymbol{x}(t'_2), ..., \boldsymbol{x}(t'_T)\}$. In the forward process, noise is injected through the transition kernel

$$q(\boldsymbol{X}^k|\boldsymbol{X}^0) = \mathcal{N}(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0, (1-\tilde{\alpha}_k)\boldsymbol{\Sigma}). \tag{29}$$

Then, the following backward transition kernel is applied to recover the original data:

$$p_{\boldsymbol{\theta}}(\boldsymbol{X}^{k-1}|\boldsymbol{X}^k) = \mathcal{N}(\mu_{\boldsymbol{\theta}}(\boldsymbol{X}^k, k), (1-\alpha_k)\boldsymbol{\Sigma}). \tag{30}$$

Consequently, the objective function should be changed as

$$\mathbb{E}_{k, \boldsymbol{X}^0, \boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0 + \sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}, k\right)\right\|^2\right], \tag{31}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ with the covariance matrix from the Gaussian process $\mathcal{GP}(\boldsymbol{0}, \boldsymbol{\Sigma})$.

Forecasting via DSPD is very similar to TimeGrad. As before, the aim is still to learn the

conditional probability $q(\boldsymbol{X}_{\mathrm{p}}^0|\boldsymbol{X}_{\mathrm{c}}^0)$, but there are two major improvements. First, the prediction is available for any future time point in the continuous time interval. Second, instead of step-by-step forecasting, DSPD can generate samples for multiple time points in one run. By adding the historical condition into the fundamental objective function in Eq. (31), the objective function for DSPD forecasting is then given by

$$\mathbb{E}_{k,\boldsymbol{X}_{\mathrm{p}}^0,\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}_{\mathrm{p}}^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{X}_{\mathrm{c}}^0,k\right)\right\|^2\right]. \tag{32}$$

Finally, supposing that the last context window is $\tilde{\boldsymbol{X}}_{\mathrm{c}}$, the sampling process to forecast for the prediction target $\tilde{\boldsymbol{X}}_{\mathrm{p}}$ in time interval $\tilde{\boldsymbol{T}}$ is given by

$$\tilde{\boldsymbol{X}}_{\mathrm{p}}^k\leftarrow\frac{\left(\tilde{\boldsymbol{X}}_{\mathrm{p}}^{k+1}-\zeta(k+1)\boldsymbol{L}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{X}}_{\mathrm{p}}^{k+1},\tilde{\boldsymbol{X}}_{\mathrm{c}},k+1)\right)}{\sqrt{\alpha_{k+1}}}+(1-\alpha_{k+1})\boldsymbol{z},$$

where $\boldsymbol{L}$ is from the factorization of the covariance matrix $\boldsymbol{\Sigma}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$, the last diffusion output is generated as $\tilde{\boldsymbol{X}}_{\mathrm{p}}^K\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$, and $\boldsymbol{z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{\Sigma})$.

Similar to the extension from TimeGrad to ScoreGrad, the continuous noise function can be adapted to the SDE framework, thus leading to the continuous stochastic process diffusion (CSPD) model (Biloš et al., 2023). The diffusion process of CSPD introduces the factorized covariance matrix $\boldsymbol{\Sigma}=\boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ to the VP SDE (see Section 2.4) as

$$\mathrm{d}\boldsymbol{X}=-\frac{1}{2}\alpha(k)\boldsymbol{X}\mathrm{d}k+\sqrt{\alpha(k)}\boldsymbol{L}\mathrm{d}\boldsymbol{w}, \tag{33}$$

where $\boldsymbol{w}$ is a matrix that represents a standard Wiener process. The perturbation distribution in the objective function is then modified as

$$\begin{aligned}q_{0k}(\boldsymbol{X}^k|\boldsymbol{X}^0)=&\mathcal{N}\big(\boldsymbol{X}^k;\boldsymbol{X}^0\mathrm{e}^{-\frac{1}{2}\int_0^k\alpha(s)\mathrm{d}s},\\&[1-\mathrm{e}^{-\int_0^k\alpha(s)\mathrm{d}s}]\boldsymbol{\Sigma}\big).\end{aligned} \tag{34}$$

### 3.6 TimeDiff

TimeGrad adopts the autoregressive sampling process, in which the prediction of the former time point should be sent back to the encoding algorithm to compute the condition for sampling the next prediction. Therefore, errors can accumulate with the increase of prediction interval length, restricting TimeGrad's ability to forecast long-term time series. In addition, the autoregressive sampling process is very slow compared to one-shot generative methods (i.e., methods that generate the prediction for the whole prediction interval in one run). To solve these problems, a non-autoregressive method called TimeDiff was proposed (Shen and Kwok, 2023).

In TimeGrad, the condition of target conditional distribution is the hidden states of the history window obtained via an RNN module. Alternatively, TimeDiff combines two sources of information as the condition, including a future mixup and a future approximation. Future mixup, inspired by Zhang HY et al. (2018), integrates past and future information at each diffusion step $k$ via a random mask $\boldsymbol{m}^k\in[0,1)^{D\times(T-t_0+1)}$ sampled from the uniform distribution on $[0,1)$ as follows:

$$\boldsymbol{h}_{\mathrm{mix}}=\boldsymbol{m}^k\odot\mathrm{Conv}(\boldsymbol{x}_{1:t_0-1}^0)+(1-\boldsymbol{m}^k)\odot\boldsymbol{x}_{t_0:T}^0, \tag{35}$$

where $\mathrm{Conv}(\cdot)$ is a convolution network for encoding local temporal patterns and long-term dependencies. In the sampling process, where the future values are unknown, the mixup is simply set as

$$\boldsymbol{h}_{\mathrm{mix}}=\mathrm{Conv}(\boldsymbol{x}_{1:t_0-1}^0). \tag{36}$$

Then future approximation is obtained via simple linear calculation to generate an initial guess for the prediction window:

$$\boldsymbol{h}_{\mathrm{apx}}=\sum_{i=1}^{t_0-1}\boldsymbol{W}_i\odot\boldsymbol{X}_i+\boldsymbol{B}_i, \tag{37}$$

where $\boldsymbol{X}_i\in\mathbb{R}^{D\times(T-t_0+1)}$ is a matrix of $(T-t_0+1)$ copies of $\boldsymbol{x}_i^0$, and $\boldsymbol{W}_i$ and $\boldsymbol{B}_i$ are learnable matrices. The future approximation is actually a linear autoregressive model, which can reduce the disharmony between the history and prediction windows (Lugmayr et al., 2022; Shen and Kwok, 2023). However, this model does not require autoregressive sampling because it is based only on historical information. Hence, TimeDiff effectively avoids the error accumulation and slow sampling caused by the autoregressive mechanism. The last step to construct the condition is to vertically concatenate future mixup and future approximation as

$$\boldsymbol{h}=\mathrm{concat}(\boldsymbol{h}_{\mathrm{mix}},\boldsymbol{h}_{\mathrm{apx}})\in\mathbb{R}^{2D\times(T-t_0+1)}. \tag{38}$$

With the new condition $\boldsymbol{h}$, the objective function is modified as

$$\begin{aligned}\mathbb{E}_{k,\boldsymbol{x}_{t_0:T}^0,\boldsymbol{\epsilon}}\Big[\delta(k)\Big\|\boldsymbol{x}_{t_0:T}^0-\boldsymbol{x}_{\boldsymbol{\theta}}\Big(&\sqrt{\tilde{\alpha}_k}\boldsymbol{x}_{t_0:T}^0\\&+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{x}_{1:t_0-1}^0,k,\boldsymbol{h}\Big)\Big\|^2\Big].\end{aligned} \tag{39}$$

Instead of approximating the noise injected at each diffusion step, TimeDiff's objective function aims to approximate the original data directly. It is assumed that the mean of the reverse transition kernel in Eq. (3) is alternatively approximated with $\boldsymbol{x_\theta}$ rather than $\boldsymbol{\epsilon_\theta}$ as

$$\begin{aligned}\boldsymbol{\mu_\theta}(\boldsymbol{x}^k,k)=&\frac{\sqrt{\alpha_k}(1-\tilde{\alpha}_{k-1})}{1-\tilde{\alpha}_k}\boldsymbol{x}^k\\&+\frac{\sqrt{\tilde{\alpha}_{k-1}}(1-\alpha_k)}{1-\tilde{\alpha}_k}\boldsymbol{x_\theta}(\boldsymbol{x}^k,k,\boldsymbol{h}).\end{aligned}\tag{40}$$

Accordingly, this mean is also applied in the sampling process with the learned data matching network $\boldsymbol{x_\theta}$.

