The Rise of Diffusion Models in Time-Series Forecasting

Caspar Meijer
EEMCS Distributed Systems
Delft University of Technology
c.j.meijer-1@student.tudelft.nl

Lydia Y. Chen
EEMCS Distributed Systems
Delft University of Technology
y.chen-10@tudelft.nl

18 December 2023

Abstract

This survey delves into the application of diffusion models in time-series forecasting. Diffusion models are demonstrating state-of-the-art results in various fields of generative AI. The paper includes comprehensive background information on diffusion models, detailing their conditioning methods and reviewing their use in time-series forecasting. The analysis covers 11 specific time-series implementations, the intuition and theory behind them, the effectiveness on different datasets, and a comparison among each other. Key contributions of this work are the thorough exploration of diffusion models' applications in time-series forecasting and a chronologically ordered overview of these models. Additionally, the paper offers an insightful discussion on the current state-of-the-art in this domain and outlines potential future research directions. This serves as a valuable resource for researchers in AI and time-series analysis, offering a clear view of the latest advancements and future potential of diffusion models.

1 Introduction

The advent of generative artificial intelligence (AI) has been transformative across various domains, ranging from education [1]–[3] to workplaces [4], [5] and daily activities [6]. Central to this transformation is deep learning, a key pillar enabling AI to analyze and synthesize complex data patterns. Initially, generative AI is defined by its ability to create new, original data samples that reflect the statistical characteristics of a specified dataset, represented mathematically as: given a sample x from distribution $q(x)$, the generative model produces outputs $\hat{x}$ that appear to be drawn from $q(x)$ [7].

Following this introductory note on generative AI, the focus shifts to time-series forecasting, an area of critical importance across various industries. Time-series data, characterized by their temporal dependencies and multifaceted interactions, present unique challenges and opportunities for forecasting future events based

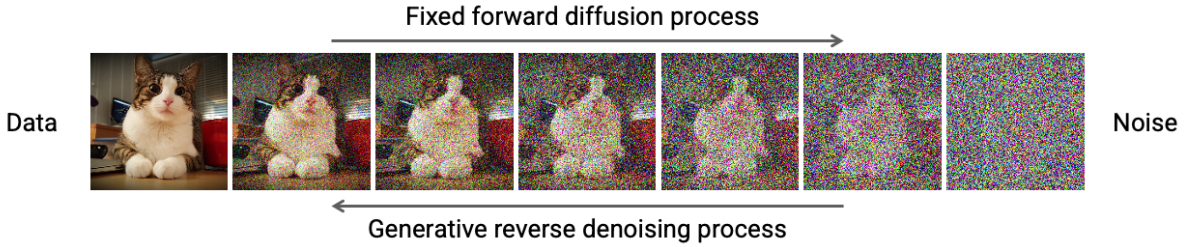

Figure 1: Denoising Diffusion Process

on historical data. This is particularly significant in fields such as healthcare prediction [8]–[13], energy management [14]–[16], and traffic control [17], [18].

In the landscape of time-series forecasting, the solution space has evolved considerably over time. Initial advancements were marked by the introduction of Long Short-Term Memory (LSTM) variants, notably, the Seq2Seq Autoencoder-LSTM [19]. However, a significant paradigm shift occurred in 2017 with the introduction of the Transformer structure, which incorporated attention mechanisms [20]. This innovation addressed the critical limitation of LSTMs, which was the loss of previous information over extended sequences [21]. Subsequent developments in Transformer-based models, [22]–[27], have furthered the field.

In the field of generative modeling, the introduction of modeling structures such as variational autoencoders (VAEs), normalizing flows (NFs), and generative adversarial networks (GANs) have marked significant advancements [28], [29]. Yet, the emergence of diffusion models signals a revolutionary period, promising superior-quality outputs that are pushing the state-of-the-art [30]–[32]. A diffusion model can be characterized by the fact that they simulate, as the name implies, a diffusion process transforming data into white noise and then reversing it back into data as shown in Figure 1. These models, capable of approximating the original data distribution, have shown exceeding results in various domains, including image [30], [33]–[35], text [36]–[38], speech [39], [40] and video synthesis [41]–[44].

In this paper, we narrow our focus to the application of diffusion models in time-series forecasting. These models, distinguished by their profound ability to comprehend intricate data dynamics, are revolutionizing this field [45], [46]. First time-series forecasting is further formalized together with how to evaluate the models in sub-section 1.1. Afterwards, the intrinsic workings of the diffusion model and how to condition it is explained in section 2. Then the diffusion-based time-series forecasting papers are discussed in section 3. Finally, a comprehensive discussion and future works is given in sections 4 and 5 respectively.

The main contributions of this paper are:

- In-depth preliminary section about diffusion models and the methods of conditioning that are used for time-series modeling.
- A chronologically ordered overview of the diffusion models capable of time-series forecasting, going in-depth on the implementation in relation to the preliminary, specify the results on which datasets, and a discussion about the work in relation to other diffusion models.
- The work serves as in depth overview for researchers seeking to acquaint themselves with the methods that make up the current state-of-the-art diffusion models for time-series forecasting.

1.1 Time-series

Time-series modeling, as highlighted by Lin *et al.* [45] and Koo and Kim [46], is a specialized form of conditional generative modeling, where segments of the time-series are used to generate other segments. This area encompasses three key types: generation, imputation, and forecasting. Generation is about creating synthetic time-series data; imputation deals with filling gaps in existing data, and forecasting is the prediction of future values. These types are interconnected, with forecasting being a specific form of imputation and both imputation and forecasting being aspects of generation. This section will delve into the problem definition for time-series forecasting and evaluation metrics to assess model performance.

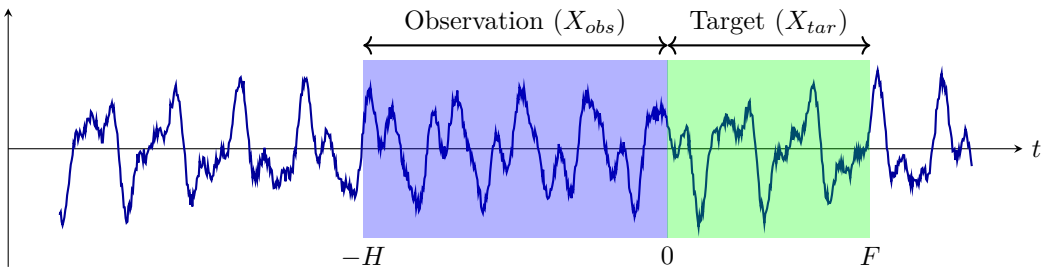

Figure 2: Example of univariate time-series forecasting situation.

1.1.1 Problem definition for Time-series Forecasting

Since time is inherently continuous, these series are represented through discrete time steps, determined by a chosen sampling frequency, which can range from milliseconds to hours. For simplicity we assume that the entire dataset X is discretized evenly in the rest of the paper if not explicitly specified. The historical time range, comprising H steps, is defined as the set of observations $obs = \{t \in \mathbb{Z} \mid -H < t \leq 0\}$, while the future range, spanning F steps, is denoted as the targets $tar = \{t \in \mathbb{Z} \mid 0 < t \leq F\}$. Consequently, historical data can be represented as $X_{obs} \in \mathbb{R}^{d \times H}$, and the target forecast data as $X_{tar} \in \mathbb{R}^{d \times F}$, where d indicates the number of distinct features. A time-series is classified as univariate when $d = 1$ as illustrated in Figure 2, and as multivariate when $d > 1$. In this context, a specific value at time step t and feature i from the time-series can be defined as $x_{i,t}$, where $i \in \{1 \dots d\}$ and $t \in \{-H, \dots, F\}$ or as x_t when talking about the entire value vector at the time step.

1.1.2 Evaluation Metrics

The evaluation of time-series forecasting can be done through a varaiaty of metrics. The most common ones are through the use of the Mean Squared Error (MSE), Mean Absolute Error (MAE), and the Continuous Ranked Probability Score (CRPS) [47]. The biggest difference between these metrics is that the MSE & MAE focus on the mean error, while the CRPS also takes into account the uncertainty of the prediction.

The MSE and MAE over a time-series forecasting are defined as:

$$\text{MSE}(\hat{X}_{tar}, X_{tar}) = \frac{1}{F}\sum_{t=0}^{F}(x_t - \hat{x}_t)^2 \qquad \text{MAE}(\hat{X}_{tar}, X_{tar}) = \frac{1}{F}\sum_{t=0}^{F}|x_t - \hat{x}_t| \tag{1}$$

The output of these error measurements are vectors for the multivariate time-series, at end the average over all features is taken as the final value. One must consider that the values ought to be normalized, as otherwise errors of different features can have very different scales.

The CRPS is a metric that is used for probabilistic forecasting and measures the compatibility of a cumulative distribution function (CDF) F with an observation X_{tar}. In other words, the CRPS metric becomes smaller if the distribution is highly concentrated on the prediction as illustrated in Figure 3. However, in many scenarios the CDF F is not available analytically, but only by estimating it through a set of N forecast samples $\hat{X}^N_{tar}$ [48]. These samples are gathered by sampling the probabilistic model N times. The CRPS metric is calculated for of a single feature at a single timestamp. It is possible to estimate the empirical CDF $\hat{F}_N(z)$ given a point z from these predictions. The empirical CDF at a point z is estimated by the proportion of forecast values that are less than or equal to z and is defined as

$$\hat{F}^N_{i,t}(z) = \frac{1}{N}\sum_{n=1}^{N}\mathbb{I}\{\hat{x}^n_{i,t} \leq z\}. \tag{2}$$

Where $\mathbb{I}\{\hat{X}^i_{i,t} \leq z\}$ is the indicator function that equals 1 if the condition $\hat{X}^i_{tar} \leq z$ is true, and 0 otherwise. The CRPS for the empirical CDF $\hat{F}_N$ and the observation X_{tar} is then calculated as:

$$\text{CRPS}(\hat{F}^N_{i,t}, x_{i,t}) = \int_{\mathbb{R}}\left(\hat{F}^N_{i,t}(z) - \mathbb{I}\{x_{i,t} \leq z\}\right)^2 dz \tag{3}$$

Where i indicates the feature and t the timestamp of the forecasts. The estimation of this formulation is further explained and optimized by Jordan *et al.* [48].

As previously mentioned, the CRPS focuses on a single feature at a specific timestamp. To transform this for a multivariate time-series forecast with multiple timestamps in the future some normalization and averaging ought to be done. This can be either done as the Normalized Average CRPS, which is formulated as such:

$$\text{NACRPS}(\hat{F}^N, X_{tar}) = \frac{\sum_{i,t}\text{CRPS}(\hat{F}^N_{i,t}, x_{i,t})}{\sum_{i,t}|x_{i,t}|} \tag{4}$$

And then there is the CRPS-sum, which is the CRPS for the distribution F for the sum of all d features:

$$\text{CRPS}_{\text{sum}}(\hat{F}^N, X_{tar}) = \frac{\sum_{t}\text{CRPS}(\hat{F}^N_{i,t}, \sum_i x_i)}{\sum_{i,t}|x_{i,t}|} \tag{5}$$

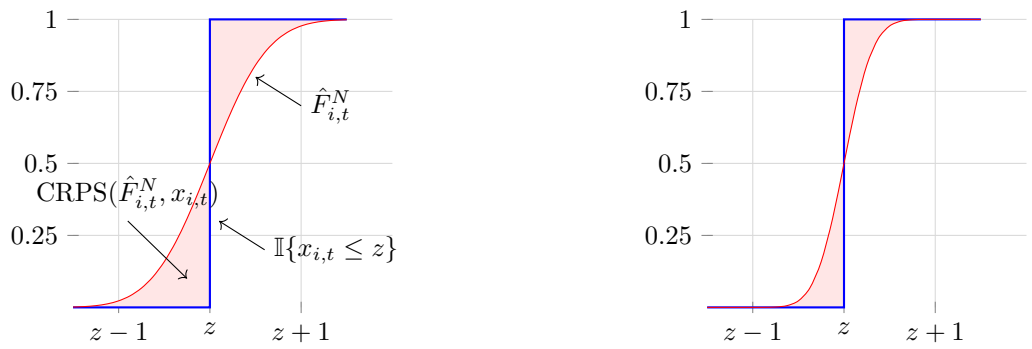

Figure 3: Comparative Visualization of Continuous Ranked Probability Scores: The left plot exhibits a higher CRPS value, indicated by the greater area between the forecast distribution (red line) and the step function (blue line), compared to the right plot.

2 Preliminary: Diffusion Models

The foundational works of diffusion models were established by Ho *et al.* [49] and Song *et al.* [50]. These foundations are the unconditional generation of data, which means to create data samples without relying on specific conditions, such as text prompts. Formally, unconditional generation can be described as viewing the training data x as a distribution $q(x)$ from which it is possible to extract samples described by $x \in \mathcal{R}^d$. Now the goal is to approximate this distribution as $p_\theta(x)$ and be able to sample new unseen data from this approximation [7].

Diffusion models approximate the distribution by learning how data can be recovered after it has been diffused to pure noise. The model attempts to transform a Gaussian distribution back into the data distribution as illustrated in Figure 4. This process enables the model to generate data samples from noise, noise is turned into data that resembles the training dataset.

Where Ho *et al.* [49] describes the diffusion and denoising processes as discrete steps and Song *et al.* [50] generalizes these processes to continuous time with the use of stochastic differential equations (SDE). The discrete implementation is described in section 2.1 and the continuous version in section 2.2. However, unconditional generation is not useful if specific data samples are desired. For instance, generating images that contain specifics that are described through text prompts [34]. Such conditional generation of data is further explained in section 2.3

2.1 Denoising Diffusion Probabilistic Model (DDPM) (2020) [49]

Core to these models are two pivotal processes: the forward diffusion and the reverse denoising process.