### 3.7 DiffSTG

Spatio-temporal graphs (STGs) are a special type of multivariate time series that encodes spatial and temporal relationships and interactions among different entities in a graph structure (Wen et al., 2023). They are commonly observed in real-life applications such as traffic flow prediction (Li YG et al., 2018) and weather forecasting (Simeunović et al., 2022). Suppose we have $N$ entities of interest, such as traffic sensors or companies in the stock market. We can model these entities and their underlying relationships as a graph $\mathcal{G}=\{\mathcal{V},\mathcal{E},\boldsymbol{W}\}$, where $\mathcal{V}$ is a set of $N$ nodes as representations of entities, $\mathcal{E}$ is a set of links that indicates the relationship between nodes, and $\boldsymbol{W}$ is a weighted adjacency matrix that describes the graph topological structure. Multivariate time series observed at all entities are models as graph signals $\boldsymbol{X}_{\mathrm{c}}^0=\{\boldsymbol{x}_1^0,\boldsymbol{x}_2^0,...,\boldsymbol{x}_{t_0-1}^0|\boldsymbol{x}_t^0\in\mathbb{R}^{D\times N}\}$, which means we have $D$-dimensional observations from $N$ entities at each time point $t$. A visualized illustration of STGs is provided in Fig. 3. Identical to the previous problem formulation, the aim of STG forecasting is to predict $\boldsymbol{X}_{\mathrm{p}}^0=\{\boldsymbol{x}_{t_0}^0,\boldsymbol{x}_{t_0+1}^0,...,\boldsymbol{x}_T^0|\boldsymbol{x}_t^0\in\mathbb{R}^{D\times N}\}$ based on the historical information $\boldsymbol{X}_{\mathrm{c}}$. Nevertheless, except for the time dependency on historical observations, we also need to consider the spatial interactions between different entities represented by the graph topology.

DiffSTG applies diffusion models on STG forecasting with a graph-based noise-matching network called UGnet (Wen et al., 2023). The idea of DiffSTG can be regarded as the extension of DDPM-based forecasting to STGs with an additional condition on the graph structure, which means the target distribution in Eq. (17) is approximated alternatively by

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t_0:T}^0|\boldsymbol{x}_{1:t_0-1}^0,\boldsymbol{W}).\tag{41}$$

Accordingly, the objective function is changed as

$$\begin{aligned}\mathbb{E}_{k,\boldsymbol{x}_{t_0:T}^0,\boldsymbol{\epsilon}}\Big[\delta(k)\Big\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon_\theta}\Big(&\sqrt{\tilde{\alpha}_k}\boldsymbol{x}_{t_0:T}^0\\&+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{x}_{1:t_0-1}^0,k,\boldsymbol{W}\Big)\Big\|^2\Big].\end{aligned}\tag{42}$$

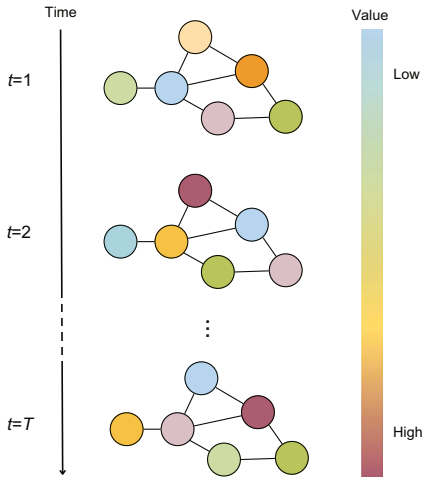


**Fig. 3 Illustration of spatio-temporal graphs**
References to color refer to the online version of this figure

The objective function in Eq. (42) actually treats the context window and the prediction window as samples from two separate sample spaces, namely, $\boldsymbol{X}_{\mathrm{c}}^0\in\mathcal{X}_{\mathrm{c}}$ and $\boldsymbol{X}_{\mathrm{p}}^0\in\mathcal{X}_{\mathrm{p}}$ with $\mathcal{X}_{\mathrm{c}}$ and $\mathcal{X}_{\mathrm{p}}$ being two individual sample spaces. However, considering the fact that the context and prediction intervals are consecutive, it may be more reasonable to treat the two windows as a complete sample from the same sample space. To this end, Wen et al. (2023) reformulated the forecasting problem and revised the approximation in Eq. (41) as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{1:T}^0|\boldsymbol{x}_{1:t_0-1}^0,\boldsymbol{W}),\tag{43}$$

in which the history condition is derived by masking the future time series from the whole time period. The associated objective function is

$$\begin{aligned}\mathbb{E}_{k,\boldsymbol{x}_{1:T}^0,\boldsymbol{\epsilon}}\Big[\delta(k)\Big\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon_\theta}\Big(&\sqrt{\tilde{\alpha}_k}\boldsymbol{x}_{1:T}^0\\&+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{x}_{1:t_0-1}^0,k,\boldsymbol{W}\Big)\Big\|^2\Big].\end{aligned}\tag{44}$$

The training process is quite straightforward and follows common practice, but note that the sample

generated in the forecasting process includes both historical and future values. So, we need to take out the forecasting target in the sample as the prediction.

Now, there is only one remaining problem. How to encode the graph structural information in the noise-matching network $\boldsymbol{\epsilon_\theta}$? Wen et al. (2023) proposed UGnet, a U-Net-based network architecture (Ronneberger et al., 2015) combined with a graph neural network (GNN) to process time dependency and spatial relationships simultaneously. UGnet takes $\boldsymbol{x}^k_{1:T}, \boldsymbol{x}^0_{1:t_0-1}, k$ and $\boldsymbol{W}$ as inputs and then outputs the prediction of the associated error $\boldsymbol{\epsilon}$.

### 3.8 GCRDD

The graph convolutional recurrent denoising diffusion (GCRDD) model is another diffusion-based model for STG forecasting (Li RK et al., 2023). It differs from DiffSTG in that it uses hidden states from a recurrent component to store historical information as TimeGrad and employs a different network structure for the noise-matching term $\boldsymbol{\epsilon_\theta}$. Note that the notations related to STGs here follow Section 3.7.

GCRDD approximates the target distribution with a probabilistic density function conditional on the hidden states and graph structure as follows:

$$\prod_{t=t_0}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}^0_t|h_{t-1}, \boldsymbol{W}), \tag{45}$$

where the hidden state is computed with a graph-modified GRU, written as

$$\boldsymbol{h}_t = \mathrm{GraphGRU}_{\boldsymbol{\theta}}(\boldsymbol{x}^0_t, \boldsymbol{h}_{t-1}, \boldsymbol{W}). \tag{46}$$

The graph-modified GRU replaces the weight matrix multiplication in a traditional GRU (Chung et al., 2014) with graph convolution such that both temporal and spatial information is stored in the hidden state. The objective function of GCRDD adopts a similar form of TimeGrad but with additional graph structural information in the noise-matching network:

$$\begin{aligned}\mathbb{E}_{k,\boldsymbol{x}^0_t,\boldsymbol{\epsilon}}\Big[\delta(k)\Big\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon_\theta}\Big(\sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0_t \\ +\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}, \boldsymbol{h}_{t-1}, \boldsymbol{W}, k\Big)\Big\|^2\Big].\end{aligned} \tag{47}$$

For the noise-matching term, GCRDD adopts a variant of DiffWave (Kong et al., 2021) that incorporates a graph convolution component to process spatial information in $\boldsymbol{W}$. STG forecasting via GCRDD is the same as TimeGrad except that the sample generated at each time point is a matrix rather than a vector.

## 4 Time-series imputation

In real-world problem settings, we usually encounter the challenge of missing values. When collecting time-series data, the collection conditions may change over time, which makes it difficult to ensure the completeness of observation. In addition, accidents such as sensor failures and human errors may result in missing historical records. Missing values in time-series data normally have a negative impact on the accuracy of analysis and forecasting because the lack of partial observations makes the inference and conclusions vulnerable in future generalization.

Time-series imputation aims to fill in the missing values in incomplete time-series data. Many previous studies have focused on designing deep learning based algorithms for time-series imputation (Osman et al., 2018). Most existing approaches involve the RNN architecture to encode time dependency in the imputation task (Cao et al., 2018; Che et al., 2018; Luo YH et al., 2018; Yoon et al., 2019). Except for these deterministic methods, probabilistic imputation models such as GP-VAE (Fortuin et al., 2020) and V-RIN (Mulyadi et al., 2022) have shown their practical value in recent years. As a rising star in probabilistic models, diffusion models have also been applied to time-series imputation tasks (Tashiro et al., 2021; Alcaraz and Strodthoff, 2023). Compared with other probabilistic approaches, diffusion-based imputation enjoys high flexibility in the assumption of the true data distribution. In this section, we will cover four diffusion-based methods, including three for multivariate time-series imputation and one for STG imputation.

### 4.1 Problem formulation

We still consider the multivariate time series $\boldsymbol{X}^0 = \{\boldsymbol{x}^0_1, \boldsymbol{x}^0_2, ..., \boldsymbol{x}^0_T | \boldsymbol{x}^0_i \in \mathbb{R}^D\}$. It is not difficult to see that $\boldsymbol{X}^0 \in \mathbb{R}^{D\times T}$, where $D$ is the number of features, and $T$ is the number of time points in the period $[1, T]$. Different from time-series forecasting in which we assume that all elements in $\boldsymbol{X}^0$ are known, here we have an incomplete matrix of

observations. In the imputation task, we try to predict the values of missing data by exploring the information from some observed data. We denote the observed data as $\boldsymbol{X}^0_{\mathrm{ob}}$ and the missing data as $\boldsymbol{X}^0_{\mathrm{ms}}$. Then, the imputation task is to find the conditional probability distribution $q(\boldsymbol{X}^0_{\mathrm{ms}}|\boldsymbol{X}^0_{\mathrm{ob}})$.

For practical purposes, zero padding is applied to the incomplete matrix $\boldsymbol{X}^0$ such that all missing entries are assigned 0. In addition, a 0–1 matrix $\boldsymbol{M}\in\mathbb{R}^{D\times T}$ is constructed as a mask to denote the position of missing values. Specifically, the element in $\boldsymbol{M}$ is 0 when the corresponding value in $\boldsymbol{X}^0$ is missing and 1 otherwise.

Both $\boldsymbol{X}^0_{\mathrm{ob}}$ and $\boldsymbol{X}^0_{\mathrm{ms}}$ have the same dimension as $\boldsymbol{X}^0$. In the training process, a fraction of the actually observed data in $\boldsymbol{X}^0$ is randomly selected to be the true values of missing data, and the rest of the observed data will be the condition for prediction. A training mask $\boldsymbol{M}'\in\mathbb{R}^{D\times T}$ is introduced to obtain $\boldsymbol{X}^0_{\mathrm{ob}}$ and $\boldsymbol{X}^0_{\mathrm{ms}}$. It is constructed by assigning 1 to entries that correspond to the remaining observed data in $\boldsymbol{X}^0$. Then, $\boldsymbol{X}^0_{\mathrm{ob}}$ is computed as $\boldsymbol{X}^0_{\mathrm{ob}}=\boldsymbol{M}'\odot\boldsymbol{X}^0$, and $\boldsymbol{X}^0_{\mathrm{ms}}$ is computed as $\boldsymbol{X}^0_{\mathrm{ms}}=(\boldsymbol{M}-\boldsymbol{M}')\odot\boldsymbol{X}^0$, where $\odot$ denotes the element-wise matrix multiplication. In the forecasting process, on the other hand, all actually observed data are used as the condition, which means $\boldsymbol{X}^0_{\mathrm{ob}}=\boldsymbol{M}\odot\boldsymbol{X}^0$.

Note that the problem formulation here is only a typical case. Later, in Section 4.4, we will introduce another formulation that takes the whole time-series matrix $\boldsymbol{X}^0$ as the target for generation.

### 4.2 CSDI

Conditional-score-based diffusion model for imputation (CSDI) is the pioneering work on diffusion-based time-series imputation (Tashiro et al., 2021). Identical to TimeGrad, the basic diffusion formulation of CSDI is DDPM. However, as we discussed in Section 3.2, the historical information is encoded by an RNN module in TimeGrad, which hampers the direct extension of TimeGrad to imputation tasks because the computation of hidden states may be interrupted by missing values in the context window.

CSDI applies the diffusion and reverse processes to the matrix of missing data, $\boldsymbol{X}^0_{\mathrm{ms}}$. Correspondingly, the reverse transition kernel is refined as a probabilistic distribution conditional on $\boldsymbol{X}^0_{\mathrm{ob}}$:

$$\begin{aligned}&p_{\boldsymbol{\theta}}(\boldsymbol{X}^{k-1}_{\mathrm{ms}}|\boldsymbol{X}^{k}_{\mathrm{ms}},\boldsymbol{X}^0_{\mathrm{ob}})\\&\quad=\mathcal{N}(\boldsymbol{X}^{k-1}_{\mathrm{ms}};\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{X}^{k}_{\mathrm{ms}},k|\boldsymbol{X}^0_{\mathrm{ob}}),\sigma_{\boldsymbol{\theta}}(\boldsymbol{X}^{k}_{\mathrm{ms}},k|\boldsymbol{X}^0_{\mathrm{ob}})\boldsymbol{I}),\end{aligned}\tag{48}$$

where

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{X}^{k}_{\mathrm{ms}},k|\boldsymbol{X}^0_{\mathrm{ob}})=\frac{1}{\sqrt{\alpha_k}}\left(\boldsymbol{X}^{k}_{\mathrm{ms}}-\zeta(k)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{X}^{k}_{\mathrm{ms}},k|\boldsymbol{X}^0_{\mathrm{ob}})\right),\tag{49}$$

with $\boldsymbol{X}^k_{\mathrm{ms}}=\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0_{\mathrm{ms}}+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon}$. Note that the variance term here is different from the version in DDPM (Ho et al., 2020). Previously, the variance term is defined with some pre-specified constant $\sigma_k$ with $k=1,2,...,K$, implying that the variance is treated as a hyperparameter. CSDI, however, defines a learnable version $\sigma_{\boldsymbol{\theta}}$ with parameter $\boldsymbol{\theta}$. Both ways are acceptable and have their respective practical value.

The objective function of CSDI is given by

$$\begin{aligned}\mathbb{E}_{k,\boldsymbol{X}^0_{\mathrm{ms}},\boldsymbol{\epsilon}}\Big[\delta(k)\Big\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\Big(&\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0_{\mathrm{ms}}\\&+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{X}^0_{\mathrm{ob}},k\Big)\Big\|^2\Big].\end{aligned}\tag{50}$$

The noise-matching network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ adopts the DiffWave (Kong et al., 2021) architecture by default. After training, the imputation is accomplished by generating the target matrix of missing values in the same way as DDPM. $\boldsymbol{X}^0_{\mathrm{ob}}$ in the sampling process is identical to the zero padding version of the original time-series matrix $\boldsymbol{X}^0$, where all missing values are assigned to 0. The starting point of the sampling process is a random Gaussian imputation target $\boldsymbol{X}^K_{\mathrm{ms}}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$. Then, for $k=K-1,K-2,...,0$, the algorithm computes

$$\boldsymbol{X}^k_{\mathrm{ms}}\leftarrow\frac{\left(\boldsymbol{X}^{k+1}_{\mathrm{ms}}-\zeta(k+1)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{X}^{k+1}_{\mathrm{ms}},\boldsymbol{X}^0_{\mathrm{ob}},k+1)\right)}{\sqrt{\alpha_{k+1}}}+\sigma_{\boldsymbol{\theta}}\boldsymbol{Z},$$

where $\boldsymbol{Z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ for $k=K-1,K-2,...,1$, and $\boldsymbol{Z}=\boldsymbol{0}$ for $k=0$.

### 4.3 DSPD

Here we discuss the simple extension of DSPD and CSPD in Section 3.5 to time-series imputation tasks. Because the rationale behind the extension of these two models is almost identical, we focus only on DSPD for illustration. The assumption of DSPD states that the observed time series is formed by values of a continuous function $\boldsymbol{x}(\cdot)$ of time $t$. Therefore,

the missing values can be obtained by computing the values of this continuous function at the corresponding time points. Recall that DSPD uses the covariance matrix $\boldsymbol{\Sigma}$ instead of the DDPM variance $\sigma_k \boldsymbol{I}$ or $\sigma_{\boldsymbol{\theta}}$ in the backward process. Therefore, one may apply DSPD to imputation tasks in a manner similar to CSDI by replacing the variance term in Eq. (48) with the covariance matrix $\boldsymbol{\Sigma}$. According to Biloš et al. (2023), the continuous noise process is a more natural choice than the discrete noise vector because it takes account of the irregularity in the measurement when collecting the time-series data.

### 4.4 SSSD

Structured state space diffusion (SSSD) differs from the aforementioned two methods by having the whole time-series matrix $\boldsymbol{X}^0$ as the generative target in its diffusion module (Alcaraz and Strodthoff, 2023). The name, "structured state space diffusion," comes from the design of the noise-matching network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, which adopts the state space model (Gu et al., 2022) as the internal architecture. As a matter of fact, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ can also take other architectures such as the DiffWave-based network in CSDI (Tashiro et al., 2021) and SaShiMi, a generative model for sequential data (Goel et al., 2022). However, the authors of SSSD have shown empirically that the structured state space model generally generates the best imputation outcome compared with other architectures (Alcaraz and Strodthoff, 2023). To emphasize the difference between this method and other diffusion-based approaches, we primarily focus on the unique problem formulation used by SSSD.