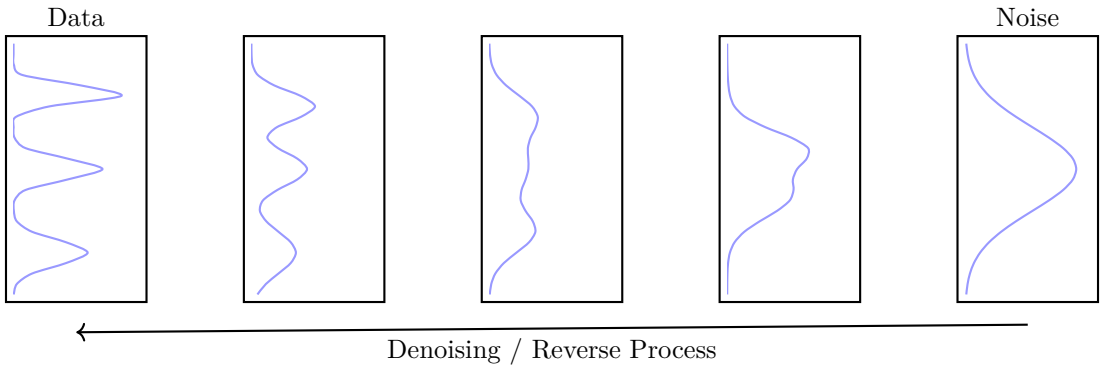

Figure 4: Probability Distribution Evolution: From Noise to Data

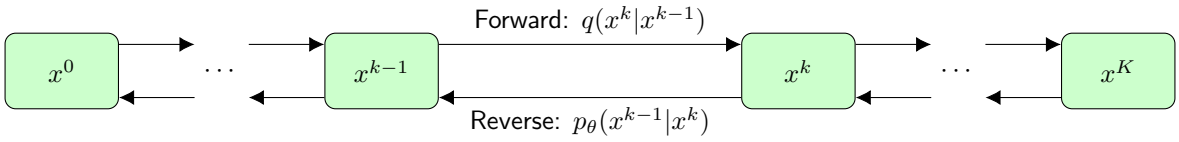

Figure 5: Illustration of the Denoising Diffusion Probabilistic Model process

The forward process transforms an input x^0 into a noise vector x^K through a Markov Chain, progressively introducing noise over K steps as can be seen in Figure 5. The transformation is described as:

$$q(x^{1:K}|x^0) = \prod_{k=1}^{K} q(x^k \mid x^{k-1}) \text{ where } q(x^k|x^{k-1}) = \mathcal{N}(x^k; \sqrt{1-\beta_k}x^{k-1}, \beta_k\mathbf{I}) \tag{6}$$

where $\beta_k \in [0,1]$ dictates the variance of the noise introduced at each stage. Obtaining x^k is achieved through:

$$q(x^k|x^0) = \mathcal{N}(x^k; \sqrt{\alpha_k}x^0, (1-\alpha_k)\mathbf{I}) \tag{7}$$

where $\alpha_k = \prod_{i=1}^{k}(1-\beta_i)$ and can be also represented in a closed form as:

$$x^k(x^0, \epsilon) = \sqrt{\alpha_k}x^0 + \sqrt{1-\alpha_k}\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{8}$$

The reverse process transforms the noised data x^k back to its input x^0. It is defined by the following Markov chain with $x^K \sim \mathcal{N}(0, \mathbf{I})$:

$$p_\theta(x^{0:K}) = p(x^K) \prod_{k=1}^{K} p_\theta(x^{k-1} \mid x^k) \text{ with } p_\theta(x^{k-1}|x^k) = \mathcal{N}(x^{k-1}; \mu_\theta(x^k, k), \sigma_k^2\mathbf{I}) \tag{9}$$

In this backwards process $\mu_\theta(x^k, k)$ is modeled using a neural network and where

$$\sigma_k^2 = \begin{cases} \beta_k & \text{optimal for } x^0 \sim N(0, \mathbf{I}) \\ \tilde{\beta}_k = \frac{1-\alpha_{k-1}}{1-\alpha_k}\beta_k & \text{optimal for } x^0 \text{ deterministically set to one point.} \end{cases} \tag{10}$$

The choices of σ_k^2 correspond to the upper and lower bounds on reverse process entropy for data with coordinatewise unit variance [51].

Training the diffusion model involves minimizing a KL-divergence loss:

$$\mathcal{L}_k = D_{KL}\left(q(x^{k-1}|x^k)||p_\theta(x^{k-1}|x^k)\right) \tag{11}$$

For more stability in training and since x^0 is known, the forward process $q(x^{k-1}|x^k)$ is replaced by:

$$q(x^{k-1}|x^k, x^0) = \mathcal{N}(x^{k-1}; \tilde{\mu}_k(x^k, x^0), \tilde{\beta}_k(x^k, k)) \tag{12}$$

$$\text{where } \tilde{\mu}_k(x^k, x^0) = \frac{\sqrt{\alpha_{k-1}}\beta_k}{1-\alpha_t}x^0 + \frac{\sqrt{(1-\beta_k)}(1-\alpha_{k-1})}{1-\alpha_k}x^k \tag{13}$$

$$\text{and } \tilde{\beta}_k = \frac{1-\alpha_{k-1}}{1-\alpha_k}\beta_k \tag{14}$$

This modification allows to reparameterize the training objective from Equation 11 to:

$$\mathcal{L}_k = \frac{1}{2\sigma_k^2}||\tilde{\mu}_k(x^k, x^0) - \mu_\theta(x^k, k)||^2 \tag{15}$$

Where x^k is obtained through Equation 8. From here, $\mu_\theta(x^k, k)$ can be expressed either by a data prediction model $x_\theta(x^k, k)$ or a noise prediction model $\epsilon_\theta(x^k, k)$ [49], [52], [53]. For image generation, Ho *et al.* [49] found that the latter performs better. Using a noise prediction model with noise loss function:

$$\mu_\epsilon(\epsilon_\theta) = \frac{1}{\sqrt{1-\beta_k}}\left(x^k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\epsilon_\theta(x^k, k)\right) \text{ with } \mathcal{L}_{\epsilon_\theta} = \mathbb{E}_{k,x^0,\epsilon}\left[||\epsilon - \epsilon_\theta(x^k, k)||^2\right] \tag{16}$$

Alternatively, using the data prediction model and its data loss function:

$$\mu_x(x_\theta) = \frac{\sqrt{1-\beta_k}(1-\alpha_{k-1})}{1-\alpha_k}x^k + \frac{\sqrt{\alpha_{k-1}}\beta_k}{1-\alpha_k}x_\theta(x^k,k) \text{ with } \mathcal{L}_{x_\theta} = \mathbb{E}_{k,x^0,\epsilon}\left[||x^0 - x_\theta(x^k,k)||^2\right] \tag{17}$$

For an elaborate derivation of all of the equations mentioned above, take a look at the paper by Luo [7]. The actual sampling of x^{k-1} is shown in Figure 6 where σ_k is defined as in Equation 10 and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ Furthermore, Benny and Wolf [52] suggest that the combination of the noise and data methods can enhance performance.

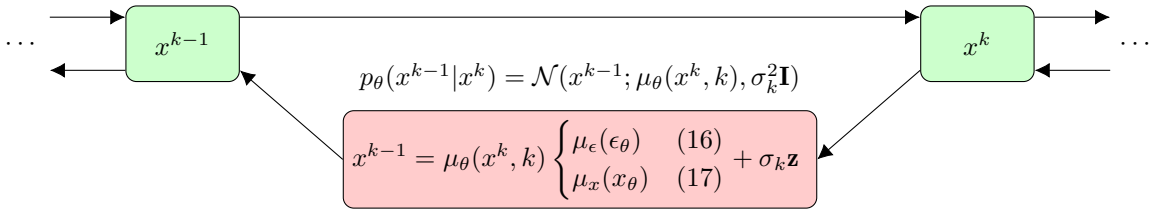

Figure 6: Illustration for prediction models of the reverse process.

2.2 Score-based generative modeling through SDE (2021) [50]

DDPMs operate in a discrete space, with discrete diffusion and denoising steps. With the use of stochastic differential equations (SDE), this process can be described in continuous time as shown in Figure 7. This can be thought of as a generalization of the DDPM, as that is a discrete form of SDE [50]. Keep in mind that even though the score-based generative modeling with SDEs is described in continuous time, the solutions are still solved iteratively [50].

The process can be formalized by letting $\mathbf{w}$ and $\bar{\mathbf{w}}$ be a standard Wiener process and its time-reverse version, respectively, and consider a continuous diffusion time $k \in [0, K]$. The forward diffusion process is described with an SDE as:

$$\mathrm{d}x = f(x,k)\mathrm{d}k + g(k)\mathrm{d}\mathbf{w} \tag{18}$$

where $g(k)\mathrm{d}\mathbf{w}$ is the stochastic diffusion and $f(x,k)\mathrm{d}k$ is the deterministic drift. On the other hand, the reverse denoising process is described with a time-reverse SDE [54]:

$$\mathrm{d}x = \left[f(x,k) - g(k)^2\nabla_x \log p_k(x)\right]\mathrm{d}k + g(k)d\bar{\mathbf{w}} \tag{19}$$

where the score of the marginal distribution $\nabla_x \log p_k(x)$ is to be estimated. It is estimated with a time-dependent score-based model $s_\theta(x,k)$ using the score objective function

$$\mathcal{L}_{s_\theta} = \mathbb{E}_{k,x(0),x(k)}\left[\left|\left|\nabla_{x(k)} \log p_{0k}(x(k)|x(0)) - s_\theta(x(k),k)\right|\right|^2\right] \tag{20}$$

Using this objective function the score-based model $s_\theta(x,k)$ can be trained on samples with score matching [50], [55]–[58].

The above formula is a general representation of a SDE, and now to be more specificly tailored to the diffusion process there are multiple methods as described in Song *et al.* [50], of which 2 are described as

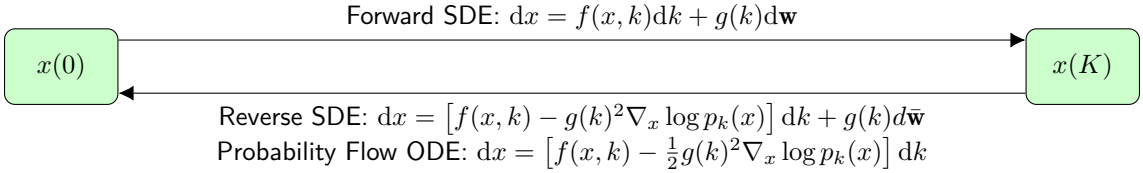

Figure 7: Illustration for prediction models of the reverse process.

following. The DDPM process can also be generalized to a SDE and is termed the variance preserving (VP) SDE:

$$\mathrm{d}x = -\frac{1}{2}\beta(k)x\mathrm{d}k + \sqrt{\beta(k)}\mathrm{d}\mathbf{w} \tag{21}$$

where $\beta(\cdot)$ is a continuous function and $\beta(\frac{k}{K}) = K(1-\beta_k)$ from (6) as $K \to \infty$. Based on the VP SDE, Song et al. [50] has designed the sub-VP SDE, specialized for likelihoods:

$$\mathrm{d}x = -\frac{1}{2}\beta(k)x\mathrm{d}k + \sqrt{\beta(k)\left(1 - \mathrm{e}^{-2\int_0^k \beta(s)ds}\right)}\mathrm{d}\mathbf{w} \tag{22}$$

Song et al. [50] demonstrated that from each SDE, one can derive an ordinary differential equation (ODE) with the same marginal distributions. This associated ODE of an SDE is called the *probability flow* ODE. Applying this concept to the reverse-SDE (19) gives the subsequent probability flow ODE:

$$\mathrm{d}x = \left[f(x,k) - \frac{1}{2}g(k)^2\nabla_x \log p_k(x)\right]\mathrm{d}k \tag{23}$$

When the $\nabla_x \log p_k(x)$ in the probability flow is replaced by its approximation $s_\theta(x,k)$, it becomes a special case of neural ODE [59]. Specifically, it resembles continuous normalizing flows because it transforms data distributions to prior noise distributions and it is fully invertible [60]. Furthermore, this allows exact log-likelihood computation and can be trained via maximum likelihood using standard methods [59].

2.3 Conditional Diffusion Models

In this section, the details are explained how the diffusion models can be conditioned. Conditioning a generative model means to control the output given some extra input. This input can be text describing an image [34], or part on an image for inpainting [34] The extra input $\mathbf{c}$ is sometimes called the covariates but often times the conditional input. Diffusion models can be conditioned in two different ways. Firstly in section 2.3.1, during training, by inputting the condition $\mathbf{c}$ into the model and training accordingly [33]. Secondly in section 2.3.2, there is Diffusion Guidance, a novel approach that shifts the conditioning from training to the inference stage, leveraging the Bayes rule [50].

2.3.1 Conditional Denoising Model

Conditional versions of the model can be extended to embrace additional input c, altering the backward denoising step in Equation 9 [33]:

$$p_\theta(x^{0:K}|\mathbf{c}) = p(x^K)\prod_{k=1}^{K} p_\theta(x^{k-1} \mid x^k, \mathbf{c}) \text{ with } p_\theta(x^{k-1}|x^k,\mathbf{c}) = \mathcal{N}(x^{k-1}; \mu_\theta(x^k,k|\mathbf{c}), \sigma_k^2\mathbf{I}) \tag{24}$$

Consequently, the noise and data variants of the prediction models are conditioned as respectively:

$$\mu_\epsilon(\epsilon_\theta,\mathbf{c}) = \frac{1}{\sqrt{1-\beta_k}}\left(x^k - \frac{\beta_k}{\sqrt{1-\alpha_k}}\epsilon_\theta(x^k,k|\mathbf{c})\right) \tag{25}$$

$$\mu_x(x_\theta,\mathbf{c}) = \frac{\sqrt{1-\beta_k}(1-\alpha_{k-1})}{1-\alpha_k}x^k + \frac{\sqrt{\alpha_{k-1}}\beta_k}{1-\alpha_k}x_\theta(x_k,k|\mathbf{c}) \tag{26}$$

where the conditioning in practice are extra inputs to the prediction model as illustrated in Figure 8. The same condition is included in each backwards step.

For the score-based generative models through SDE such conditioning also exists, where the conditioning happens in the marginal distributions of the score [61]. The reverse-time SDE becomes:

$$dx = \left[f(x,k) = g(k)^2\nabla_x \log p_k(x|\mathbf{c})\right]dk + g(k)d\bar{w} \tag{27}$$

and the objective function then becomes

$$\mathcal{L}_k = \mathbb{E}_{k,x(0),x(k)}\left[\left|\left|\nabla_{x(k)}\log p_{0k}(x(k)|x(0)) - s_\theta(x(k),k|\mathbf{c})\right|\right|^2\right] \tag{28}$$

.

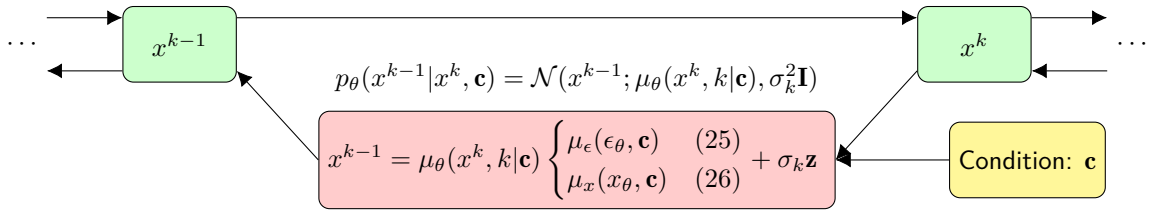

Figure 8: Illustration for conditional prediction models of the reverse process.

2.3.2 Diffusion Guidance

In traditional diffusion models, conditioning is typically introduced during the training process, as represented in equation (24). However, a novel approach introduces conditions during the inference stage by employing the concept of guidance [50], [51]. This method is grounded in the application of the Bayes rule to the probability $p(x^k|\mathbf{c})$ given by:

$$p(x^k|\mathbf{c}) = \frac{p(\mathbf{c}|x^k) \times p(x^k)}{p(\mathbf{c})} \tag{29}$$

Taking the natural logarithm of both sides, we can further derive:

$$\log p(x^k|\mathbf{c}) = \log p(\mathbf{c}|x^k) + \log p(x^k) - \log p(\mathbf{c}) \tag{30}$$

From which, upon differentiation with respect to x^k, we obtain:

$$\nabla_{x^k} \log p(x^k|\mathbf{c}) = \nabla_{x^k} \log p(\mathbf{c}|x^k) + \nabla_{x^k} \log p(x^k) \tag{31}$$

The relation in (31) provides a bridge to guide the reverse diffusion process in equation (9) and (19). Instead of merely reversing the diffusion, this guides the samples to conform to the desired condition $\mathbf{c}$ [50], [51]. Thus, equation (9) can be conditioned as:

$$p_\theta(x^{k-1}|x^k, \mathbf{c}) = \mathcal{N}(x^{k-1}; \mu_\theta(x^k, k), \sigma_k^2\mathbf{I}) + s\sigma_k^2 \nabla_{x^k} \log p(\mathbf{c}|x^k) \tag{32}$$

Here, $s\sigma_k^2 \nabla_{x^k} \log p(\mathbf{c}|x^k)$ is the guidance term that drives the generated image towards the target condition. s is a scale parameter controlling the guidance strength. The gradient $\nabla_x \log p_k(\mathbf{c}|x)$ can be estimated in two ways: the first involves training an auxiliary model as described by [7] and the second method, described by Luo [7] and Song *et al.* [50], which doesn't require a auxiliary model. In Figure 9 the $\nabla_{x^k} \log p(\mathbf{c}|x^k)$ is termed $g(\mathbf{c}, x^k)$ as the guidance gradient.