As we mentioned, the generative target of SSSD is the whole time-series matrix, $\boldsymbol{X}^0 \in \mathbb{R}^{D\times T}$, rather than a matrix that particularly represents the missing values. For the purpose of training, $\boldsymbol{X}^0$ is also processed with zero padding. The conditional information, in this case, is from a concatenated matrix $\boldsymbol{X}_{\mathrm{c}}^0 = \mathrm{Concat}(\boldsymbol{X}^0 \odot \boldsymbol{M}_{\mathrm{c}}, \boldsymbol{M}_{\mathrm{c}})$, where $\boldsymbol{M}_{\mathrm{c}}$ is a 0–1 matrix indicating the position of observed values as the condition. The element in $\boldsymbol{M}_{\mathrm{c}}$ can only be 1 if its corresponding value in $\boldsymbol{X}^0$ is known.

There are two options for the objective function used in the training process. Similar to other approaches, the objective function can be a simple conditional variant of the DDPM objective function:

$$\mathbb{E}_{k,\boldsymbol{X}^0,\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{X}_{\mathrm{c}}^0,k\right)\right\|^2\right], \tag{51}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is built upon the structured state space model by default. The other choice of the objective function is computed with only known data, which is mathematically expressed as

$$\mathbb{E}_{k,\boldsymbol{X}^0,\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}\odot\boldsymbol{M}_{\mathrm{c}}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{X}_{\mathrm{c}}^0,k\right)\odot\boldsymbol{M}_{\mathrm{c}}\right\|^2\right]. \tag{52}$$

According to Alcaraz and Strodthoff (2023), the second objective function is typically a better choice in practice. For forecasting, SSSD employs the usual sampling algorithm and applies to the unknown entries in $\boldsymbol{X}^0$, namely, $(1-\boldsymbol{M}_{\mathrm{c}})\odot\boldsymbol{X}^0$.

An interesting point proposed along with SSSD is that imputation models can also be applied to forecasting tasks. This is because future time series can be viewed as a long block of missing values on the right of $\boldsymbol{X}^0$. Nevertheless, experiments have shown that the diffusion-based approaches underperform other methods such as the autoformer (Wu et al., 2021) in forecasting tasks.

### 4.5 PriSTI

PriSTI is a diffusion-based model for STG imputation (Liu MZ et al., 2023). However, different from DiffSTG, the existing framework of PriSTI is designed for STGs with only one feature, which means the graph signal has the form $\boldsymbol{X}^0 = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, ..., \boldsymbol{x}_T^0\} \in \mathbb{R}^{N\times T}$. Each vector $\boldsymbol{x}_t^0 \in \mathbb{R}^N$ represents the observed values of $N$ nodes at time point $t$. This kind of data is often observed in traffic prediction (Li YG et al., 2018) and weather forecasting (Yi et al., 2016). METR-LA, for example, is an STG dataset that contains traffic speed collected by 207 sensors on a Los Angeles highway in a four-month time period (Li YG et al., 2018). There is only one node attribute, that is, traffic speed. However, unlike the multivariate time-series matrix, where features (sensors in this case) are usually assumed to be uncorrelated, the geographic relationship between different sensors is stored in the weighted adjacency matrix $\boldsymbol{W}$, allowing a more pertinent representation of real-world traffic data.

The number of nodes $N$ can be considered as the number of features $D$ in CSDI. The only difference

is that PriSTI incorporates the underlying relationship between each pair of nodes in the conditional information for imputation. So, the problem formulation adopted by PriSTI is the same as our discussion in Section 4.1, and thus the goal is still to find $q(\boldsymbol{X}^0_{\rm ms}|\boldsymbol{X}^0_{\rm ob})$.

To encode graph structural information, the mean in Eq. (49) is modified as

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{X}^k_{\rm ms},k|\boldsymbol{X}^0_{\rm ob},\boldsymbol{W})=\frac{1}{\sqrt{\alpha_k}}\left(\boldsymbol{X}^k_{\rm ms}-\zeta(k)\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{X}^k_{\rm ms},\boldsymbol{X}^0_{\rm ob},k,\boldsymbol{W})\right), \tag{53}$$

where $\boldsymbol{W}$ is the weighted adjacency matrix. Consequently, the objective function is changed to

$$\mathbb{E}_{k,\boldsymbol{X}^0_{\rm ms},\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{X}^0_{\rm ms}+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},\boldsymbol{X}^0_{\rm ob},k,\boldsymbol{W}\right)\right\|^2\right]. \tag{54}$$

The conditional information, $\boldsymbol{X}^0_{\rm ob}$, is processed with linear interpolation before it is fed into the algorithm to incorporate extra noises, which will enhance the denoising capability of the model and eventually lead to better consistency in prediction (Choi et al., 2022). The noise-matching network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$ is composed of two modules, including a conditional feature extraction module and a noise estimation module. The conditional feature extraction module takes the interpolated information $\boldsymbol{X}^0_{\rm ob}$ and adjacency matrix $\boldsymbol{W}$ as inputs and generates a global context with both spatial and temporal information as the condition for diffusion. Then, the noise estimation module uses this global context to estimate the injected noises with a specialized attention mechanism to capture temporal dependencies and geographic information. Ultimately, the STG imputation is fulfilled with the usual sampling process of DDPM, but with the specially designed noise-matching network here to incorporate the additional spatial relationship.

Because PriSTI works only for the imputation of STGs with a single feature, which is simply a special case of STGs, this model's practical value is somewhat limited. So, the extension of the idea here to more generalized STGs is a notable topic for future researchers.

## 5 Time-series generation

The rapid development of the machine learning paradigm requires high-quality data for different learning tasks in finance, economics, physics, and other fields. The performance of the machine learning model and algorithm may be highly subject to the underlying data quality. Time-series generation refers to the process of creating synthetic data that resembles the real-world time series. Because the time-series data are characterized by their temporal dependencies, the generation process usually requires the learning of underlying patterns and trends from past information.

Time-series generation is a developing topic in the literature and several methods exist (Yoon et al., 2019; Desai et al., 2021). Time-series data can be seen as a case of sequential data, whose generation usually involves the generative adversarial network (GAN) architecture (Mogren, 2016; Esteban et al., 2017; Donahue et al., 2019; Xu TL et al., 2020). Accordingly, TimeGAN is proposed to generate time-series data based on an integration of RNN and GAN for the purpose of processing time dependency and generation (Yoon et al., 2019). However, the GAN-based generative methods have been criticized because they are unstable (Chu et al., 2020) and subject to the model collapse issue (Xiao et al., 2022). Another way to generate time-series data stems from the VAE, leading to the so-called TimeVAE model (Desai et al., 2021). As a common shortcoming of VAE-based models, TimeVAE requires a user-defined distribution for its probabilistic process. Here we present a different probabilistic time-series generator that originates from diffusion models and is more flexible with the form of the target distribution (Coletta et al., 2023; Lim et al., 2023). This section aims to enlighten researchers about this new research direction, and we expect to see more derivative works in the future.

### 5.1 Problem formulation

With the multivariate time series $\boldsymbol{X}^0=\{\boldsymbol{x}^0_1,\boldsymbol{x}^0_2,...,\boldsymbol{x}^0_T|\boldsymbol{x}^0_i\in\mathbb{R}^D\}$, the time-series generation problem aims to synthesize time series $\boldsymbol{x}^0_{1:T}$ in the same time period. The problem formulation may vary in different papers. Here, we will focus on two different methods, conditional score based time-series generative model (TSGM) (Lim et al., 2023) and TimeDiff (Coletta et al., 2023). TSGM synthesizes time series by generating observation $\boldsymbol{x}^0_t$ at time point $t\in[2,T]$ with the consideration of its previous historical data $\boldsymbol{x}^0_{1:t-1}$. Correspondingly, the target distribution is the conditional density $q(\boldsymbol{x}^0_t|\boldsymbol{x}^0_{1:t-1})$

for $t \in [2,T]$, and the associated generative process involves the recursive sampling of $\boldsymbol{x}_t$ for all time points in the observed period. With DiffTime, alternatively, time-series generation is formed as a constrained optimization problem, and the condition in target distribution is designed based on specific constraints rather than just historical information. Details about the training and generation processes will be discussed in the following subsections.

### 5.2 TSGM

TSGM was proposed to conditionally generate each time-series observation based on the past generated observations (Lim et al., 2023). The TSGM architecture includes three components: an encoder, a decoder, and a conditional score-matching network. The pre-trained encoder is used to embed the underlying time series into a latent space. The conditional score-matching network is used to sample the hidden states, which are then converted to the time-series samples via the decoder.

Given the multivariate time series $\boldsymbol{X}^0 = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, ..., \boldsymbol{x}_T^0 | \boldsymbol{x}_i^0 \in \mathbb{R}^D\}$, the encoder **En** and decoder **De** enable the mapping between the time-series data and hidden states in a latent space. The mapping process can be defined as

$$\boldsymbol{h}_t = \mathbf{En}(\boldsymbol{h}_{t-1}^0, \boldsymbol{x}_t^0), \quad \hat{\boldsymbol{x}}_t^0 = \mathbf{De}(\boldsymbol{h}_t^0), \tag{55}$$

where $\hat{\boldsymbol{x}}_t^0$ refers to the reconstructed time-series data at time $t$, after the mapping process. This is a recursive process because both the encoder **En** and decoder **De** are constructed with the RNN structure. The training objective function $\mathcal{L}_{\mathrm{ED}}$ for both the encoder and decoder is defined as

$$\mathcal{L}_{\mathrm{ED}} = \mathbb{E}_{\boldsymbol{x}_{1:T}^0}\left[\|\hat{\boldsymbol{x}}_{1:T}^0 - \boldsymbol{x}_{1:T}^0\|_2^2\right]. \tag{56}$$

Given the auto-dependency characteristic of the time-series data, learning the conditional log-likelihood function is essential. To address this, the conditional score-matching network is designed based on the SDE formulation of diffusion models. Note that TSGM focuses on the generation of hidden states rather than producing the time series directly with the sampling process. At time step $t$, instead of applying the diffusion process to $\boldsymbol{x}_t^0$, the hidden state $\boldsymbol{h}_t^0$ is diffused to a Gaussian distribution by the following forward SDE:

$$\mathrm{d}\boldsymbol{h}_t = f(k, \boldsymbol{h}_t)\mathrm{d}k + g(k)\mathrm{d}\boldsymbol{\omega}, \tag{57}$$

where $k \in [0,K]$ refers to the integral time. With the diffused sample $\boldsymbol{h}_{1:t}^k$, the conditional score-matching network $s_{\boldsymbol{\theta}}$ learns the gradient of the conditional log-likelihood function with the following objective function:

$$\mathcal{L}_{\mathrm{Score}} = \mathbb{E}_{\boldsymbol{h}_{1:T}^0, k}\sum_{t=1}^{T}\left[\mathcal{L}(t,k)\right], \tag{58}$$

with

$$\mathcal{L}(t,k) = \mathbb{E}_{\boldsymbol{h}_t^k}\left[\delta(k)\|s_{\boldsymbol{\theta}}(\boldsymbol{h}_t^k, \boldsymbol{h}_{t-1}, k) - \nabla_{\boldsymbol{h}_t}\log q_{0k}(\boldsymbol{h}_t|\boldsymbol{h}_t^0)\|^2\right]. \tag{59}$$

The network architecture of $s_{\boldsymbol{\theta}}$ is designed based on U-Net (Ronneberger et al., 2015), which was adopted by the classic SDE model (Yang S et al., 2021).

In the training process, the encoder and decoder are pre-trained using the objective $\mathcal{L}_{\mathrm{ED}}$. They can also be trained simultaneously with the network $s_{\boldsymbol{\theta}}$, but Lim et al. (2023) showed that the pre-training generally led to better performance. Then, to learn the score-matching network, hidden states are first obtained through inputting the entire time series $\boldsymbol{x}_{1:T}^0$ into the encoder, and then fed into the training algorithm with the objective function $\mathcal{L}_{\mathrm{Score}}$. The time-series generation is achieved by sampling hidden states and then applying the decoder, where the sampling process is analogous to solving the time-reverse SDE.

The TSGM method can achieve state-of-the-art sampling quality and diversity, compared to a range of well-developed time-series generation methods. However, it is still subject to the fundamental limitation that all diffusion models may have—they are generally more computationally expensive than GANs.

### 5.3 DiffTime, Loss-DiffTime, and Guided-DiffTime

Sometimes synthetic time series are required to be statistically similar to historical time series but involve some specific constraints. For example, synthetic energy data are expected to follow the principle of energy conservation (Seo et al., 2021). Hence, Coletta et al. (2023) proposed DiffTime to handle the constrained time-series generation problem.

TimeDiff considers two types of constraints, the trend constraint as a time series $\boldsymbol{s} \in \mathbb{R}^{D\times T}$ and the fixed-point constraint on a specific variable at a certain time point. To incorporate the trend constraint,

TimeDiff simply replaces the condition in the conditional DDPM objective function with $\boldsymbol{s}$ as

$$\mathbb{E}_{k,\boldsymbol{x}^0_{1:T},\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{\tilde{\alpha}_k}\boldsymbol{x}^0_{1:T}+\sqrt{1-\tilde{\alpha}_k}\boldsymbol{\epsilon},k,\boldsymbol{s}\right)\right\|^2\right]. \tag{60}$$

For the fixed-point constraint, TimeDiff explicitly enforces the fixed values in the disturbed time series $\boldsymbol{x}^k_{1:T}$ at each diffusion and reverse step $k$.

To enhance the role of constraints, Loss-DiffTime was proposed to penalize the generative process in proportion to how much the synthetic data violate the constraints. Specifically, Loss-DiffTime uses a differentiable penalty function

$$f_{\mathrm{c}}=\lambda_g\mathrm{ReLU}(g(\hat{\boldsymbol{x}}^0_{1:T}))+\lambda_h\mathrm{ReLU}(h(\hat{\boldsymbol{x}}^0_{1:T})), \tag{61}$$

where $g(\cdot)$ and $h(\cdot)$ are inequality and equality constraints on the synthetic time series $\hat{\boldsymbol{x}}^0_{1:T}$ with $\lambda_g$ and $\lambda_h$ being the respective penalty terms, respectively. This penalty function is added to the objective function of Loss-DiffTime as

$$\mathbb{E}_{k,\boldsymbol{x}^0_{1:T},\boldsymbol{\epsilon}}\left[\delta(k)\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^k_{1:T},k,\boldsymbol{s}\right)+\rho f_{\mathrm{c}}(\hat{\boldsymbol{x}}^0_{1:T})\right\|^2\right], \tag{62}$$

where $\hat{\boldsymbol{x}}^0_{1:T}=\frac{1}{\sqrt{\alpha_k}}\left(\boldsymbol{x}^k_{1:T}-\frac{1-\alpha_k}{\sqrt{1-\tilde{\alpha}_k}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\boldsymbol{x}^k_{1:T},k,\boldsymbol{s}\right)\right)$.

Loss-DiffTime suffers from two major issues. First, as a drawback of almost all diffusion models, the sampling process requires $K$ iterations, which makes it slow and computationally expensive. Second, once the model is trained, the constraints are fixed. So, if one needs to change the constraints, the model should be trained with a new penalty function again. To solve these problems, Coletta et al. (2023) proposed Guided-DiffTime, which adopts the denoising diffusion implicit model (DDIM) as the underlying structure. DDIM shares the same training objective as DDPMs, but it requires fewer backward steps to produce high-quality samples, efficiently accelerating the sampling process (Song JM et al., 2021). Then, to solve the second problem, Guided-TimeDiff applies the penalty function only in the sampling process with the information given by its gradient with respect to $\boldsymbol{x}^k_{1:T}$ at each sampling step $k$ (Dhariwal and Nichol, 2021). Specifically, in each sampling iteration $k$, after the noise is obtained from the noise-matching network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$, it will be updated as follows:

$$\hat{\boldsymbol{\epsilon}}\leftarrow\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}^k_{1:T},k,\boldsymbol{s}), \tag{63}$$

$$\hat{\boldsymbol{\epsilon}}\leftarrow\hat{\boldsymbol{\epsilon}}-\rho\nabla_{\boldsymbol{x}^k_{1:T}}f_{\mathrm{c}}(\hat{\boldsymbol{x}}^0_{1:T}). \tag{64}$$

Finally, the updated noise will be used to compute the time series at the former diffusion step $k-1$.

## 6 Model comparison

In Table 2, we provide a comparison of all 15 methods mentioned in this review. We compare the time-series tasks that each model can conduct. Although most models were designed specifically for one of the three tasks, some of them can be applied to multiple tasks by nature. Specifically, models for

**Table 2 Comparison of diffusion models for time-series applications**

| Method | Forecasting | Imputation | Generation | Data | Diffusion | Sampling |
|---|---|---|---|---|---|---|
| TimeGrad | ✓ | | | MTS/STG | Discrete | Autoregressive |
| ScoreGrad | ✓ | | | MTS/STG | Continuous | Autoregressive |
| $D^3$VAE | ✓ | | | MTS/STG | Discrete | One-shot |
| DSPD | ✓ | ✓ | | MTS/STG | Discrete | One-shot |
| CSPD | ✓ | ✓ | | MTS/STG | Continuous | One-shot |
| TimeDiff | ✓ | | | MTS/STG | Discrete | One-shot |
| DiffSTG | ✓ | | | STG | Discrete | One-shot |
| GCRDD | ✓ | | | STG | Discrete | Autoregressive |
| CSDI | ✓ | ✓ | | MTS/STG | Discrete | One-shot |
| SSSD | ✓ | ✓ | | MTS/STG | Discrete | One-shot |
| PriSTI | ✓ | ✓ | | STG | Discrete | One-shot |
| TSGM | | | ✓ | MTS/STG | Continuous | Autoregressive |
| DiffTime | | | ✓ | MTS/STG | Discrete | One-shot |
| Loss-DiffTime | | | ✓ | MTS/STG | Discrete | One-shot |
| Guided-DiffTime | | | ✓ | MTS/STG | Discrete | One-shot |

imputation can be applied to forecasting tasks by regarding the future time series as missing values. In addition, we compare the data types that each model can handle. Generally, models designed for multivariate time series can be extended to STG tasks by simply discarding the graph structure. However, practically, the performance of these models on STG datasets may not be satisfactory. We also compare the diffusion process of different models, categorizing them into discrete diffusion and continuous diffusion. Most models with discrete diffusion are based on the DDPM formulation (Ho et al., 2020), while models with continuous diffusion follow the SDE-based design (Yang S et al., 2021). Finally, we consider the distinctions in the sampling process. Specifically, we categorize the models as either autoregressive generators or one-shot generators. With the autoregressive sampling process, models produce predictions for one time point in one sampling iteration. By contrast, one-shot models generate full time series for the whole target time period in one run. Normally, autoregressive methods are more accurate, but one-shot generation is more computationally efficient because it requires only one sampling iteration (Wen et al., 2023).