Such guidance can also be applied to the continuous diffusion models as described by Song *et al.* [50]. For equation (19), it can be conditioned as:

$$\mathrm{d}x = \left\{ f(x,k) - g(k)^2 \left[\nabla_x \log p_k(x) + \nabla_x \log p_k(\mathbf{c}|x) \right] \right\} \mathrm{d}k + g(k) d\bar{w} \tag{33}$$

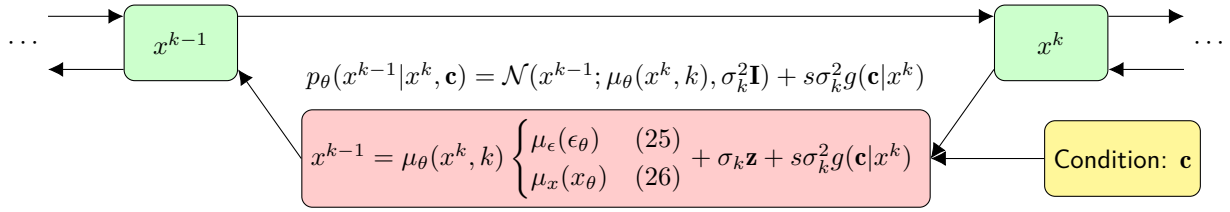

Figure 9: Illustration for conditional prediction models of the reverse process.

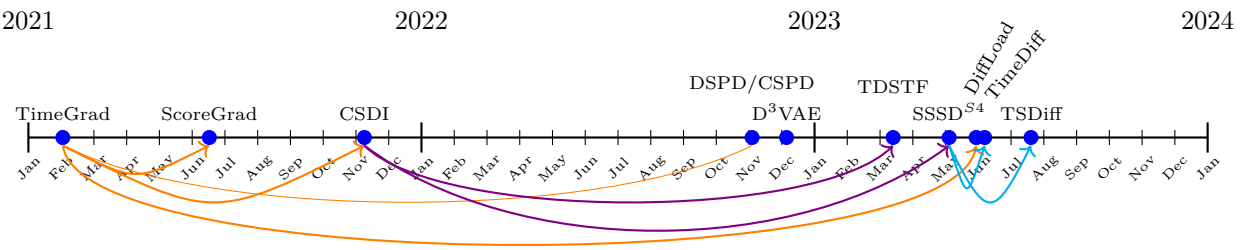

Figure 10: Diffusion for Time-series Forecasting Research Timeline

Paper	Name	Method	Input Data	AR	Gen. Method	Rev. Model	Task	Horizon
[63] (3.1)	TimeGrad	Cond. Model	⬅️	🔁	DDPM	Noise	🔮	Short
[61] (3.2)	ScoreGrad	Cond. Model	⬅️	🔁	SDE	Score	🔮	Short
[64] (3.3)	CSDI	Cond. Model	↔️	🔦	DDPM	Noise	🩹	Short
[65] (3.4)	DSPD	Cond. Model	↔️	🔦	DDPM	Noise	🩹	Short
[65] (3.4)	CSPD	Cond. Model	↔️	🔦	SDE	Noise	🩹	Short
[66] (3.5)	D^{3}VAE	Cond. Model	⬅️	🔦	Coupled DPM	BVAE	🔮	Short
[67] (3.6)	TDSTF	Cond. Model	⬅️	🔦	DDPM	Noise	🔮	Short
[68] (3.7)	SSSD^{S4}	Cond. Model	↔️	🔦	DDPM	Noise	🩹	Long
[16] (3.8)	DiffLoad	Cond. Model	⬅️	🔁	DDPM	Noise	🔮	Short
[69] (3.9)	TimeDiff	Cond. Model	⬅️ + Extras	🔦	DDPM	Data	🔮	Long
[70] (3.10)	TSDiff	Guidance	⬅️	🔦	DDPM	Noise	🔮	Short

Table 1: Comparison of diffusion models for time-series prediction. **Method**: Conditioned Model (2.3.1), Guidance (2.3.2); **Input Data**: ⬅️ Data Before Prediction, ↔️ Data Before and After Prediction; **AR**: 🔁 Autoregressive, 🔦 Generated at Once (Non-AR); **Generative Method**: DDPM (2.1), SDE (2.2) **Reverse Model**: Noise (16), Data (17), Score (27); **Task**: 🔮 Forecasting, 🩹 Imputation. **Horizon**: Whether evaluated for Short ($\leq$ 50 steps) or Long ($\geq$ 50 steps) horizons.

3 Diffusion Models for Time Series Modeling

Time-series modeling using diffusion models has resurrected from sound generation via the methods WaveNet [62] and DiffWave [39], with the pioneering implementation named TimeGrad by Rasul *et al.* [63]. From the TimeGrad paper, many more followed as illustrated in Figure 10, and will be layed out in this chapter. First a brief introduction to the specifics of conditioning diffusion models for time-series forecasting is given. The conditioning of diffusion models for time-series can be done by conditioning the generation of the forecast through the reverse process in Equation 24 on the historical data like so:

$$p_\theta(X_{tar}^{0:K}|X_{obs}) = p(X_{tar}^K)\prod_{k=1}^{K} p_\theta(X_{tar}^{k-1} \mid X_{tar}^k, X_{obs}) \text{ where } p(X_{tar}^K) \sim \mathcal{N}(0, \mathbf{I}) \tag{34}$$

The probability of $p_\theta(X_{tar}^{k-1} \mid X_{tar}^k, X_{obs})$ can be implemented through a conditional model as described in 2.3.1 or guidance in 2.3.2 with corresponding loss functions. The conditioning is not the historical data directly, but an encoding with the use of RNNs or Transformers [20] as will be evident in the following sections. In Table 1 an overview is presented of each of the implementations and their characteristics. In each section discussing a paper, the inspirational papers are mentioned that served as the foundation for the work. This is followed by an in depth but brief overview of the theory and intuition behind the models. Then the datasets and results are presented, finalizing with honest remarks and possible shortcomings are discussed.

Dataset	DIM.	DOM.	FREQ.	# STEPS
Exchange	8	$\mathbb{R}^+$	DAY	6,071
Solar	137	$\mathbb{R}^+$	HOUR	7,009
Electricity	370	$\mathbb{R}^+$	HOUR	5,833
Traffic	963	(0,1)	HOUR	4,001
Taxi	1,214	$\mathbb{N}$	30-MIN	1,488
Wikipedia	2,000	$\mathbb{N}$	DAY	792

Table 2: Gluon-TS Dataset Descriptions.

3.1 TimeGrad: Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting (2021) [63]

TimeGrad represents the pioneer in using diffusion models for time-series forecasting, where its methodology lays in the DDPM [49]. The model architectures are based on those of WaveNet [62] and DiffWave [39].

TimeGrad was skillfully engineered to make probabilistic forecasts of the multivariate time-series, leveraging both past data and covariates. In this context covariates are extra conditions besides the historical data. Specifically, it employs an RNN mechanism to encode the time-series sequence, which, when combined with the DDPM, enables TimeGrad to proficiently learn the distribution of ensuing time steps. Given the use of the RNN, the model works by diffusing and denoising each multivariate value vector x_t in an autoregressive manner. For this it encodes the historical data and covariates using the RNN starting from x_{-H} and $\mathbf{h}_{-H} = 0$ as such

$$\mathbf{h}_t = \text{RNN}_\theta(\text{concat}(x_t^0, c_t), \mathbf{h}_{t-1}) \tag{35}$$

Where the reverse process as described in Equation 34 is done over the entire forecast, in TimeGrad it is done per time step.

$$p_\theta(X_{tar}^{0:K}|X_{obs}) = \prod_{t=1}^{F} p_\theta(x_t^{0:K}|\mathbf{h}_{t-1}) \tag{36}$$

Where $\mathbf{h}_0$ is the encoded historical data plus covaraites, furthermore, you can see, the conditioning $\mathbf{h}_{t-1}$ also evolves as the predictions go into the future. To continue, the paper models $\prod_{t=1}^{F} p_\theta(x_t^{0:K}|\mathbf{h}_{t-1})$ through the diffusion process, resulting in the extension of (36), which is in par with equation (34) as such:

$$p_\theta(X_{tar}^{0:K}|X_{obs}) = \prod_{t=1}^{F} p(x_t^K) \prod_{k=1}^{K} p_\theta(x_t^{k-1}|x_t^k, \mathbf{h}_{t-1}) \tag{37}$$

Because of this, the training objective is also extended to minimize the KL divergence over the entire forecast, which can be further simplified to with a noise prediction model:

$$\mathcal{L}_{t,\epsilon_\theta} = \mathbb{E}_{k,x_t^0,\epsilon}\left[||\epsilon - \epsilon_\theta(x^k, k|\mathbf{h}_{t-1})||^2\right] \tag{38}$$

$$\mathcal{L}_{\epsilon_\theta} = \frac{1}{F}\sum_{t=1}^{F} \mathcal{L}_{t,\epsilon_\theta} \tag{39}$$

The paper continuous to evaluate the model on six datasets. These datasets are preprocessed and stored by Salinas *et al.* [71][1] and can be easily accessed through GluonTS[2] and are described in Table 2. The number of historical steps used for conditioning is equal to the prediction length. The datasets and their original sources are Exchange [73], Solar [73], Electricity [74], Traffic [75], Taxi [76], and Wiki [77]. The results are displayed in Table 3.

While TimeGrad was recognized as state-of-the-art in multivariate probabilistic time-series forecasting at the time [63], it was not without limitations. Notably, its utilization of an RNN to encode historical time

[1] GP-Copula [71]: `https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets`
[2] GluonTS: Probabilistic Time Series Models in Python [72] `https://github.com/awslabs/gluonts`

Dataset	H	F	TimeGrad		
			CRPS_{sum}	CRPS	MSE
Exchange	24	24	0.006±0.001	0.009±0.001	2.5e−4
Solar	24	24	0.287±0.020	0.367±0.001	8.8e2
Electricity	24	24	0.0206±0.001	0.049±0.002	1.97e5
Traffic	24	24	0.044±0.006	0.110±0.002	4.2e−4
Taxi	24	24	0.114±0.02	0.311±0.03	2.6e1
Wikipedia	30	30	0.0485±0.002	0.261±0.02	3.8e7

Table 3: TimeGrad Multivariate Forecasting results [63]

steps renders it challenging to apply to long-term time-series because it introduces the error accumulation issue [69]. As a recommendation Rasul *et al.* [63] suggest the use of Transformers [20]. Furthermore, The use of an RNN to encode the historical data and make future predictions makes it much slower compared to newer methods [69]. Overall, TimeGrad has served as the foundational paper for time-series forecasting, opening the doors for more diffusion research into this problem space.

3.2 ScoreGrad: Multivariate Probabilistic Time Series Forecasting with Continuous Energy-based Generative Models (2021) [61]

Where TimeGrad [63] uses DDPM [49] for the diffusion process, ScoreGrad uses SDE [50] for the diffusion process as described in section 2.2. This shift to using SDE brings changes in terms of implementating the diffusion denoising processes, but the rest of the paper resembles that of TimeGrad. Just as TimeGrad the model is designed for forecasting tasks of multivariate time-series by diffusion and denoising single time-series vectors x_t, furthermore ScoreGrad also uses an RNN encode the historical data. Yan *et al.* [61] define this encoding with $\mathbf{F}_t$ and the update function as:

$$\mathbf{F}_t = R(\mathbf{F}_{t-1}, x_{t-1}(0), c_{t-1}) \tag{40}$$

This is much like the hidden state in TimeGrad, but the difference is that Yan *et al.* [61] leave the implementation up the user, suggesting the use of Temporal Convolutional Networks [62] or Transformers [20]. Just as the conditioning in (27), ScoreGrad conditions the reverse process with the encoding of $\mathbf{F}_t$ as follows:

$$\mathrm{d}x_t = \left[f(x_t, k) = g(k)^2 \nabla_{x_t} \log p_k(x_t|\mathbf{F}_t)\right] \mathrm{d}k + g(k)\mathrm{d}\bar{w} \tag{41}$$

With this extra conditioning and the autoregressive method, the loss function as shown in 28 changes to:

$$\mathcal{L}_{t,s_\theta} = \mathbb{E}_{k,x_t(0),x_t(k)} \left[\left|\left| \nabla_{x_t(k)} \log p_{0k}(x_t(k)|x_t(0)) - s_\theta(x_t(k), k|\mathbf{F_t}) \right|\right|^2 \right] \tag{42}$$

Again, keep in mind that only a single value vector is diffused and denoised, and this is done for every future time step. As a result, the final loss function for the whole forecast is:

$$\mathcal{L}_{s_\theta} = \sum_{t=1}^{F} \mathcal{L}_{t,s_\theta} \tag{43}$$

The authors evaluate ScoreGrad on the same datasets accessed through GluonTS[2] as TimeGrad and compare with the CRPS_{sum} evaluation metric. The results show that using the sub-VP SDE from equation (22) give the best results on all datasets apart from Electricity.

The results are a clear indication that the use of SDE for the diffusion process is superior compared to DDPM in this use-case. Furthermore, because ScoreGrad resembles TimeGrad so much, the remarks are similar too, such as for future works and as Yan *et al.* [61] already mentioned, it could benefit from the use

Dataset	H	F	CRPS_{sum}		
			TimeGrad	VP SDE	sub-VP SDE
Exchange	30	24	0.006±0.001	0.006±0.001	0.006±0.001
Solar	24	24	0.287±0.020	0.268±0.021	**0.256±0.015**
Electricity	24	24	0.0206±0.001	**0.0192±0.001**	0.0194±0.001
Traffic	24	24	0.044±0.006	0.043±0.004	**0.041±0.004**
Taxi	24	30	0.114±0.02	0.102±0.006	**0.101±0.004**
Wikipedia	30	24	0.0485±0.002	**0.041±0.003**	0.043±0.002

Table 4: ScoreGrad Multivariate Forecasting results [61]

of more advanced encoding models like Transformers [20] to extract time-series features. Furthermore, the forecasting is again autoregressive which causes errors to accumulate over the time horizon and is slow [69]. It is noteworthy that Yan *et al.* [61]'s paper omits specific details on the ODE application, as described in Equation 23. However they do mention it briefly on the GitHub page and it allows for faster sampling option the use of ODE, allowing a increase of up to 4.9 times for the the prediction speed[3].

3.3 CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation (2021) [64]

The CSDI implementation is much like that of TimeGrad [63], which is also based on DiffWave [39]. However instead of using a RNN to encode the conditional input, CSDI follows the future works of TimeGrad and makes use of a Transformer [20]. This modification allows the conditional input to be at arbitrary locations with respect to the target values, which allows imputation.

CSDI changes the problem statement to be compatible with imputation and does this by introducing a mask variable m and a timestamp s to the value vector, resulting in each time step resembling $\{x, m, s\}$. By introducing the mask, it can define which values are missing and require predictions. Keep in mind that forecasting is a specific case of imputation where all masked values are at the end of the time series sample. Furthermore, by introducing the timestamp, CSDI assumes that time intervals between consecutive data entries can be different allowing more flexibility in the data.

CSDI uses a slight modification of formalization of reverse process described in Equation 34, namely because of the imputation task, the *tar* and *obs* can be arbitrary points on the time-series and not necessarily split in historical and future points. Thus it becomes

$$p_\theta(X_{ta}^{0:K}|X_{co}) = p(X_{ta}^K)\prod_{k=1}^{K} p_\theta(X_{ta}^{k-1} \mid X_{ta}^k, X_{co}) \text{ where } p(X_{ta}^K) \sim \mathcal{N}(0, \mathbf{I}) \tag{44}$$

where $X_{ta}^{0:K}$ are the target timestamps and X_{co} the conditional timestamps, future and past.