## 7 Practical applications

To highlight the practical value of diffusion models for time series, we provide a brief review of relevant real-world applications. In general, diffusion models can be applied to almost all kinds of time-series-based problems, because most are designed to handle broad time-series forecasting, imputation, and generation tasks. In this section, we delve into some primary application domains, and our discussion is divided into two subsections depending on the data type. Specifically, we focus on general time-series and STG applications. Although there can be an overlap between the applications of these two data types because STGs are special cases of multivariate time series, studies in these two areas usually have different emphases.

### 7.1 General time-series applications

The study of diffusion models for time series involves data from numerous real-world application domains. Because diffusion models are highly flexible and can generate data from any distribution, they can be used to generate time-series data with complicated patterns and dependencies. Specifically, when the data are subject to high uncertainty by nature, diffusion models can generate probabilistic outputs to better represent the uncertainty, leading to better insights and decisions (Capel and Dumas, 2023). Here we will present two frequently discussed examples from the literature, including energy forecasting and medical practices.

#### 7.1.1 Energy forecasting

Energy forecasting is of essential importance for decision-making in environment control, energy management, and power distribution. For example, renewable energy forecasting helps enhance the operational predictability of modern power systems, contributing to the control of energy mix for consumption, and eventually, leading to better control of carbon emissions and climate change. However, renewable energies such as solar energy and wind energy are affected by external factors such as unexpected natural conditions, resulting in unstable production and consumption. To capture the uncertainty in renewable energy generation, Capel and Dumas (2023) proposed to use DDPMs for renewable energy forecasting and showed the superiority of DDPMs in terms of accuracy over other generative methods. As another example, Wang et al. (2023) proposed DiffLoad, a diffusion-based Seq2Seq structure, for electricity load forecasting that will ultimately contribute to the planning and operation of power systems.

#### 7.1.2 Medical practices

To meet increasing requirements, analysis of physiological signals has become crucial (Zhang YF et al., 2021). In clinical settings, an accurate predictive model is important for detecting potential unusual signals. The transform-based diffusion probabilistic model for sparse time-series forecasting (TDSTF), a transformer-based diffusion model, was proposed for heart rate and blood pressure forecasting in the intensive care unit (ICU) and shown as an effective and efficient solution compared to other models in the field (Chang et al., 2023). In addition, diffusion models can be designed for electroencephalogram (EEG) and electrocardiogram (ECG) signal prediction, imputation, and generation,

offering valuable support for clinical treatments (Neifar et al., 2023; Shu et al., 2023).

### 7.2 Spatial temporal graph applications

STGs are a special type of multivariate time-series data that record the change of values of one or multiple variables of different entities in a system over time. The literature on diffusion models for STGs covers two application domains, transportation (Li RK et al., 2023; Wen et al., 2023) and air quality monitoring (Liu MZ et al., 2023; Wen et al., 2023).

#### 7.2.1 Transportation

For transportation, the system of interest is usually traffic networks with numerous sensors that monitor traffic conditions and collect traffic data such as vehicle speed and traffic flow (Xu DW et al., 2017; Seng et al., 2021). In traffic STGs, graph nodes are traffic sensors, and edges are usually constructed based on the geometric distance between each pair of sensors. Given the record of historical traffic data, the objective of traffic forecasting is to predict the average traffic speed or traffic flow in the next time period (e.g., an hour) for all sensors. Existing diffusion models serving this purpose include DiffSTG (Wen et al., 2023) and GCRDD (Li RK et al., 2023). Diffusion models can also be used for traffic data imputation (Liu MZ et al., 2023). Due to sensor failures or human errors, missing records are common in traffic data, which causes issues in analysis and prediction (Yi et al., 2016); thus, diffusion models can be applied to predict the missing values before forwarding the data for further analysis.

#### 7.2.2 Air quality monitoring

Diffusion models have also been applied to address air quality problems, including air quality prediction and air data imputation. In such tasks, air monitoring stations are modeled as graph nodes, and the edges are also formed based on geometric information. The objectives of air quality prediction and air data imputation are analogous to traffic forecasting and imputation. For example, $PM_{2.5}$ prediction, one of the famous tasks in this application domain, aims to predict future $PM_{2.5}$ readings of several air monitoring stations in a city (Yi et al., 2018; Wen et al., 2023).

### 7.3 Publicly available datasets

In Table 3, we list some frequently used publicly available datasets in the study of diffusion models for time series. We present datasets from papers focusing on either multivariate time series or STGs. We also categorize the datasets by their application domains and provide their source links. With high data quality, public availability, and close reflection of real-world implications, these datasets can serve as valuable resources for researchers and practitioners to test and understand diffusion models in various application domains.

## 8 Future research directions

Up to this point, we have discussed different methods for time-series forecasting, imputation, generation, and some relevant practical application domains. Now, we look into the limitations of existing models and challenges in this research area that may lead to future research directions and inspire future researchers.

1. Heavy model selection

How to select the optimal diffusion model structure still remains an open problem. Designing a diffusion model for time series requires the selection of many building blocks, such as the condition, the noise-matching network or other networks serving similar purposes, and the sampling acceleration technique. However, there are many potential choices for these building blocks without a unified framework or criterion for evaluation. For example, to construct a good condition for the target distribution, how to represent a time series effectively is quite essential (Yin et al., 2015). While TimeGrad (Kashif et al., 2021) and GCRDD (Li RK et al., 2023) choose RNN to encode historical information as the condition, other serial data encoders such as the transformer encoder (Vaswani et al., 2017) can be adopted to boost good performance as well (Sikder et al., 2023). For each building block, the choice is highly flexible but suffers from heavy tuning costs of structures and hyperparameters. This highlights the importance of research on the comparison of different options and a unified framework or guidance on model selection.

2. Insufficient research on noise injection

Noise injection, as an important component of diffusion models, has not been sufficiently studied in the research for time series. First, current research

**Table 3 Publicly available datasets of multivariate time series (MTS) and spatio-temporal graphs (STG)**

| Type | Domain | Dataset | Source |
|---|---|---|---|
| MTS | Energy | SOLAR | https://github.com/laiguokun/multivariate-time-series-data |
| | | ELECTRICITY | https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 |
| | | NORPOOL | https://www.nordpoolgroup.com/Market-data1/Power-system-data |
| | | CAISO | http://www.energyonline.com/Data |
| | | ETTm1 | https://github.com/zhouhaoyi/ETDataset |
| | | ETTh1 | https://github.com/zhouhaoyi/ETDataset |
| | Traffic | TRAFFIC | http://pems.dot.ca.gov |
| | | TAXI | https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page |
| | Climate | WEATHER | https://www.bgc-jena.mpg.de/wetter/ |
| | Finance | EXCHANGE | https://github.com/laiguokun/multivariate-time-series-data |
| | Website | WIKIPEDIA | https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets |
| STG | Traffic | METR_LA | https://github.com/liyaguang/DCRNN |
| | | PEMS_BAY | https://github.com/liyaguang/DCRNN |
| | | PEMS_08 | https://github.com/guoshnBJTU/ASTGNN/tree/main/data |
| | Air | AIR_BJ | https://www.microsoft.com/en-us/research/uploads/prod/2016/02/Data-1.zip |
| | | AIR_GZ | https://www.microsoft.com/en-us/research/uploads/prod/2016/02/Data-1.zip |
| | | AQI_36 | https://www.microsoft.com/en-us/research/uploads/prod/2016/02/Data-1.zip |

considers only injecting Gaussian noise, leaving lack of study on other distributions and noises designed specifically for discrete time-series data. In addition, the noise schedule (i.e., how $\alpha_k$ changes with the diffusion steps) is not carefully designed for time-series data. Currently, we can observe research only on the noise schedule for images (Chen, 2023).

3. Unsatisfactory performance with spatio-temporal graphs

Although there are several studies on diffusion models for STG forecasting, the performance of these models is still unsatisfactory. Specifically, diffusion models fail to outperform the deterministic methods such as DCRNN (Li YG et al., 2018) and GMSDR (Liu DC et al., 2022). To enhance performance, one may consider how to better encode the spatial information in diffusion models. Currently, the graph structure $\boldsymbol{W}$ is used only as part of the condition, so how to enhance the role of spatial information is still a problem to be solved. Moreover, exiting diffusion models for STGs are very slow at sampling (Li RK et al., 2023). This shows the need for designing or incorporating suitable sampling acceleration techniques.

## 9 Conclusions

Diffusion models, a rising star in advanced generative techniques, have shown their exceptional power in various real-world applications. In recent years, many successful attempts have been made to incorporate diffusion in time-series applications to boost model performance. As compensation for the lack of a methodical summary and discourse on diffusion-based approaches for time series, we have furnished a self-contained survey of these approaches while discussing the interactions and differences among them. Specifically, we have presented seven models for time-series forecasting, four models for time-series imputation, and four models for time-series generation. Although these models have shown good performance with empirical evidence, we feel obligated to emphasize that they are usually associated with very high computational costs. In addition, because most models are constructed with a high level of theoretical background, there is still a lack of deeper discussion and exploration of the rationale behind these models. This survey is expected to serve as a starting point for new researchers in this area and inspiration for future directions.

## Contributors

Junbin GAO initialized the idea. Lequan LIN collected all relevant literature for review and created all the figures and tables. Lequan LIN and Zhengkun LI drafted the paper. Ruikun LI and Xuliang LI helped organize the paper. All authors revised and finalized the paper.

## Compliance with ethics guidelines

Junbin GAO is a guest editor of this special feature, and he was not involved with the peer review process of this paper. Lequan LIN, Zhengkun LI, Ruikun LI, Xuliang LI, and Junbin GAO declare that they have no conflict of interest.

## References


Alcaraz JML, Strodthoff N, 2023. Diffusion-based time series imputation and forecasting with structured state space models. *Trans Mach Learn Res*.

Anderson BDO, 1982. Reverse-time diffusion equation models. *Stoch Process Their Appl*, 12(3):313-326. https://doi.org/10.1016/0304-4149(82)90051-5

Austin J, Johnson DD, Ho J, et al., 2021. Structured denoising diffusion models in discrete state-spaces. Proc 35th Conf on Neural Information Processing Systems, p.17981-17993.

Biloš M, Rasul K, Schneider A, et al., 2023. Modeling temporal data as continuous functions with stochastic process diffusion. Proc 40th Int Conf on Machine Learning, p.2452-2470.

Cao W, Wang D, Li J, et al., 2018. BRITS: bidirectional recurrent imputation for time series. Proc 32nd Conf on Neural Information Processing Systems, p.6776-6786. https://doi.org/10.5555/3327757.3327783

Capel EH, Dumas J, 2023. Denoising diffusion probabilistic models for probabilistic energy forecasting. https://doi.org/10.48550/arXiv.2212.02977

Chang P, Li HY, Quan SF, et al., 2023. TDSTF: transformer-based diffusion probabilistic model for sparse time series forecasting. https://doi.org/10.48550/arXiv.2301.06625

Che ZP, Purushotham S, Cho K, et al., 2018. Recurrent neural networks for multivariate time series with missing values. *Sci Rep*, 8(1):6085. https://doi.org/10.1038/s41598-018-24271-9

Chen T, 2023. On the importance of noise scheduling for diffusion models. https://doi.org/10.48550/arXiv.2301.10972

Choi J, Choi H, Hwang J, et al., 2022. Graph neural controlled differential equations for traffic forecasting. Proc 36th AAAI Conf on Artificial Intelligence, p.6367-6374. https://doi.org/10.1609/aaai.v36i6.20587

Chu C, Minami K, Fukumizu K, 2020. Smoothness and stability in GANs. Proc 8th Int Conf on Learning Representations.

Chung J, Gulcehre C, Bengio K, et al., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. https://doi.org/10.48550/arXiv.1412.3555

Coletta A, Gopalakrishan S, Borrajo D, et al., 2023. On the constrained time-series generation problem. https://doi.org/10.48550/arXiv.2307.01717

Croitoru FA, Hondru V, Ionescu RT, et al., 2023. Diffusion models in vision: a survey. *IEEE Trans Patt Anal Mach Intell*, 45(9):10850-10869. https://doi.org/10.1109/TPAMI.2023.3261988

Desai A, Freeman C, Wang ZH, et al., 2021. TimeVAE: a variational auto-encoder for multivariate time series generation. https://doi.org/10.48550/arXiv.2111.08095

Dhariwal P, Nichol A, 2021. Diffusion models beat GANs on image synthesis. Proc 35th Conf on Neural Information Processing Systems, p.8780-8794.

Donahue C, McAuley JJ, Puckette MS, 2019. Adversarial audio synthesis. Proc 7th Int Conf on Learning Representations.

Esteban C, Hyland SL, Rätsch G, 2017. Real-valued (medical) time series generation with recurrent conditional GANs. https://doi.org/10.48550/arXiv.1706.02633

Fortuin V, Baranchuk D, Rätsch G, et al., 2020. GP-VAE: deep probabilistic time series imputation. Proc 23rd Int Conf on artificial intelligence and statistics, p.1651-1661.

Goel K, Gu A, Donahue C, et al., 2022. It's raw! Audio generation with state-space models. Proc 39th Int Conf on Machine Learning, p.7616-7633.

Gu A, Goel K, Ré C, 2022. Efficiently modeling long sequences with structured state spaces. Proc 10th Int Conf on Learning Representations.

Harvey W, Naderiparizi S, Masrani V, et al., 2022. Flexible diffusion modeling of long videos. Proc 36th Conf on Neural Information Processing Systems, p.27953-27965.

Ho J, Jain A, Abbeel P, 2020. Denoising diffusion probabilistic models. Proc 34th Int Conf on Neural Information Processing Systems, p.6840-6851.

Ho J, Saharia C, Chan W, et al., 2022a. Cascaded diffusion models for high fidelity image generation. *J Mach Learn Res*, 23(1):47. https://doi.org/10.5555/3586589.3586636

Ho J, Salimans T, Gritsenko A, et al., 2022b. Video diffusion models. Proc 36th Conf on Neural Information Processing Systems, p.8633-8646.

Hochreiter S, Schmidhuber J, 1997. Long short-term memory. *Neur Comput*, 9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hyvärinen A, 2005. Estimation of non-normalized statistical models by score matching. *J Mach Learn Res*, 6(24):695-709.

Kashif R, Abdul-Saboor S, Ingmar S, et al., 2021. Multivariate probabilistic time series forecasting via conditioned normalizing flows. https://doi.org/10.48550/arXiv.2002.06103

Kingma DP, Welling M, 2013. Auto-encoding variational Bayes. https://doi.org/10.48550/arXiv.1312.6114

Kingma DP, Salimans T, Jozefowicz R, et al., 2016. Improved variational inference with inverse autoregressive flow. Proc 30th Conf on Neural Information Processing Systems, p.29.

Kong ZF, Ping W, Huang JJ, et al., 2021. DiffWave: a versatile diffusion model for audio synthesis. Proc 9th Int Conf on Learning Representations.

Li RK, Li XL, Gao SY, et al., 2023. Graph convolution recurrent denoising diffusion model for multivariate probabilistic temporal forecasting. Proc 19th Int Conf on Advanced Data Mining and Applications.

Li XL, Thickstun J, Gulrajani I, et al., 2022. Diffusion-LM improves controllable text generation. Proc 36th Conf on Neural Information Processing Systems, p.4328-4343.

Li Y, Lu XJ, Wang YQ, et al., 2022. Generative time series forecasting with diffusion, denoise, and disentanglement. Proc 36th Conf on Neural Information Processing System, p.23009-23022.

Li YG, Yu R, Shahabi C, et al., 2018. Diffusion convolutional recurrent neural network: data-driven traffic forecasting. Proc 6th Int Conf on Learning Representations.