CSDI uses the same parameterization of DDPM and conditions the model itself as in Equation 25. This also leads to corresponding conditioned noise loss function

$$\mathcal{L}_{\epsilon_\theta} = \mathbb{E}_{k, X_{ta}^0, \epsilon}\left[||\epsilon - \epsilon_\theta(X_{ta}^k, k|X_{co})||^2\right] \tag{45}$$

In CSDI, the same datasets are used as in TimeGrad and ScoreGrad but Exchange [73] is left out. In the paper, mistakes about the descriptions of the datasets are made, namely the total number of steps as described in CSDI for the Traffic & Solar datasets are wrong. They should be 4001 and 7009 respectively to be the same as in TimeGrad & ScoreGrad through Gluon-TS[2]. These mistakes were verified through contact with the author on GitHub. Furthermore the historical window for the forecasting is increased, compared to Tables 3 and 4.

[3] `https://github.com/yantijin/ScoreGradPred`

Dataset	H	F	CSDI		
			CRPS	CRPS_{sum}	MSE
Solar	168	24	0.338 ± 0.012	0.298 ± 0.004	9.0e2 ± 6.1e1
Electricity	168	24	0.041 ± 0.000	0.017 ± 0.000	1.1e5 ± 2.8e3
Traffic	168	24	0.073 ± 0.000	0.020 ± 0.001	3.5e−4 ± 7.0e−7
Taxi	48	24	0.271 ± 0.001	0.123 ± 0.003	1.7e1 ± 6.8e−2
Wiki	90	30	0.207 ± 0.002	0.047 ± 0.003	3.5e7 ± 4.4e4

Table 5: CSDI Multivariate Forecasting results [64]

Comparing the results to TimeGrad should be approached with caution, as the extended historical window length used in the experiments may serve as an advantage. This extended context could positively impact the results, potentially diminishing the apparent superiority of CSDI's forecasting performance.

Furthermore, the method of masking used in CSDI is similar to inpainting in computer vision tasks. This inpainting has shown to cause artifacts between masked and observed regions call boundary disharmony [78]. The masking method used in CSDI is also empirically shown the have the same effects in time-series [69].

CSDI employs a automatic masking algorithm in order to generate the training data. This feature has a negative effect in situations where the data is extremely sparse as tackled in TDSTF by Chang *et al.* [67]. In this paper, CSDI is used as a baseline for sparse data and does not deal with such sparse situations by excluding enough invalid information and thus over 99.5% of the conditional data points were missing on average. This resulted in TDSTF being a order of magnitude faster than CSDI in training.

Furthermore the computation time of the denoising network is quadratic in the number of variables and number of time-series points because it is based on two transformer networks. This causes it to run out of memory easily when modeling long multivariate time-series [69].

Just like SSSD^{S4} the conditioning is done in the denoising networks' intermediate layers and introduce inductive bias into the denoising objective, Shen and Kwok [69] hypothesises that this may not be enough to guide the network in capturing information from the history and leads to inaccurate predictions.

3.4 DSPD & CSPD: Modeling Temporal Data as Continuous Functions with Process Diffusion (2022) [65]

The implementation is based on that of TimeGrad [63] with two key contributions. The main contribution and novelty of this paper is that they model the time-series as continuous functions instead of discrete sequences. This allows the model to predict values at any time, even between the existing time-series data steps. The second contribution is a change in the architecture that allows the model to predict multiple values at the same time which scales better on modern hardware.

Formally, they make the assumption that each observed time series comes from an underlying continuous function $x(\cdot)$ and thus are interested in modeling the distribution $p(x(\cdot))$. Because of this assumption, they have to add a time correlated noise function $\epsilon(\cdot)$ to $x(\cdot)$ instead of independent noise. As noise functions they propose a Gaussian process prior (GP) or a Ornstein-Uhlenback (OU) diffusion process. Biloš *et al.* [65] define both the GP and OU processes with a multivariate normal distribution $\mathcal{N}(0, \Sigma)$. Both $\epsilon(\cdot)$ and Σ are calculated using the timestamps of the observations.

They continue by proposing their implementations Discrete Stochastic Process Diffusion (DSPD) and Continuous Stochastic Process Diffusion (CSPD), based on DDPM and SDE respectively. The fact that the implementations model continuous functions has nothing to do with whether the diffusion process is done discretely or continuously. In both the DSPD and CSPD, they use a forecasting model that outputs noise ϵ as in Equation 25. For CSPD, they refactor the noise predictions to the score function value.

For the case of forecasting, an RNN is used to obtain the historical condition $\mathbf{z}$. This condition does not change during the forecasting as poposed to TimeGrad and ScoreGrad. Furthermore, because the architecture uses 2D instead of 1D convolutional layers, the model is capable of forecasting all of the values at once.

Discrete Stochastic Process Diffusion (DSPD): In the reverse process the marginal distributions, as in Equation 24 are adjust for the assumption of modeling continuous functions as such:

$$p_\theta(X_{tar}^{k-1} \mid X_{tar}^k, X_{obs}) = \mathcal{N}(x^{k-1}; \mu_\theta(x^k, t, k|\mathbf{z}), \sigma_k^2 \Sigma) \tag{46}$$

where t are the timestamps of the to be forecasted values and $\mathbf{z}$ the encoded historical data. Furthermore, the variance is adjusted to the time correlated noise with Σ instead of $\mathbf{I}$ as in Equation 9.

Continuous Stochastic Process Diffusion (CSPD): Given the factorized covariance matrix $\Sigma = \mathbf{L}\mathbf{L}^T$ the variance preserving SDE of Equation 21 is modified to:

$$dX_{tar}^k = -\frac{1}{2}\beta(k)X dk + \sqrt{\beta(k)}\mathbf{L}dW \tag{47}$$

For both DPSD and CSPD, the authors employ the reparameterization to predict noise which also results in the following objective function:

$$\mathcal{L}_{\epsilon_\theta} = \mathbb{E}_{k, X_{tar}^0, \epsilon}\left[||\epsilon - \epsilon_\theta(X_{tar}^k, t, k, |\mathbf{z})||^2\right] \tag{48}$$

Unfortunately, the paper does not compare with the same CRPS_{sum} metric as previous papers, but it does state that the order compared to TimeGrad does not change. The compare on NRMSE and the energy score [79].

Dataset	H	F	NRMSE		Energy Score	
			TimeGrad	DSPD-GP	TimeGrad	DSPD-GP
Electricity	24	24	0.064 ± 0.007	$\mathbf{0.045 \pm 0.002}$	8425 ± 613	$\mathbf{7079 \pm 164}$
Exchange	30	30	0.013 ± 0.003	$\mathbf{0.012 \pm 0.001}$	0.057 ± 0.002	$\mathbf{0.031 \pm 0.002}$
Solar	24	24	0.799 ± 0.096	$\mathbf{0.757 \pm 0.026}$	$\mathbf{150 \pm 17}$	166 ± 12

Table 6: DSPD-GP Multivariate Forecasting results [65]

The process has the same limitation as TimeGrad[3.1] and ScoreGrad [3.2], namely that it uses an RNN to represent the historical data. The paper states to use the same architecture, with Transformer, as CSDI [3.3] for imputation and only changes the noise source to that of a Gaussian process. The paper only compares to TimeGrad for forecasting capabilities, which is rather disappointing because it later on also compares the imputation capabilities with CSDI. CSDI already performs better at forecasting than TimeGrad, making me wonder whether this was done deliberately. Furthermore, the code is not available for reproducibility. The paper could make a much stronger case for the superiority of time correlated noise if they had evaluated it against more time-series forecasting models and used Transformers for encoding the historical data.

3.5 D^3 VAE: Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement (2022) [66]

This paper, even though it mentions the diffusion and denoising process of Ho *et al.* [49], it does so differently from what has been described so far in this survey. Furthermore, the paper focuses on the interpretability of the forecasts and tries to decompose the aleatoric and epistemic uncertainty of the data and model. The model contains three key components. Firstly a coupled diffusion process, secondly a bidirectional variational auto-encoder(BVAE) [80] for disentanglement, and lastly extra denoising through denoising score matching (DSM). These three components also motivate the name with diffusion denoise and disentanglement, hence D^3 VAE.

Coupled Diffusion Process: The name, coupled diffusion process, comes from the fact that it adds noise to both the historical and future windows of a sample. This helps because the diffused samples can augment the dataset giving more support to forecasting with short and noisy time series. Furthermore, the paper proves that by decomposing the X_{tar}^k from Equation 8 and X_{obs}^k, into an ideal part and noisy part, it

can reduce the difference between diffusion noise and generation noise. The decomposition is formalized as:

$$X_{tar}^k = \sqrt{\alpha'_k} X_{tar}^0 + \sqrt{1-\alpha'_k}\epsilon_{X_{tar}} = \underbrace{\sqrt{\alpha'_k}\bar{X}_{tar}}_{\text{ideal part}} + \underbrace{\sqrt{\alpha'_k}\delta_{X_{tar}} + \sqrt{1-\alpha'_k}\epsilon_{X_{tar}}}_{\text{noisy part}} \tag{49}$$

$$X_{obs}^k = \sqrt{\alpha_k} X_{obs}^0 + \sqrt{1-\alpha_k}\epsilon_{X_{obs}} = \underbrace{\sqrt{\alpha_k}\bar{X}_{obs}}_{\text{ideal part}} + \underbrace{\sqrt{\alpha_k}\delta_{X_{obs}} + \sqrt{1-\alpha_k}\epsilon_{X_{obs}}}_{\text{noisy part}} \tag{50}$$

where the diffusion of the targets uses a scaled version of α_k, namely given a scale parameter $\omega \in (0, 1)$ such that $\beta'_k = \omega\beta_k$ and subsequently $\alpha'_k = \prod_{i=1}^{k}(1-\beta'_i)$. The noisy part is split into the diffusion noise ϵ_X and data noise δ_X. In other words, this decomposition reduces the uncertainty of the generated forecasts as proven by Li *et al.* [66].

Bidirectional Variational Auto-Encoder: The second component is the more efficient generative model, i.e. BVAE [80], that replaces the reverse process. Besides replacing the reverse process the BVAE is also used to facilitate model interpretability through disentanglement. In contrast to the standard reverse process that diffuses and denoises only the targets, the BVAE processes the diffused observations X_{obs}^k. Formally, the BVAE encodes the X_{obs}^k into a latent variable Z as $p_\phi(Z, X_{obs}^k)$. Given this latent variable the decoder generates the prediction $\hat{X}_{tar}^k$ through $p_\theta(\hat{X}_{tar}^k, Z)$. The generated prediction still contains noise, which is subsequently removed by the Scaled DSM. The Latent variable Z is used for disentanglement, which is achieved by minimizing the Total Correlation [81], [82]. This helps with interpretability by identifying the independent factors of the data [83]–[85].

Scaled Denoising Score Matching: The forecasts produced by the BVAE tend to move towards the diffused target series. To further clean the generated target series $\hat{X}_{tar}^k$, the authors employ the DSM to accelerate the de-uncertainty process without sacrificing the model flexibility. This denoising is done via a single-step gradient jump [86].

$$\hat{X}_{tar}^0 = \hat{X}_{tar}^k - \sigma_0^2 \nabla_{\hat{X}_{tar}^k} E(\hat{X}_{tar}^k, \zeta) \tag{51}$$

where σ_0 is a hyperparameter and $E(\hat{X}_{tar}^k, \zeta)$ is an energy function. The denoising step is taken in the direction of the greatest increase of this energy function, hence $\nabla_{\hat{X}_{tar}^k}$.

All of these components are trained using their own loss functions, which results in the total loss function:

$$\mathcal{L} = \underbrace{\psi \cdot \mathcal{D}_{KL}\left(q(X_{tar}^k || p_\theta(X_{tar}^k))\right)}_{BVAE} + \underbrace{\lambda \cdot \mathcal{L}(\zeta, k)}_{DSM} + \underbrace{\gamma \cdot \mathcal{L}_{TC}}_{Disent.} + \underbrace{\mathcal{L}_{MSE}(\hat{X}_{tar}^k, X_{tar}^k)}_{BVAE} \tag{52}$$

D^{3}VAE use the datasets: Traffic [75][4], Electricity [74], Weather [87], ETTm1[23][5], ETTh1[23][5], Wind[66][6]. It is not explained how the datasets of Electricity [74] and Weather [87] are gathered or pre-processed, only the source websites are mentioned. It is important to note that the Traffic and Electricity datasets used in D^{3}VAE do not share characteristics with those used from GluonTS[2] in TimeGrad, ScoreGrad or CSDI even though they are from the same source. Furthermore and because of the motivation to be able to train on short time-series datasets via augmentation, the authors only take a small percentage of the datasets previously mentioned. The results are presented in Table 7.

Even though not directly comparable, the results of TimeGrad in Table 7 are way worse than what is presented in Table 3. My initial thought was that this was due to the fact that D^{3}VAE only takes a small percentage of the dataset to show that its augmentation improves the results. However, for the dataset Electricity they also experimented with using the full dataset and the performance of TimeGrad does not seem to resemble the same performance as presented by Rasul *et al.* [63]. Moreover, in the D^{3}VAE paper are plots of the TimeGrad performance and it is not on par with what is shown by Rasul *et al.* [63] even though they both use the same dataset source. Luckily, the authors of TimeDiff 3.9 have used D^{3}VAE in their comparison and the results show for multivariate time-series forecasting with long forecasting windows that the model does perform better than TimeGrad and CSDI [69]. The paper claims the BVAE replaces the reverse process while the inputs to the BVAE is the observation section of the time-series instead of the

[4] LSTNet [73]: `https://github.com/laiguokun/multivariate-time-series-data/tree/master`
[5] Informer [23]: `https://github.com/zhouhaoyi/ETDataset`
[6] D^{3}VAE [66]: `https://github.com/PaddlePaddle/PaddleSpatial/tree/main/paddlespatial/datasets`

Dataset	H	F	MSE		CRPS	
			D3VAE	TimeGrad	**D3VAE**	TimeGrad
Traffic (5%)	8	8	0.081 ± 0.003	3.695 ± 0.246	0.207 ± 0.003	1.410 ± 0.027
	16	16	0.081 ± 0.009	3.495 ± 0.362	0.200 ± 0.014	1.329 ± 0.057
	32	32	0.091 ± 0.007	5.195 ± 2.26	0.216 ± 0.012	1.565 ± 0.329
	64	64	0.125 ± 0.005	3.692 ± 1.54	0.244 ± 0.006	1.412 ± 0.257
Electricity (3%)	8	8	0.251 ± 0.015	2.703 ± 0.087	0.398 ± 0.011	1.208 ± 0.024
	16	16	0.308 ± 0.030	2.770 ± 0.237	0.437 ± 0.020	1.240 ± 0.048
	32	32	0.410 ± 0.075	2.640 ± 0.138	0.534 ± 0.058	1.234 ± 0.027
Weather (2%)	8	8	0.169 ± 0.022	1.110 ± 0.083	0.357 ± 0.024	0.733 ± 0.016
	16	16	0.187 ± 0.047	1.065 ± 0.145	0.361 ± 0.046	0.724 ± 0.021
	32	32	0.203 ± 0.008	1.178 ± 0.069	0.383 ± 0.007	0.696 ± 0.011
	64	64	0.191 ± 0.022	1.063 ± 0.061	0.358 ± 0.044	0.696 ± 0.011
ETTm1 (1%)	8	8	0.527 ± 0.073	0.984 ± 0.074	0.557 ± 0.048	0.908 ± 0.038
	16	16	0.968 ± 0.104	2.032 ± 0.234	0.821 ± 0.072	0.919 ± 0.031
	32	32	0.707 ± 0.061	1.251 ± 0.133	0.697 ± 0.040	0.822 ± 0.032
ETTh1 (5%)	8	8	0.292 ± 0.036	4.259 ± 1.13	0.424 ± 0.033	1.092 ± 0.028
	16	16	0.374 ± 0.061	1.332 ± 0.125	0.488 ± 0.039	0.879 ± 0.037
	32	32	0.334 ± 0.008	1.514 ± 0.042	0.461 ± 0.004	0.925 ± 0.016
	64	64	0.349 ± 0.039	1.150 ± 0.118	0.473 ± 0.024	0.835 ± 0.045
Wind (2%)	8	8	0.681 ± 0.075	12.67 ± 1.75	0.596 ± 0.052	1.440 ± 0.059
	16	16	1.033 ± 0.062	12.86 ± 2.60	0.757 ± 0.053	1.240 ± 0.070
	32	32	1.224 ± 0.060	13.10 ± 0.955	0.869 ± 0.074	1.518 ± 0.020
	64	64	0.902 ± 0.024	3.857 ± 0.597	0.761 ± 0.021	1.110 ± 0.143

Table 7: D^3VAE Multivariate forecasting results [66]

future section. This makes the diffusion process untypical more difficult [69]. Furthermore, the authors claim that the outputs of the BVAE converge to the diffused outputs of the coupled diffusion process. The outputs of the coupled diffusion process are maximum noise, indicating that the BVAE tends to move towards that noisy output. Moreover, the final denoising step is taken by the DSM creating the cleaned-up prediction. The model has many hyperparameters that need to be tuned, this can be a significant limitation in finding its optimal performance.

3.6 TDSTF: Transformer-based Diffusion probabilistic model for Sparse Time series Forecasting (2023) [67]

The TDSTF model, drawing inspiration from CSDI [64], is specifically designed for forecasting ICU patient vital signs using sparse historical data. Addressing the challenge of sparse multivariate ICU data, the model introduces key innovations. Its main contribution is efficiently representing the spare data with a triplet form, which contains a feature, time, and value. On top of that, there is also a mask bit indicating the presence or absence of data of a triplet. This compact representation effectively stores sparse data, maintaining the integrity of temporal information and reducing the noise that typically arises from imputation or aggregation methods. Furthermore, as the name implies, a Transformer [20] is used to encode the historical data. There are also adjustments in selecting the inputs to the encoder. First of all, if the mask indicates the absence of a triplet, it won't be included to the Transformer. Furthermore, if there are too many triplets as input to the model, only the data points of the same target feature, or most correlated to the target data are included. This ensures that the most relevant information is included in the model's input.

The reverse process is formalized as in Equation 34 with a noise prediction model and corresponding loss

function

$$\mathcal{L}_{\epsilon_\theta} = \mathbb{E}_{k, X^0_{tar}, \epsilon} \left[||\epsilon - \epsilon_\theta(X^k_{tar}, k|X_{obs})||^2 \right]. \tag{53}$$

The ICU data contains the vital signs Heart Rate (HR), Systolic Blood Pressure (SBP), and Diastolic Blood Pressure (DBP). Furthermore, it also contains special events that occur with the patient such as administering drugs. This approach allows for a more accurate forecast by considering the interrelated events, interventions, and conditions that could impact the patient. The paper states that 60 triplets are used to encode all of the historical data, however it is not mentioned how many steps in the future the vital signs are predicted. The paper merely states it makes a 10-minute forecast of the three vital signs HR, SBP, and DBP.

The specific data set is MIMIC-III [88][7] and the performance are evaluated with MSE and NACRPS as described in subsubsection 1.1.2. The results are presented in Table 8.

Dataset	H	F	NACRPS		MSE	
			CSDI	TDSTF	CSDI	TDSTF
MIMIC-III	60tr	10m	0.5470 ± 0.0040	**0.4438± 0.0078**	0.6341 ± 0.0098	**0.4168 ± 0.0232**

Table 8: TDSTF ICU Vital Signs Forecasting results [67]

Besides achieving better predictive results, TSDTF is also much faster to train and inference compared to CSDI [3.3]. In the experiments, CSDI has 280 thousand parameters and TDSTF 560 thousand, even with this increase, TDSTF only took 30 minutes to train compared to the 12 hours for CSDI. Furthermore, CSDI took 14.913 seconds to inference while TDSTF only took 0.865 seconds, making TDSTF more than 17 times faster.

Unfortunately, TSDTF only shown to be an improvement over CSDI for the sparse dataset MIMIC-III[7]. More insights in its performance could be achieved by testing it against the same datasets mentioned in the CSDI paper [64]. Overall the paper makes a strong case, solving many of the shortcomings of CSDI when it comes to sparse data.

3.7 SSSD^{S4}: Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models (2023) [68]

The model of SSSD^{S4}, is heavily inspired by DiffWave [39] and CSDI [64], with three main differences. The first is the fact that SSSD^{S4} uses S4 models [89] instead of dilated convolutions or transformer layers [20]. The S4 models are computationally more efficient and particularly suited to handling long-term dependencies in time-series data [89], [90]. Secondly, SSSD^{S4} has reduced the internal representation of shape (batch *dim*, diffusion *dim*, input channel *dim*, time *dim*) in CSDI to shape (batch *dim*, diffusion *dim*, time *dim*). This maps the input channels into the diffusion dimension and thus the model only performs diffusion along the time dimension instead of both the time and input channel dimensions. This corresponds to the common approach when applying diffusion models to image or sound data [68].

At last, SSSD^{S4} only applies the diffusion process to the segments that need to be imputed, as this yields better results than applying the diffusion process to all segments. This change in applying the noise during the diffusion process turns out to be better than the methods in image inpainting [78]. This is different from CSDI, which applies noise to the imputation targets and missing values. The missing values in CSDI are padded with zeros to fix the shape of the inputs. Whereas in SSSD^{S4} no noise is generated for the missing values.

The input data is represented differently from CSDI, leaving out the timestamp. This results in $\{x, m\}$ pairs where x is the value vector and m is the binary mask value.

The reverse process and loss function are formulated the same as in Equation 44 and Equation 45. The authors investigate both types of conditioning, as extra input during training and training unconditionally and including the conditional information during inference.

[7] TDSTF [67]: `https://github.com/PingChang818/TDSTF`

Structured State Space Sequence Model (S4): The S4 architecture is composed of State Space Models (SSM). The SSM [89] is based on a linear state space transition equation, connecting a 1D input sequence $u(t)$ to a 1D output sequence $y(t)$ via a N-dimensional hidden state $x(t)$. This can be formalized as

$$x'(t) = Ax(t) + Bu(t) \text{ and } y(t) = Cx(t) + Du(t), \tag{54}$$

where A, B, C, D area transition matrices. These SSMs can be evaluated efficiently on modern GPUs [89] and can capture long-term dependencies according to the HiPPO theory [91], [92]. By stacking several SSM blocks and additional layers the Structured State Space Sequence model (S4) was created and demonstrated good performance on sequence classification tasks. Structuring the S4 layers in a U-Net-inspired configuration gave rise to SaShiMi [90], which is a generative sequence generative model.

Because the model is inspired by DiffWave and CSDI, the authors propose to compare 3 variations of models to better understand each of the individual components. Namely, SSSD^{S4}, SSSD^{SA}, CSDI^{S4}. SSSD^{S4} is a variant of DiffWave but with S4 components. SSSD^{SA} is a variant of DiffWave but with the SaShiMi instead of S4 components. Lastly, CSDI^{S4} is a variant of CSDI but where the time direction transformer is replaced with an S4 layer.

SSSD^{S4} performs time-series forecasting on the PTB-XL [93], the Solar [73] dataset from Gluon-TS[2] and on the ETTm1[5] dataset from Zhou *et al.* [23]. All data gathering and pre-processing for $SSSD^{S4}$ is explained on their Github[8].

Dataset	H	F	MAE				MSE	
			CSDI	CSDI^{S4}	SSSD^{SA}	SSSD^{S4}	CSDI	SSSD^{S4}
PTB-XL	800	200	0.165±9e−4	0.120±2e−4	**0.087**±8e−3	0.090±3e−3	-	-
Solar	168	24	-	-	-	-	900±61	**503±10.6**
ETTm1	96	24	0.370	-	-	**0.361**	0.354	**0.351**
	48	48	0.546	-	-	**0.479**	0.750	**0.612**
	284	96	0.756	-	-	**0.547**	1.468	**0.538**
	288	288	**0.530**	-	-	0.648	**0.608**	0.797
	384	672	0.891	-	-	**0.783**	0.946	**0.804**

Table 9: SSSD^{S4} Multivariate Forecasting Results [68]

On the PTB-XL dataset, it can be seen that SSSD^{SA} and SSSD^{S4} perform similarly well for forecasting. However, for imputation on PTB-XL the S4 variant outperforms SA all the time. Furthermore, in the visual plots shown by [68] the S4 variant is much more certain in its predictions. The comparison of CSDI [3.3] and SSSD^{S4} on Solar indicates a great improvement, just as on ETTm1. There are however also limitations to the SSSD^{S4} method. As hypothesized in the paper, because of the dimension reduction, it becomes increasingly difficult for the SSSD^{S4} model to reconstruct the original input channels from the internal diffusion channels in situations with a large number of input channels. Alcaraz and Strodthoff [68] see the central strength of SSSD^{S4} to be its ability to capture long-term dependencies along the time direction. As a result, CSDI performs better in situations with many input channels or situations where the relations between input channels are more important than the temporal consistency [68]. The conditioning strategy is, like CSDI, taken from models for image or text data, and not specifically for time series. It has been empirically shown that its long-range prediction performance is inferior to other time-series prediction models [69]. Furthermore, The performance deteriorated due to the boundary disharmony by the semi-supervised learning strategy [69]. The research in SSMs is continuing at a high pace, surpassing Transformers in foundation models such as [94].

3.8 DiffLoad: Uncertainty Quantification in Load Forecasting with Diffusion Model (2023) [16]

The model mentions TimeGrad [63] but in essence is a sequence-to-sequence model [19] that performs the diffusion denoising process on the hidden state in between the encoder and decoder. It is specifically designed

[8] $SSSD^{S4}$ [68]: `https://github.com/AI4HealthUOL/SSSD`

for forecasting tasks of electrical loads and to tackle situations where the temporal data has a distribution shift and contains outliers. The authors make two main contributions, the first one being a new uncertainty quantification method for neural network forecasting utilizing a diffusion-based encoder to concentrate uncertainty in the latent variable before inputting it into the decoder, providing better insights into the epistemic uncertainty. Furthermore, they propose an emission head based on the additive Cauchy distribution to capture the aleatoric uncertainty.

Hidden-state Diffusion: The historical data is encoded with the use of a GRU, resulting in a hidden state h_0. This hidden state is then put through the diffusion and denoising process of an unconditional DDPM [2.1] with $p(h^K) \sim \mathcal{N}(0, \mathbf{I})$ as:

$$p_\theta(h^{0:K}) = p(h^K) \prod_{k=1}^{K} p_\theta(h^{k-1} \mid h^k) \text{ where } p_\theta(h^{k-1}|h^k) = \mathcal{N}(h^{k-1}; \mu_\theta(h^k, k), \sigma_k^2 \mathbf{I}) \tag{55}$$

The theory continuous with Equation 10 where $\sigma_k^2 = \tilde{\beta}_k$ and $\mu_\theta(h^k, k)$ is reparameterized with the noise prediction model explained in Equation 16 with loss function $\mathcal{L}_{\epsilon_\theta}$. The idea behind this diffusion process is to be able to capture and quantify the epistemic uncertainty by generating multiple forecasts, from which an uncertainty can be determined. By estimating the distribution of the hidden state of encoded historical data, the randomness that is left in the estimated hidden state quantifies the model uncertainty.

Cauchy distribution model head: The new hidden state $\hat{h}_0$ is used as the starting hidden state for the decoder and it continuous to produce the forecast by outputting $\hat{\mu}_\phi$ and $\hat{\sigma}_\phi$ for the Cauchy distribution. The Cauchy distribution is used because it is more heavy-tailed compared to the traditional Gaussian distribution. This heavy-tail property makes the Cauchy distribution more robust to outliers and mutation [95], [96]. The Cauchy distribution can be defined by a mean and scale parameter, similar to the Gaussian distribution and is formalized by the authors as:

$$\mu_{\phi(t+1)} = \text{NN}_1(\hat{h}_{t+1}), \tag{56}$$

$$\sigma_{\phi(t+1)} = \text{SoftPlus}\left[\text{NN}_2(\hat{h}_{t+1})\right] \tag{57}$$

Using these parameters, the conditional distribution of the error can be described with the Cauchy distribution $\mathcal{C}$:

$$p_\phi(x_{t+1}|\hat{h}_{h+1} = \mathcal{C}(x_{t+1}; \mu_{\phi(t+1)}, \sigma_{\phi(t+1)}) \tag{58}$$

The loss for the decoder and the diffusion model are combined as such

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{\epsilon_\theta} - \log \sigma_\phi + \log \left(X_{tar} - \mu_\phi\right)^2 + \sigma_\phi^2 \tag{59}$$

During inference the model makes M predictions. These predictions have an uncertainty, because of the probabilistic nature of the diffusion model, which can be described as a Gaussian distribution with σ_θ. This Gaussian distribution represents the epistemic uncertainty. The uncertainties outputted through the Cauchy head can be regarded as the eleatoric uncertainty. Because both uncertainties are α-stable they can be added together to form the final uncertainty of the prediction.

$$\sigma = \underbrace{\sigma_\phi}_{aleatoric} + \underbrace{\sigma_\theta}_{epistemic} \tag{60}$$

Authors continue to evaluate the model on 3 datasets, Global Energy Forecasting (GEF) [97], Building Data Genome Project 2 (BDG2) [98], and Day-ahead electricity demand forecasting (COV) [99]. The authors performed an ablation study to measure the effects of the model as can be seen in Table 10. The o/o variant is without diffusion and Gaussian distribution heads on the decoder. The d/o variant is with diffusion and Gaussian, and d/c is with diffusion and Cauchy distribution. The results show a clear superiority to the model with diffusion and Cauchy distribution.

The authors use the same ϕ for both the encoder and decoder GRUs, it is unclear whether this means that they are using the same parameters. If the models are using the same parameters it would mean that this is not equal to the traditional sequence-to-sequence model. Moreover, the sequence-to-sequence is already becoming old, and better variants with Transformer layers have already been published [27]. There

Dataset	H	F	MAE			CRPS		
			o/o	d/o	d/c	o/o	d/o	d/c
GEF	-	-	119.47	115.37	**110.44**	86.61	**83.07**	83.24
COV	-	-	21844.0	22512.37	**20308.3**	16263.42	16705.65	**15435.79**
BDG2	-	-	12.76	12.38	**11.18**	9.35	9.16	**8.72**

Table 10: DiffLoad Forecasting results [16]

is very little theory about how the epistemic and aleatoric uncertainties are split, and it is not compared with evaluations of their actual uncertainties. The details about how the model captures the different uncertainties are missing. Furthermore, it is not mentioned what the historical and future lengths are making it difficult to reproduce the results. Furthermore, the paper makes the mistake of switching the meaning of aleatoric and epistemic uncertainty often, making it confusing what they mean sometimes. This is the only implementation that performs latent space diffusion [34], however, it is not typically conditioned latent space diffusion because the encoder's output is the "historical condition" for the decoder to generate the forecasts.

3.9 TimeDiff: Non-autoregressive Conditional Diffusion Models for Time Series Prediction (2023) [69]

The TimeDiff model addresses the shortcomings of CSDI and SSSD^{S4} by introducing additional inductive bias in the conditioning module that is tailor-made for time-series. It does this by two conditioning mechanisms: "Future Mixup" and "Autoregressive initialization". Future mixup is based on mixup by Zhang *et al.* [100] and is extended to randomly reveal parts of the ground-truth future predictions during training. Autoregressive initialization works by learning a linear autoregressive model on the data that provides an initial linear guess for the predictions.