Li YN, Chen ZZ, Zha DC, et al., 2021. Learning disentangled representations for time series. https://doi.org/10.48550/arXiv.2105.08179

Lim H, Kim M, Park S, et al., 2023. Regular time-series generation using SGM. https://doi.org/10.48550/arXiv.2301.08518

Liu DC, Wang J, Shang S, et al., 2022. MSDR: multi-step dependency relation networks for spatial temporal forecasting. Proc 28th ACM SIGKDD Conf on Knowledge Discovery and Data Mining, p.1042-1050. https://doi.org/10.1145/3534678.3539397

Liu MZ, Huang H, Feng H, et al., 2023. PriSTI: a conditional diffusion framework for spatiotemporal imputation. Proc IEEE 39th Int Conf on Data Engineering, p.1927-1939. https://doi.org/10.1109/ICDE55515.2023.00150

Lugmayr A, Danelljan M, Romero A, et al., 2022. RePaint: inpainting using denoising diffusion probabilistic models. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.11451-11461. https://doi.org/10.1109/CVPR52688.2022.01117

Luo C, 2022. Understanding diffusion models: a unified perspective. https://doi.org/10.48550/arXiv.2208.11970

Luo YH, Cai XR, Zhang Y, et al., 2018. Multivariate time series imputation with generative adversarial networks. Proc 32nd Conf on Neural Information Processing Systems, p.1603-1614. https://doi.org/10.5555/3326943.3327090

Mogren O, 2016. C-RNN-GAN: continuous recurrent neural networks with adversarial training. https://doi.org/10.48550/arXiv.1611.09904

Mulyadi AW, Jun E, Suk HI, 2022. Uncertainty-aware variational-recurrent imputation network for clinical time series. *IEEE Trans Cybern*, 52(9):9684-9694. https://doi.org/10.1109/TCYB.2021.3053599

Neifar N, Ben-Hamadou A, Mdhaffar A, et al., 2023. Diff-ECG: a generalized probabilistic diffusion model for ECG signals synthesis. https://doi.org/10.48550/arXiv.2306.01875

Nikolay S, Junyoung C, Mikolaj B, et al., 2022. Step-unrolled denoising autoencoders for text generation. Int Conf on Learning Representations.

Osman MS, Abu-Mahfouz AM, Page PR, 2018. A survey on data imputation techniques: water distribution system as a use case. *IEEE Access*, 6:63279-63291. https://doi.org/10.1109/ACCESS.2018.2877269

Rasul K, Seward C, Schuster I, et al., 2021. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. Proc 38th Int Conf on Machine Learning, p.8857-8868.

Ronneberger O, Fischer P, Brox T, 2015. U-Net: convolutional networks for biomedical image segmentation. Proc 18th Int Conf on Medical Image Computing and Computer-Assisted Intervention, p.234-241. https://doi.org/10.1007/978-3-319-24574-4_28

Saremi S, Hyvärinen A, 2019. Neural empirical Bayes. *J Mach Learn Res*, 20(181):1-23.

Seng DW, Lv FS, Liang ZY, et al., 2021. Forecasting traffic flows in irregular regions with multi-graph convolutional network and gated recurrent unit. *Front Inform Technol Electron Eng*, 22(9):1179-1193. https://doi.org/10.1631/FITEE.2000243

Seo S, Arık SÖ, Yoon J, et al., 2021. Controlling neural networks with rule representations. Proc 35th Conf on Neural Information Processing Systems, p.11196-11207.

Shen LF, Kwok J, 2023. Non-autoregressive conditional diffusion models for time series prediction. Proc 40th Int Conf on Machine Learning, p.31016-31029.

Shu K, Zhao YC, Wu L, et al., 2023. Data augmentation for seizure prediction with generative diffusion model. https://doi.org/10.48550/arXiv.2306.08256

Sikder MF, Ramachandranpillai R, Heintz F, 2023. Transfusion: generating long, high fidelity time series using diffusion models with transformers. https://doi.org/10.48550/arXiv.2307.12667

Simeunović J, Schubnel B, Alet PJ, et al., 2022. Spatio-temporal graph neural networks for multi-site PV power forecasting. *IEEE Trans Sustain Energy*, 13(2):1210-1220. https://doi.org/10.1109/TSTE.2021.3125200

Sohl-Dickstein J, Weiss E, Maheswaranathan N, et al., 2015. Deep unsupervised learning using nonequilibrium thermodynamics. Proc 32nd Int Conf on Machine Learning, p.2256-2265.

Sønderby CK, Raiko T, Maaløe L, et al., 2016. Ladder variational autoencoders. Proc 30th Int Conf on Neural Information Processing Systems, p.3745-3753. https://doi.org/10.5555/3157382.3157516

Song JM, Meng CL, Ermon S, 2021. Denoising diffusion implicit models. Proc 9th Int Conf on Learning Representations.

Song Y, Ermon S, 2019. Generative modeling by estimating gradients of the data distribution. Proc 33rd Int Conf on Neural Information Processing Systems, p.11918-11930.

Song Y, Garg S, Shi JX, et al., 2020. Sliced score matching: a scalable approach to density and score estimation. Proc 35th Uncertainty in Artificial Intelligence Conf, p.574-584.

Tashiro Y, Song JM, Song Y, et al., 2021. CSDI: conditional score-based diffusion models for probabilistic time series imputation. Proc 35th Conf on Neural Information Processing Systems, p.24804-24816.

Vahdat A, Kautz J, 2020. NVAE: a deep hierarchical variational autoencoder. Proc 34th Int Conf on Neural Information Processing Systems, p.19667-19679. https://doi.org/10.5555/3495724.3497374

van den Oord A, Dieleman S, Zen HG, et al., 2016. WaveNet: a generative model for raw audio. 9th ISCA Speech Synthesis Workshop, p.135.

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010. https://doi.org/10.5555/3295222.3295349

Vincent P, 2011. A connection between score matching and denoising autoencoders. *Neur Comput*, 23(7):1661-1674. https://doi.org/10.1162/NECO_a_00142

Wang ZX, Wen QS, Zhang CL, et al., 2023. DiffLoad: uncertainty quantification in load forecasting with diffusion model. https://doi.org/10.48550/arXiv.2306.01001

Wen HM, Lin YF, Xia YT, et al., 2023. DiffSTG: probabilistic spatio-temporal graph forecasting with denoising diffusion models. https://doi.org/10.48550/arXiv.2301.13629

Wu HX, Xu JH, Wang JM, et al., 2021. Autoformer: decomposition transformers with auto-correlation for long-term series forecasting. Proc 35th Conf on Neural Information Processing Systems, p.22419-22430.

Xiao ZS, Kreis K, Vahdat A, 2022. Tackling the generative learning trilemma with denoising diffusion GANs. Proc 10th Int Conf on Learning Representations.

Xu DW, Wang YD, Jia LM, et al., 2017. Real-time road traffic state prediction based on ARIMA and Kalman filter. *Front Inform Technol Electron Eng*, 18(2):287-302. https://doi.org/10.1631/FITEE.1500381

Xu TL, Wenliang LK, Munn M, et al., 2020. Cot-GAN: generating sequential data via causal optimal transport. Proc 34th Conf on Neural Information Processing Systems, p.8798-8809.

Yan TJ, Zhang HW, Zhou T, et al., 2021. ScoreGrad: multivariate probabilistic time series forecasting with continuous energy-based generative models. https://doi.org/10.48550/arXiv.2106.10121

Yang L, Zhang ZL, Song Y, et al., 2023. Diffusion models: a comprehensive survey of methods and applications. https://doi.org/10.48550/arXiv.2209.00796

Yang RH, Srivastava P, Mandt S, 2022. Diffusion probabilistic modeling for video generation. https://doi.org/10.48550/arXiv.2203.09481

Yang S, Sohl-Dickstein J, Kingma DP, et al., 2021. Score-based generative modeling through stochastic differential equations. Proc 9th Int Conf on Learning Representations.

Yi XW, Zheng Y, Zhang JB, et al., 2016. ST-MVL: filling missing values in geo-sensory time series data. Proc 25th Int Joint Conf on Artificial Intelligence, p.2704-2710. https://doi.org/10.5555/3060832.3060999

Yi XW, Zhang JB, Wang ZY, et al., 2018. Deep distributed fusion network for air quality prediction. Proc 24th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining, p.965-973. https://doi.org/10.1145/3219819.3219822

Yin H, Yang SQ, Zhu XQ, et al., 2015. Symbolic representation based on trend features for knowledge discovery in long time series. *Front Inform Technol Electron Eng*, 16(9):744-758. https://doi.org/10.1631/FITEE.1400376

Yoon J, Zame WR, van der Schaar M, 2019. Estimating missing data in temporal data streams using multi-directional recurrent neural networks. *IEEE Trans Biomed Eng*, 66(5):1477-1490. https://doi.org/10.1109/TBME.2018.2874712

Yu PY, Xie SR, Ma XJ, et al., 2022. Latent diffusion energy-based model for interpretable text modelling. Proc 39th Int Conf on Machine Learning, p.25702-25720.

Zhang HY, Cissé M, Dauphin YN, et al., 2018. Mixup: beyond empirical risk minimization. Proc 6th Int Conf on Learning Representations.

Zhang YF, Zhao ZD, Deng YJ, et al., 2021. ECGID: a human identification method based on adaptive particle swarm optimization and the bidirectional LSTM model. *Front Inform Technol Electron Eng*, 22(12):1641-1654. https://doi.org/10.1631/FITEE.2000511