Future mixup: The ideal condition generates the forecast is X_{tar} itself, this is not available during inference, but it is during training. Inspired by mixup [100], the revealing of the ground-truth works by creating a conditioning variable $z_{mix} \in \mathbb{R}^{d \times F}$ that is of the same size as X_{tar}. During inference this variable will be filled with a mapping of the historical data as seen in Equation 61. However, since this historical data can have a different size it is mapped through a convolution network $\mathcal{F}$ to end up with the same dimensions.

$$z_{mix} = \mathcal{F}(X_{obs}) \tag{61}$$

Now, during training, randomly selected data points of X_{tar} are also included, as seen in Equation 62, by using a mask variable $m^k \in [0, 1)^{d \times F}$ where each value is sampled from the uniform distribution $[0, 1)$.

$$z_{mix} = m^k \odot \mathcal{F}(X_{obs}) + (1 - m^k) \odot X^0_{tar} \tag{62}$$

Autoregressive initialization: For non-autoregressive models often produce disharmony at the boundaries between masked and observed regions [78]. Shen and Kwok [69] state that for time series forecasting, this translates to disharmony between historical and future segments. To solve this problem they propose the linear autoregressive (AR) model $\mathcal{M}_{ar}$ which will make an initial guess $z_{ar} \in \mathbb{R}^{d \times H}$ for X_{tar}. This model cannot capture the complex non-linear patterns in time-series, but it can approximate the simple ones, such as short-term trends [73]. The initial guess z_{ar} is made by training the following to predict X_{tar}:

$$z_{ar} = \sum_{t=-H}^{0} W_t \odot \bar{X}_t + B \tag{63}$$

where $\bar{X}^0_t \in \mathbb{R}^{d \times F}$ is a matrix that is filled with F vectors of x_t where t is the timestep in *obs*, and $W_t \in \mathbb{R}^{d \times F}$ and $B \in \mathbb{R}^{d \times H}$ are to be trained.

Both of the conditions z_{mix} and $_{ar}$ are combined as:

$$\mathbf{c} = \text{concat}([z_{mix}, z_{ar}]) \in \mathbb{R}^{2d \times F} \tag{64}$$

Furthermore, the objective function is set to reduce the error in predicting the data directly (26) instead of the noise (25). The paper shows empirically that predicting the data performs better and hypothesizes that it is due time-series containing quite some noise inherently.

$$\mathcal{L}_{x_\theta} = \mathbb{E}_{k, X^0_{tar}, \epsilon} \left[||X^0_{tar} - x_\theta(X^k_{tar}, k|\mathbf{c})||^2 \right] \tag{65}$$

Shen and Kwok [69] evaluates its model TimeDiff on nine real world datasets: NordPool [101][9], Caiso [103][9], Traffic[75][10], Electricity[74][10], Weather[87][10], Exchange [73][10], ETTh1[5], ETTm1[5], Wind[6]. For each model the historical length H is selected from $\{96, 192, 720, 1440\}$ by testing which works best on the validation set. The paper does not specify which is used in the experiments as shown in Table 11.

Dataset	H	F	MSE				
			TimeDiff	TimeGrad	CSDI	SSSD	D3VAE
NorPool	...	720	**0.665**	1.152	1.011	0.872	0.745
Caiso	...	720	**0.136**	0.258	0.253	0.195	0.241
Weather	...	672	**0.311**	0.392	0.356	0.349	0.375
ETTm1	...	192	**0.336**	0.874	0.529	0.464	0.362
Wind	...	192	**0.896**	1.209	1.066	1.188	1.118
Traffic	...	168	**0.564**	1.745	N/A	0.642	0.928
Electricity	...	168	**0.193**	0.736	N/A	0.255	0.286
ETTh1	...	168	**0.407**	0.993	0.497	0.726	0.504
Exchange	...	14	**0.018**	0.079	0.077	0.061	0.200

Table 11: TimeDiff Multivariate Forecasting results [69]

Furthermore, the authors perform an ablation study on the future mixup and AR, resulting in a clear win if both are implemented. They even went as far as implementing these features on CSDI [3.3] and SSSD^{S4} [3.7] which also showed that by combining the features resulted in the best performance on ETTh1[5] and ETTm1[5]. CSDI with mixup and AR fell short on performance compared to TimeDiff, while SSSD^{S4} achieves comparable results. Besides the better conditioning, the performance of using the data prediction model is also shown across the Caiso[9], Electricity[10], Exchange[10] and Etth1[5] datasets. However, unfortunately, the authors do not perform the experiments of SSSD^{S4} with mixup, AR, and a data prediction model. By also including the data prediction model it could have been even better. Furthermore, the implementation still struggles with a large amount of input features learning the multivariate dependencies. The paper considers using graph neural networks to capture these dependencies. The paper mentions it has not solved the open question of how to design an efficient denoising network and conditioning network in time series diffusion models.

3.10 TSDiff: Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting (2023) [70]

TSDiff is a model that is trained unconditionally and employs the diffusion guidance, explained in subsubsection 2.3.2, to condition univariate forecasts during inference. Based on models that use guidance [30], [104], the authors propose a self-guidance mechanism that allows the conditioning during inference without an auxiliary network. This makes it applicable to various downstream tasks. It specifically investigates the usability of task-agnostic unconditional diffusion models for forecasting tasks.

The authors have based the architecture on that of SSSD^{S4} with a modification on how the time-series is represented. Since the model is designed for univariate time-series it has a whole input dimension left to fill,

[9] DEPTS [102]: `https://github.com/weifantt/DEPTS`
[10] AutoFormer [24]: `https://github.com/thuml/Autoformer`

thus the model sort of folds the time-series in C sections resulting in an input $x \in \mathbb{R}^{L \times C}$, replacing the d as explained in subsubsection 1.1.1. L is the length of the time-series $H + F$, and additional historical data can be included in the other $C - 1$ 'folds'. The target series becomes x_{tar} and the observations are x_{obs}, the small x denoting the univariate series. It is important to understand that the model will be trained on $X_{obs+tar}$, as the intuition behind the self-guidance is that a model that can generate $obs + tar$ sequences, should also be able to reasonably generate sequences tar while being guided towards obs. The authors propose two observation self-guidance approaches, namely the mean square self-guidance and the quantile self-guidance.

Mean Square Self-Guidance: Following from Equation 32, the guidance term $p(\mathbf{c}|x^k_{tar})$ parameterized and modeled as a multivariate Gaussian distribution

$$p_\theta(\mathbf{c}|x^k) = \mathcal{N}(x_{obs}|f_\theta(x^k, k), \mathbf{I}) \tag{66}$$

where f_θ is a function that approximates x^0 given the diffused time-series x^k. Now this function can be expressed using the noise prediction model (16) by rearranging Equation 8 as

$$f_\theta(x^k_{tar}, k) = \frac{x^k_{tar} - \sqrt{1 - \alpha_k}\epsilon_\theta(x^k_{tar}, k)}{\sqrt{\alpha_k}} \tag{67}$$

With this function, the MSE loss is calculated as $\text{MSE}(f_\theta(x^k_{tar}, k), x_{obs})$ then the gradients are calculated against x^k_{tar}. Thus by guiding the denoising steps such that it minimizes the MSE of the observed data in the generated series, it also guides it toward the forecasted data.

Quantile Self-Guidance: This guidance goes further to minimize the quantile loss, also known the pinball loss. This loss is based on the CRPS metric, and thus takes all quantiles of the distribution into account, making it more refined than the Mean Square Self-Guidance. So instead of minimizing the MSE, the quantile self-guidance minimizes the quantile loss.

$$\mathcal{L}_{quantile} = \max\{\kappa \cdot (x_{obs} - f_\theta(x^k, k)), (\kappa - 1) \cdot (x_{obs} - f_\theta(x^k, k))\} \tag{68}$$

where $\kappa \in (0, 1)$ specifies the quantile level. In practice Kollovieh *et al.* [70] used multiple evenly spaced quantile levels, based on the number of forecasts. Because the uncertainty is estimated by forecasting multiple times.

Kollovieh *et al.* [70] perform forecasting experiments on eight univariate time series datasets available through GluonTS[2]: Solar[73], Electricity [74], Traffic [75], Exchange [73], M4 [105], UberTLC[76][11], KDD-Cup[106], and Wikipedia[77]. The evaluation method used is CRPS (3) and the distributions were gathered by forecasting 100 times.

Dataset	H	F	CRPS			
			CSDI	TSDiff-Cond	TSDiff-MS	TSDiff-Q
Solar	336	24	0.352 ± 0.005	$\mathbf{0.338 \pm 0.014}$	0.391 ± 0.003	0.358 ± 0.020
Electricity	336	24	0.054 ± 0.000	0.050 ± 0.002	0.062 ± 0.001	$\mathbf{0.049 \pm 0.000}$
Traffic	336	24	0.159 ± 0.002	$\mathbf{0.094 \pm 0.003}$	0.116 ± 0.001	0.098 ± 0.002
Exchange	360	30	0.033 ± 0.014	0.013 ± 0.002	0.018 ± 0.003	$\mathbf{0.011 \pm 0.001}$
M4	312	48	0.040 ± 0.003	0.039 ± 0.006	0.045 ± 0.000	$\mathbf{0.036 \pm 0.001}$
UberTLC	336	24	0.206 ± 0.002	0.172 ± 0.008	0.183 ± 0.007	$\mathbf{0.172 \pm 0.005}$
KDDCup	312	48	0.318 ± 0.002	0.754 ± 0.007	0.325 ± 0.028	$\mathbf{0.311 \pm 0.026}$
Wikipedia	360	30	0.289 ± 0.017	$\mathbf{0.218 \pm 0.010}$	0.257 ± 0.001	0.221 ± 0.001

Table 12: TSDiff Univariate Forecasting results [70]

Unfortunately, the model is designed and evaluated for univariate time-series. The authors mention that the model can be extended with a Transformer [20] model operating across the feature dimensions after the

[11] FiveThirtyEight: `https://github.com/fivethirtyeight/uber-tlc-foil-response`

S4 layer. However, this would limit the current implementation of extra historical information in the folds. C will be replaced with d, reducing the possible historical information. The model using quantile guidance performs well compared to CSDI [3.3], however, it is rather unfortunate the paper does not compare the results with SSSD^{S4} [3.7] on which the model is based. The paper also extends the usage of the guidance for other tasks such as refining existing predictions of other forecasting models or training models on synthetic data. To conclude, the model is not as strong for forecasting as other conditioned models but the guidance is a novel approach and deserves more research attention concerning time-series.

4 Discussion

The surveyed papers go into multiple interesting directions tackling time-series forecasting through the use of diffusion denoising processes. Most of the models provide probabilistic forecasting, thereby giving more insights in the uncertainty of the predictions. Notably, the D^{3}VAE [66] and DiffLoad [16] papers advance this field by dissecting both epistemic and aleatoric uncertainties within predictions. DiffLoad does so by performing the diffusion process on a latent space variable of the historical data [16]. Extending on the encoding of historical data, a progression is observed in the methods, evolving from RNNs to Transformers, and ultimately to S4 layers, with the latter demonstrating superior performance in foundational models like [94]. Specifically The SSSD^{S4} [68] and TSDiff [70] models experiment with this type of layer. Additionally, TSDiff reveals that unconditional forecasting with guidance can be as good as conditional forecasting and can bring more benefits for other time-series related tasks. With respect to the type of prediction model used in the reverse process, TimeDiff [69] empirically shows the superiority of data prediction models over noise prediction models for forecasting. This is only a recent revelation and not mentioned in any of the other Furthermore, TimeDiff makes a good comparison with other state-of-the-art transformer forecasting models and shows the superiority of diffusion models by introducing additional indicative bias in the conditioning module. Furthermore, throughout the papers, comparative evaluations utilizing the metrics CRPS_{sum} (5) or MSE (1) on diverse datasets like Electricity [74], Traffic [75], or those from Gluon-TS[2], have made the comparisons more straightforward and accessible. Lastly, the only implementations that make use of the more generalized diffusion process with the use of SDEs are ScoreGrad [61] and CSPD [65]. With ScoreGrad going as far as to also implement the ODE method enhancing prediction speeds by up to 4.9 times. Overall, this literature survey provides an in-depth overview of 11 seminal papers in the realm of diffusion-based time-series forecasting models. It serves as an invaluable starting point for future researchers delving into this domain, offering a comprehensive understanding of the current state-of-the-art methodologies and their evolutionary trajectories.

5 Future Works

Future research directions should include the implementation of diffusion models using the ODE method to enhance prediction speeds without compromising accuracy. The adoption of encoder-decoder frameworks for latent space diffusion, moving beyond mere historical data encoding, is also recommended as this has show to improve prediction qualities [34]. Continued investigation into the S4 layers [90] is encouraged, given their promising results in surpassing Transformers in foundation models [94]. Overall, the integrating of various approaches, such as TimeDiff [69], DiffLoad [16], SSSD^{S4} [68], TDSTF [67], and DSPD-GP [65], would be beneficial. This integration could focus on combining extra conditioning information 3.9, latent space diffusion 3.8, the use of SSMs 3.7, efficient data representation 3.6, and modeling time-series as continuous functions with time-dependent noise 3.4. Attention should also be given to long-term multivariate time-series forecasting, decomposing epistemic and aleatoric uncertainty, and evaluating models with both data and noise prediction models to establish when each approach is more advantageous.

References

[1] D. Baidoo-Anu and L. Owusu Ansah, "Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning," en,

SSRN Electronic Journal, 2023, ISSN: 1556-5068. DOI: 10.2139/ssrn.4337484. [Online]. Available: https://www.ssrn.com/abstract=4337484 (visited on 11/07/2023) (cit. on p. 1).

[2] J. Qadir, "Engineering Education in the Era of ChatGPT: Promise and Pitfalls of Generative AI for Education," en, in *2023 IEEE Global Engineering Education Conference (EDUCON)*, Kuwait, Kuwait: IEEE, May 2023, pp. 1–9, ISBN: 9798350399431. DOI: 10.1109/EDUCON54358.2023.10125121. [Online]. Available: https://ieeexplore.ieee.org/document/10125121/ (visited on 11/07/2023) (cit. on p. 1).

[3] W. M. Lim, A. Gunasekara, J. L. Pallant, J. I. Pallant, and E. Pechenkina, "Generative AI and the future of education: Ragnarök or reformation? A paradoxical perspective from management educators," *The International Journal of Management Education*, vol. 21, no. 2, p. 100 790, Jul. 2023, ISSN: 1472-8117. DOI: 10.1016/j.ijme.2023.100790. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1472811723000289 (visited on 11/07/2023) (cit. on p. 1).

[4] S. Noy and W. Zhang, "Experimental Evidence on the Productivity Effects of Generative Artificial Intelligence," en, Mar. 2023 (cit. on p. 1).

[5] E. Brynjolfsson, D. Li, and L. R. Raymond, *Generative AI at Work*, Working Paper, Apr. 2023. DOI: 10.3386/w31161. [Online]. Available: https://www.nber.org/papers/w31161 (visited on 11/07/2023) (cit. on p. 1).

[6] Y. K. Dwivedi, N. Kshetri, L. Hughes, *et al.*, "Opinion Paper: "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy," *International Journal of Information Management*, vol. 71, p. 102 642, Aug. 2023, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2023.102642. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0268401223000233 (visited on 11/07/2023) (cit. on p. 1).

[7] C. Luo, *Understanding Diffusion Models: A Unified Perspective*, arXiv:2208.11970 [cs], Aug. 2022. [Online]. Available: http://arxiv.org/abs/2208.11970 (visited on 10/17/2023) (cit. on pp. 1, 4, 6, 8).

[8] R. B. Penfold and F. Zhang, "Use of Interrupted Time Series Analysis in Evaluating Health Care Quality Improvements," *Academic Pediatrics*, Quality Improvement in Pediatric Health Care, vol. 13, no. 6, Supplement, S38–S44, Nov. 2013, 674 citations (Crossref) [2023-11-08], ISSN: 1876-2859. DOI: 10.1016/j.acap.2013.08.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876285913002106 (visited on 11/08/2023) (cit. on p. 2).

[9] C. Bui, N. Pham, A. Vo, A. Tran, A. Nguyen, and T. Le, "Time Series Forecasting for Healthcare Diagnosis and Prognostics with the Focus on Cardiovascular Diseases," en, in *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6)*, T. Vo Van, T. A. Nguyen Le, and T. Nguyen Duc, Eds., ser. IFMBE Proceedings, 11 citations (Crossref) [2023-11-08], Singapore: Springer, 2018, pp. 809–818, ISBN: 978-981-10-4361-1. DOI: 10.1007/978-981-10-4361-1_138 (cit. on p. 2).

[10] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, "Recurrent Neural Networks for Multivariate Time Series with Missing Values," en, *Scientific Reports*, vol. 8, no. 1, p. 6085, Apr. 2018, 858 citations (Crossref) [2023-11-08] Number: 1 Publisher: Nature Publishing Group, ISSN: 2045-2322. DOI: 10.1038/s41598-018-24271-9. [Online]. Available: https://www.nature.com/articles/s41598-018-24271-9 (visited on 11/08/2023) (cit. on p. 2).

[11] S. Kaushik, A. Choudhury, P. K. Sheron, *et al.*, "AI in Healthcare: Time-Series Forecasting Using Statistical, Neural, and Ensemble Architectures," *Frontiers in Big Data*, vol. 3, 2020, 64 citations (Crossref) [2023-11-08], ISSN: 2624-909X. DOI: 10.3389/fdata.2020.00004. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fdata.2020.00004 (visited on 11/08/2023) (cit. on p. 2).

[12] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons & Fractals*, vol. 135, p. 109 864, Jun. 2020, 598 citations (Crossref) [2023-11-08], ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.109864. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077920302642 (visited on 11/07/2023) (cit. on p. 2).

[13] A. Zeroual, F. Harrou, A. Dairi, and Y. Sun, "Deep learning methods for forecasting COVID-19 time-Series data: A Comparative study," *Chaos, Solitons & Fractals*, vol. 140, p. 110 121, Nov. 2020, 285 citations (Crossref) [2023-11-08], ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.110121. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S096007792030518X (visited on 11/07/2023) (cit. on p. 2).

[14] C. Deb, F. Zhang, J. Yang, S. E. Lee, and K. W. Shah, "A review on time series forecasting techniques for building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 74, pp. 902–924, Jul. 2017, ISSN: 1364-0321. DOI: 10.1016/j.rser.2017.02.085. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032117303155 (visited on 11/08/2023) (cit. on p. 2).

[15] J.-S. Chou and D.-S. Tran, "Forecasting energy consumption time series using machine learning techniques based on usage patterns of residential householders," *Energy*, vol. 165, pp. 709–726, Dec. 2018, ISSN: 0360-5442. DOI: 10.1016/j.energy.2018.09.144. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360544218319145 (visited on 11/08/2023) (cit. on p. 2).

[16] Z. Wang, Q. Wen, C. Zhang, L. Sun, and Y. Wang, *DiffLoad: Uncertainty Quantification in Load Forecasting with Diffusion Model*, arXiv:2306.01001 [cs, stat], May 2023. [Online]. Available: http://arxiv.org/abs/2306.01001 (visited on 10/19/2023) (cit. on pp. 2, 9, 18, 20, 23).

[17] M. Lippi, M. Bertini, and P. Frasconi, "Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 871–882, Jun. 2013, Conference Name: IEEE Transactions on Intelligent Transportation Systems, ISSN: 1558-0016. DOI: 10.1109/TITS.2013.2247040. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6482260 (visited on 11/08/2023) (cit. on p. 2).

[18] D. Pavlyuk, "Short-term Traffic Forecasting Using Multivariate Autoregressive Models," *Procedia Engineering*, RelStat-2016: Proceedings of the 16th International Scientific Conference Reliability and Statistics in Transportation and Communication October 19-22, 2016. Transport and Telecommunication Institute, Riga, Latvia, vol. 178, pp. 57–66, Jan. 2017, ISSN: 1877-7058. DOI: 10.1016/j.proeng.2017.01.062. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705817300620 (visited on 11/08/2023) (cit. on p. 2).

[19] I. Sutskever, O. Vinyals, and Q. V. Le, *Sequence to Sequence Learning with Neural Networks*, arXiv:1409.3215 [cs], Dec. 2014. DOI: 10.48550/arXiv.1409.3215. [Online]. Available: http://arxiv.org/abs/1409.3215 (visited on 11/22/2023) (cit. on pp. 2, 18).

[20] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 09/27/2023) (cit. on pp. 2, 9, 11, 12, 16, 17, 22).

[21] C. J. Murray, N. Du Bois, L. Hollywood, and D. Coyle, *State-of-The-Art Deep Learning Models are Superior for Time Series Forecasting and are Applied Optimally with Iterative Prediction Methods*, en, SSRN Scholarly Paper, Rochester, NY, Feb. 2023. DOI: 10.2139/ssrn.4361707. [Online]. Available: https://papers.ssrn.com/abstract=4361707 (visited on 11/22/2023) (cit. on p. 2).

[22] N. Nguyen and B. Quanz, "Temporal Latent Auto-Encoder: A Method for Probabilistic Multivariate Time Series Forecasting," en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 9117–9125, May 2021, Number: 10, ISSN: 2374-3468. DOI: 10.1609/aaai.v35i10.17101. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17101 (visited on 10/03/2023) (cit. on p. 2).

[23] H. Zhou, S. Zhang, J. Peng, *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," en, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11 106–11 115, May 2021, Number: 12, ISSN: 2374-3468. DOI: 10.1609/aaai.v35i12.17325. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/17325 (visited on 10/30/2023) (cit. on pp. 2, 15, 18).

[24] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*, arXiv:2106.13008 [cs], Jan. 2022. DOI: 10.48550/arXiv.2106.13008. [Online]. Available: http://arxiv.org/abs/2106.13008 (visited on 10/31/2023) (cit. on pp. 2, 21).

[25] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin, *FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting*, arXiv:2201.12740 [cs, stat], Jun. 2022. DOI: 10.48550/arXiv.2201.12740. [Online]. Available: http://arxiv.org/abs/2201.12740 (visited on 11/22/2023) (cit. on p. 2).

[26] Y. Zhang and J. Yan, "Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting," en, Sep. 2022. [Online]. Available: https://openreview.net/forum?id=vSVLM2j9eie (visited on 11/22/2023) (cit. on p. 2).

[27] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*, arXiv:2211.14730 [cs], Mar. 2023. DOI: 10.48550/arXiv.2211.14730. [Online]. Available: http://arxiv.org/abs/2211.14730 (visited on 11/22/2023) (cit. on pp. 2, 19).

[28] D. Foster, *Generative Deep Learning*, en. "O'Reilly Media, Inc.", Jun. 2022, Google-Books-ID: BEq8EAAAQBAJ, ISBN: 978-1-09-813414-3 (cit. on p. 2).

[29] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," en, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, Nov. 2022, ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2021.3116668. [Online]. Available: https://ieeexplore.ieee.org/document/9555209/ (visited on 11/07/2023) (cit. on p. 2).

[30] P. Dhariwal and A. Nichol, "Diffusion Models Beat GANs on Image Synthesis," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 8780–8794. [Online]. Available: https://papers.nips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html (visited on 10/18/2023) (cit. on pp. 2, 21).

[31] L. Yang, Z. Zhang, Y. Song, *et al.*, *Diffusion Models: A Comprehensive Survey of Methods and Applications*, arXiv:2209.00796 [cs], Mar. 2023. [Online]. Available: http://arxiv.org/abs/2209.00796 (visited on 09/18/2023) (cit. on p. 2).

[32] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion Models in Vision: A Survey," en, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 9, pp. 10 850–10 869, Sep. 2023, ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2023.3261988. [Online]. Available: https://ieeexplore.ieee.org/document/10081412/ (visited on 11/07/2023) (cit. on p. 2).

[33] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, *Cascaded Diffusion Models for High Fidelity Image Generation*, arXiv:2106.15282 [cs], Dec. 2021. DOI: 10.48550/arXiv.2106.15282. [Online]. Available: http://arxiv.org/abs/2106.15282 (visited on 11/08/2023) (cit. on pp. 2, 7).

[34] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-Resolution Image Synthesis with Latent Diffusion Models*, arXiv:2112.10752 [cs], Apr. 2022. [Online]. Available: http://arxiv.org/abs/2112.10752 (visited on 10/24/2023) (cit. on pp. 2, 4, 7, 20, 23).

[35] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, "Structured Denoising Diffusion Models in Discrete State-Spaces," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 17 981–17 993. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/958c530554f78bcd8e97125b70e6973d-Abstract.html (visited on 11/08/2023) (cit. on p. 2).

[36] S. Gong, M. Li, J. Feng, Z. Wu, and L. Kong, *DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models*, arXiv:2210.08933 [cs], Feb. 2023. DOI: 10.48550/arXiv.2210.08933. [Online]. Available: http://arxiv.org/abs/2210.08933 (visited on 10/16/2023) (cit. on p. 2).

[37] X. L. Li, J. Thickstun, I. Gulrajani, P. Liang, and T. B. Hashimoto, *Diffusion-LM Improves Controllable Text Generation*, arXiv:2205.14217 [cs], May 2022. DOI: 10.48550/arXiv.2205.14217. [Online]. Available: http://arxiv.org/abs/2205.14217 (visited on 11/08/2023) (cit. on p. 2).

[38] P. Yu, S. Xie, X. Ma, *et al.*, *Latent Diffusion Energy-Based Model for Interpretable Text Modeling*, arXiv:2206.05895 [cs], Oct. 2023. DOI: 10.48550/arXiv.2206.05895. [Online]. Available: http://arxiv.org/abs/2206.05895 (visited on 11/08/2023) (cit. on p. 2).

[39] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, *DiffWave: A Versatile Diffusion Model for Audio Synthesis*, en, Sep. 2020. [Online]. Available: https://arxiv.org/abs/2009.09761v3 (visited on 10/09/2023) (cit. on pp. 2, 9, 10, 12, 17).

[40] D. Yang, J. Yu, H. Wang, *et al.*, "Diffsound: Discrete Diffusion Model for Text-to-Sound Generation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 1720–1733, 2023, 3 citations (Crossref) [2023-10-17] Conference Name: IEEE/ACM Transactions on Audio, Speech, and Language Processing, ISSN: 2329-9304. DOI: 10.1109/TASLP.2023.3268730. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10112585 (visited on 10/16/2023) (cit. on p. 2).

[41] R. Yang, P. Srivastava, and S. Mandt, *Diffusion Probabilistic Modeling for Video Generation*, arXiv:2203.09481 [cs, stat], Dec. 2022. DOI: 10.48550/arXiv.2203.09481. [Online]. Available: http://arxiv.org/abs/2203.09481 (visited on 11/08/2023) (cit. on p. 2).

[42] W. Harvey, S. Naderiparizi, V. Masrani, C. Weilbach, and F. Wood, *Flexible Diffusion Modeling of Long Videos*, arXiv:2205.11495 [cs], Dec. 2022. DOI: 10.48550/arXiv.2205.11495. [Online]. Available: http://arxiv.org/abs/2205.11495 (visited on 11/08/2023) (cit. on p. 2).

[43] J. Ho, W. Chan, C. Saharia, *et al.*, *Imagen Video: High Definition Video Generation with Diffusion Models*, arXiv:2210.02303 [cs], Oct. 2022. DOI: 10.48550/arXiv.2210.02303. [Online]. Available: http://arxiv.org/abs/2210.02303 (visited on 10/16/2023) (cit. on p. 2).

[44] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet, *Video Diffusion Models*, arXiv:2204.03458 [cs], Jun. 2022. DOI: 10.48550/arXiv.2204.03458. [Online]. Available: http://arxiv.org/abs/2204.03458 (visited on 11/08/2023) (cit. on p. 2).

[45] L. Lin, Z. Li, R. Li, X. Li, and J. Gao, *Diffusion Models for Time Series Applications: A Survey*, arXiv:2305.00624 [cs], Apr. 2023. DOI: 10.48550/arXiv.2305.00624. [Online]. Available: http://arxiv.org/abs/2305.00624 (visited on 09/18/2023) (cit. on p. 2).

[46] H. Koo and T. E. Kim, *A Comprehensive Survey on Generative Diffusion Models for Structured Data*, arXiv:2306.04139 [cs], Jul. 2023. DOI: 10.48550/arXiv.2306.04139. [Online]. Available: http://arxiv.org/abs/2306.04139 (visited on 09/18/2023) (cit. on p. 2).

[47] R. L. Winkler, J. Muñoz, J. L. Cervera, *et al.*, "Scoring rules and the evaluation of probabilities," en, *Test*, vol. 5, no. 1, pp. 1–60, Jun. 1996, 158 citations (Crossref) [2023-10-17], ISSN: 1863-8260. DOI: 10.1007/BF02562681. [Online]. Available: https://doi.org/10.1007/BF02562681 (visited on 10/09/2023) (cit. on p. 3).

[48] A. Jordan, F. Krüger, and S. Lerch, "Evaluating Probabilistic Forecasts with scoringRules," en, *Journal of Statistical Software*, vol. 90, pp. 1–37, Aug. 2019, ISSN: 1548-7660. DOI: 10.18637/jss.v090.i12. [Online]. Available: https://doi.org/10.18637/jss.v090.i12 (visited on 11/15/2023) (cit. on p. 3).

[49] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://papers.nips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html (visited on 09/26/2023) (cit. on pp. 4, 5, 10, 11, 14).

[50] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-Based Generative Modeling through Stochastic Differential Equations*, arXiv:2011.13456 [cs, stat], Feb. 2021. DOI: 10.48550/arXiv.2011.13456. [Online]. Available: http://arxiv.org/abs/2011.13456 (visited on 10/19/2023) (cit. on pp. 4, 6–8, 11).

[51] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," en, in *Proceedings of the 32nd International Conference on Machine Learning*, ISSN: 1938-7228, PMLR, Jun. 2015, pp. 2256–2265. [Online]. Available: https://proceedings.mlr.press/v37/sohl-dickstein15.html (visited on 10/16/2023) (cit. on pp. 5, 8).

[52] Y. Benny and L. Wolf, *Dynamic Dual-Output Diffusion Models*, arXiv:2203.04304 [cs, eess], Mar. 2022. DOI: 10.48550/arXiv.2203.04304. [Online]. Available: http://arxiv.org/abs/2203.04304 (visited on 10/17/2023) (cit. on pp. 5, 6).

[53] Z. Chang, G. A. Koulieris, and H. P. H. Shum, *On the Design Fundamentals of Diffusion Models: A Survey*, arXiv:2306.04542 [cs], Oct. 2023. [Online]. Available: http://arxiv.org/abs/2306.04542 (visited on 11/08/2023) (cit. on p. 5).

[54] B. D. O. Anderson, "Reverse-time diffusion equation models," *Stochastic Processes and their Applications*, vol. 12, no. 3, pp. 313–326, May 1982, ISSN: 0304-4149. DOI: 10.1016/0304-4149(82)90051-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304414982900515 (visited on 10/20/2023) (cit. on p. 6).

[55] A. Hyvärinen, "Estimation of Non-Normalized Statistical Models by Score Matching," *Journal of Machine Learning Research*, vol. 6, no. 24, pp. 695–709, 2005, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v6/hyvarinen05a.html (visited on 11/14/2023) (cit. on p. 6).

[56] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced Score Matching: A Scalable Approach to Density and Score Estimation," en, in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, ISSN: 2640-3498, PMLR, Aug. 2020, pp. 574–584. [Online]. Available: https://proceedings.mlr.press/v115/song20a.html (visited on 11/14/2023) (cit. on p. 6).

[57] T. Pang, K. Xu, C. Li, Y. Song, S. Ermon, and J. Zhu, *Efficient Learning of Generative Models via Finite-Difference Score Matching*, arXiv:2007.03317 [cs, stat], Nov. 2020. DOI: 10.48550/arXiv.2007.03317. [Online]. Available: http://arxiv.org/abs/2007.03317 (visited on 11/14/2023) (cit. on p. 6).

[58] B. J. Holzschuh, S. Vegetti, and N. Thuerey, *Score Matching via Differentiable Physics*, arXiv:2301.10250 [physics], Jan. 2023. [Online]. Available: http://arxiv.org/abs/2301.10250 (visited on 11/13/2023) (cit. on p. 6).

[59] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural Ordinary Differential Equations," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://papers.nips.cc/paper_files/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html (visited on 10/23/2023) (cit. on p. 7).

[60] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, *FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models*, arXiv:1810.01367 [cs, stat], Oct. 2018. DOI: 10.48550/arXiv.1810.01367. [Online]. Available: http://arxiv.org/abs/1810.01367 (visited on 11/09/2023) (cit. on p. 7).

[61] T. Yan, H. Zhang, T. Zhou, Y. Zhan, and Y. Xia, *ScoreGrad: Multivariate Probabilistic Time Series Forecasting with Continuous Energy-based Generative Models*, arXiv:2106.10121 [cs, stat], Jun. 2021. DOI: 10.48550/arXiv.2106.10121. [Online]. Available: http://arxiv.org/abs/2106.10121 (visited on 09/25/2023) (cit. on pp. 7, 9, 11, 12, 23).

[62] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, *WaveNet: A Generative Model for Raw Audio*, arXiv:1609.03499 [cs], Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.03499 (visited on 10/09/2023) (cit. on pp. 9–11).

[63] K. Rasul, C. Seward, I. Schuster, and R. Vollgraf, *Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting*, arXiv:2101.12072 [cs], Feb. 2021. DOI: 10.48550/arXiv.2101.12072. [Online]. Available: http://arxiv.org/abs/2101.12072 (visited on 10/24/2023) (cit. on pp. 9–13, 15, 18).

[64] Y. Tashiro, J. Song, Y. Song, and S. Ermon, "CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., Nov. 2021, pp. 24 804–24 816. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/cfe8504bda37b575c70ee1a8276f3486-Abstract.html (visited on 09/27/2023) (cit. on pp. 9, 12, 13, 16, 17).

[65] M. Biloš, K. Rasul, A. Schneider, Y. Nevmyvaka, and S. Günnemann, *Modeling Temporal Data as Continuous Functions with Stochastic Process Diffusion*, arXiv:2211.02590 [cs] version: 1, Nov. 2022. DOI: 10.48550/arXiv.2211.02590. [Online]. Available: http://arxiv.org/abs/2211.02590 (visited on 10/24/2023) (cit. on pp. 9, 13, 14, 23).

[66] Y. Li, X. Lu, Y. Wang, and D. Dou, "Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement," en, *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 009–23 022, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/91a85f3fb8f570e6be52b333b5ab017a-Abstract-Conference.html (visited on 10/26/2023) (cit. on pp. 9, 14–16, 23).

[67] P. Chang, H. Li, S. F. Quan, *et al.*, *TDSTF: Transformer-based Diffusion probabilistic model for Sparse Time series Forecasting*, arXiv:2301.06625 [cs], Mar. 2023. [Online]. Available: http://arxiv.org/abs/2301.06625 (visited on 09/28/2023) (cit. on pp. 9, 13, 16, 17, 23).

[68] J. M. L. Alcaraz and N. Strodthoff, *Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models*, arXiv:2208.09399 [cs, stat], May 2023. DOI: 10.48550/arXiv.2208.09399. [Online]. Available: http://arxiv.org/abs/2208.09399 (visited on 09/19/2023) (cit. on pp. 9, 17, 18, 23).

[69] L. Shen and J. Kwok, *Non-autoregressive Conditional Diffusion Models for Time Series Prediction*, arXiv:2306.05043 [cs], Jun. 2023. DOI: 10.48550/arXiv.2306.05043. [Online]. Available: http://arxiv.org/abs/2306.05043 (visited on 09/28/2023) (cit. on pp. 9, 11–13, 15, 16, 18, 20, 21, 23).

[70] M. Kollovieh, A. F. Ansari, M. Bohlke-Schneider, J. Zschiegner, H. Wang, and Y. Wang, *Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting*, arXiv:2307.11494 [cs, stat], Jul. 2023. DOI: 10.48550/arXiv.2307.11494. [Online]. Available: http://arxiv.org/abs/2307.11494 (visited on 10/05/2023) (cit. on pp. 9, 21–23).

[71] D. Salinas, M. Bohlke-Schneider, L. Callot, R. Medico, and J. Gasthaus, "High-dimensional multivariate forecasting with low-rank Gaussian Copula Processes," en, *Advances in Neural Information Processing Systems*, vol. 32, 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/0b105cf1504c4e241fcc6d519ea962fb-Abstract.html?ref=https://githubhelp.com (visited on 09/27/2023) (cit. on p. 10).

[72] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, *et al.*, *GluonTS: Probabilistic Time Series Models in Python*, arXiv:1906.05264 [cs, stat], Jun. 2019. DOI: 10.48550/arXiv.1906.05264. [Online]. Available: http://arxiv.org/abs/1906.05264 (visited on 10/30/2023) (cit. on p. 10).

[73] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, *Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks*, arXiv:1703.07015 [cs], Apr. 2018. DOI: 10.48550/arXiv.1703.07015. [Online]. Available: http://arxiv.org/abs/1703.07015 (visited on 10/18/2023) (cit. on pp. 10, 12, 15, 18, 20–22).

[74] A. Trindade, *ElectricityLoadDiagrams20112014*, 2015. DOI: 10.24432/C58C86. [Online]. Available: https://archive.ics.uci.edu/dataset/321 (visited on 10/30/2023) (cit. on pp. 10, 15, 21–23).

[75] M. Cuturi, *PEMS-SF*, 2011. DOI: 10.24432/C52G70. [Online]. Available: https://archive.ics.uci.edu/dataset/204 (visited on 10/30/2023) (cit. on pp. 10, 15, 21–23).

[76] NYC Taxi and Limousine Commission, *TLC Trip Record Data - TLC*, 2015. [Online]. Available: https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page (visited on 10/30/2023) (cit. on pp. 10, 22).

[77] J. Gasthaus, K. Benidis, Y. Wang, *et al.*, "Probabilistic Forecasting with Spline Quantile Function RNNs," en, in *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, ISSN: 2640-3498, PMLR, Apr. 2019, pp. 1901–1910. [Online]. Available: https://proceedings.mlr.press/v89/gasthaus19a.html (visited on 10/30/2023) (cit. on pp. 10, 22).

[78] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, *RePaint: Inpainting using Denoising Diffusion Probabilistic Models*, arXiv:2201.09865 [cs], Aug. 2022. DOI: 10.48550/arXiv.2201.09865. [Online]. Available: http://arxiv.org/abs/2201.09865 (visited on 10/16/2023) (cit. on pp. 13, 17, 20).

[79] T. Gneiting and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, Mar. 2007, 2669 citations (Crossref) [2023-10-17] Publisher: Taylor & Francis _eprint: https://doi.org/10.1198/016214506000001437, ISSN: 0162-1459. DOI: 10.1198/016214506000001437. [Online]. Available: https://doi.org/10.1198/016214506000001437 (visited on 10/05/2023) (cit. on p. 14).

[80] A. Vahdat and J. Kautz, "NVAE: A Deep Hierarchical Variational Autoencoder," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 19 667–19 679. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html (visited on 11/27/2023) (cit. on pp. 14, 15).

[81] S. Watanabe, "Information Theoretical Analysis of Multivariate Correlation," *IBM Journal of Research and Development*, vol. 4, no. 1, pp. 66–82, Jan. 1960, Conference Name: IBM Journal of Research and Development, ISSN: 0018-8646. DOI: 10.1147/rd.41.0066. [Online]. Available: https://ieeexplore.ieee.org/document/5392532 (visited on 11/28/2023) (cit. on p. 15).

[82] H. Kim and A. Mnih, "Disentangling by Factorising," en, in *Proceedings of the 35th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 2018, pp. 2649–2658. [Online]. Available: https://proceedings.mlr.press/v80/kim18b.html (visited on 11/28/2023) (cit. on p. 15).

[83] Y. Li, Z. Chen, D. Zha, *et al.*, *Learning Disentangled Representations for Time Series*, arXiv:2105.08179 [cs], May 2021. DOI: 10.48550/arXiv.2105.08179. [Online]. Available: http://arxiv.org/abs/2105.08179 (visited on 12/08/2023) (cit. on p. 15).

[84] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," eng, *The Behavioral and Brain Sciences*, vol. 40, e253, Jan. 2017, ISSN: 1469-1825. DOI: 10.1017/S0140525X16001837 (cit. on p. 15).

[85] I. Higgins, L. Matthey, A. Pal, *et al.*, "Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework," en, Nov. 2016. [Online]. Available: https://openreview.net/forum?id=Sy2fzU9gl (visited on 12/08/2023) (cit. on p. 15).

[86] S. Saremi and A. Hyvärinen, "Neural Empirical Bayes," *Journal of Machine Learning Research*, vol. 20, no. 181, pp. 1–23, 2019, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v20/19-216.html (visited on 11/28/2023) (cit. on p. 15).

[87] Max-Planck-Institut fuer Biogeochemie, *Weather Data*, 2008. [Online]. Available: https://www.bgc-jena.mpg.de/wetter/ (visited on 10/31/2023) (cit. on pp. 15, 21).

[88] A. Johnson, T. Pollard, and R. Mark, *MIMIC-III Clinical Database*, 2015. DOI: 10.13026/C2XW26. [Online]. Available: https://physionet.org/content/mimiciii/1.4/ (visited on 10/31/2023) (cit. on p. 17).

[89] A. Gu, K. Goel, and C. Ré, *Efficiently Modeling Long Sequences with Structured State Spaces*, arXiv:2111.00396 [cs], Aug. 2022. DOI: 10.48550/arXiv.2111.00396. [Online]. Available: http://arxiv.org/abs/2111.00396 (visited on 10/03/2023) (cit. on pp. 17, 18).

[90] K. Goel, A. Gu, C. Donahue, and C. Re, "It's Raw! Audio Generation with State-Space Models," en, in *Proceedings of the 39th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jun. 2022, pp. 7616–7633. [Online]. Available: https://proceedings.mlr.press/v162/goel22a.html (visited on 10/03/2023) (cit. on pp. 17, 18, 23).

[91] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Ré, "HiPPO: Recurrent Memory with Optimal Polynomial Projections," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 1474–1487. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/102f0bb6efb3a6128a3c750dd16729be-Abstract.html (visited on 11/29/2023) (cit. on p. 18).

[92] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré, *How to Train Your HiPPO: State Space Models with Generalized Orthogonal Basis Projections*, arXiv:2206.12037 [cs], Aug. 2022. DOI: 10.48550/arXiv.2206.12037. [Online]. Available: http://arxiv.org/abs/2206.12037 (visited on 11/29/2023) (cit. on p. 18).

[93] P. Wagner, N. Strodthoff, R.-D. Bousseljot, *et al.*, "PTB-XL, a large publicly available electrocardiography dataset," en, *Scientific Data*, vol. 7, no. 1, p. 154, May 2020, Number: 1 Publisher: Nature Publishing Group, ISSN: 2052-4463. DOI: 10.1038/s41597-020-0495-6. [Online]. Available: https://www.nature.com/articles/s41597-020-0495-6 (visited on 11/29/2023) (cit. on p. 18).

[94] A. Gu and T. Dao, *Mamba: Linear-Time Sequence Modeling with Selective State Spaces*, arXiv:2312.00752 [cs], Dec. 2023. DOI: 10.48550/arXiv.2312.00752. [Online]. Available: http://arxiv.org/abs/2312.00752 (visited on 12/11/2023) (cit. on pp. 18, 23).

[95] P. J. Huber, "Robust Statistics," en, in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer, 2011, pp. 1248–1251, ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_594. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_594 (visited on 11/30/2023) (cit. on p. 19).

[96] L. Li, J. Yan, Q. Wen, Y. Jin, and X. Yang, "Learning Robust Deep State Space for Unsupervised Anomaly Detection in Contaminated Time-Series," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 6058–6072, Jun. 2023, Conference Name: IEEE Transactions on Knowledge and Data Engineering, ISSN: 1558-2191. DOI: 10.1109/TKDE.2022.3171562. [Online]. Available: https://ieeexplore.ieee.org/document/9773982 (visited on 11/30/2023) (cit. on p. 19).

[97] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond," *International Journal of Forecasting*, vol. 32, no. 3, pp. 896–913, Jul. 2016, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2016.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207016000133 (visited on 11/30/2023) (cit. on p. 19).

[98] C. Miller, A. Kathirgamanathan, B. Picchetti, *et al.*, "The Building Data Genome Project 2, energy meter data from the ASHRAE Great Energy Predictor III competition," *Scientific Data*, vol. 7, no. 1, p. 368, Oct. 2020, arXiv:2006.02273 [stat], ISSN: 2052-4463. DOI: 10.1038/s41597-020-00712-x. [Online]. Available: http://arxiv.org/abs/2006.02273 (visited on 11/30/2023) (cit. on p. 19).

[99] M. Farrokhabadi, *Day-Ahead Electricity Demand Forecasting: Post-COVID Paradigm*, en, Nov. 2020. [Online]. Available: https://ieee-dataport.org/competitions/day-ahead-electricity-demand-forecasting-post-covid-paradigm (visited on 11/30/2023) (cit. on p. 19).

[100] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, *Mixup: Beyond Empirical Risk Minimization*, arXiv:1710.09412 [cs, stat], Apr. 2018. DOI: 10.48550/arXiv.1710.09412. [Online]. Available: http://arxiv.org/abs/1710.09412 (visited on 10/18/2023) (cit. on p. 20).

[101] NordPool Group, *NordPool Market Data*, 2020. [Online]. Available: https://www.nordpoolgroup.com/Market-data1/Power-system-data (visited on 10/31/2023) (cit. on p. 21).

[102] W. Fan, S. Zheng, X. Yi, *et al.*, *DEPTS: Deep Expansion Learning for Periodic Time Series Forecasting*, arXiv:2203.07681 [cs, stat], Mar. 2022. DOI: 10.48550/arXiv.2203.07681. [Online]. Available: http://arxiv.org/abs/2203.07681 (visited on 10/31/2023) (cit. on p. 21).

[103] Energy Online, *CAISO: Actual Load Data*, 2012. [Online]. Available: http://www.energyonline.com/Data/GenericData.aspx?DataId=18&CAISO___Actual_Load (visited on 10/31/2023) (cit. on p. 21).

[104] J. Ho and T. Salimans, *Classifier-Free Diffusion Guidance*, arXiv:2207.12598 [cs], Jul. 2022. DOI: 10.48550/arXiv.2207.12598. [Online]. Available: http://arxiv.org/abs/2207.12598 (visited on 12/03/2023) (cit. on p. 21).

[105] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, M4 Competition, vol. 36, no. 1, pp. 54–74, Jan. 2020, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2019.04.014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207019301128 (visited on 12/05/2023) (cit. on p. 22).

[106] R. Godahewa, C. Bergmeir, G. I. Webb, R. J. Hyndman, and P. Montero-Manso, *Monash Time Series Forecasting Archive*, arXiv:2105.06643 [cs, stat], May 2021. DOI: 10.48550/arXiv.2105.06643. [Online]. Available: http://arxiv.org/abs/2105.06643 (visited on 12/05/2023) (cit. on p. 22).