An Analysis of Probabilistic Time Series Forecasting with Autoregressive Diffusion Models

Diego Martí Monsó*
TUM School of Computation, Information and Technology
Technical University of Munich
diego.marti@tum.de

Abstract

Time series forecasting holds great relevance across numerous domains, serving as a vital tool for making informed decisions and predictions based on historical data. Its applications are diverse, ranging from finance and economics to weather forecasting and healthcare. Recently, autoregressive diffusion-based methods for time series forecasting have been proposed, showing great potential in the field. Diffusion models are deep generative models with an extraordinary ability to approximate complex and high-dimensional data distributions. Such models have primarily been employed in tasks like image and video synthesis, and have attracted considerable interest from the media and the general public due to their generative power. Current research efforts remain focused on reducing the computational cost for training and inference while enhancing the generative capabilities of diffusion models, as well as applying these models to novel problem settings that also require generative modeling. Therefore, this paper offers an overview of diffusion models and presents the latest advancements in the field. Additionally, we argue that diffusion models hold great promise for time series forecasting due to their autoregressive nature, despite their current performance lagging behind that of the state-of-the-art. We analyze autoregressive diffusion time series models, exploring their strengths, limitations, and potential avenues for future research in the field. By investigating the potential of diffusion models in time series forecasting, we aim to contribute to the advancement of this field and highlight areas for further improvement.

1 Introduction

Time series forecasting is a fundamental task in the field of machine learning, with numerous applications spanning diverse domains such as finance, meteorology, energy management, and healthcare. In these areas, accurately predicting future values in a time-dependent sequence enables informed decision-making, resource allocation, and risk mitigation. However, forecasting time series data poses significant challenges due to the temporal dependencies, non-stationarity and complex dynamics inherent to most of the modeled processes. Over the years, researchers have explored various approaches to tackle the time series forecasting problem, including autoregressive models, recurrent neural networks (RNNs), and ensemble techniques [21, 28, 47, 4, 42, 41, 24, 5, 37, 36, 46]. While these methods have achieved considerable success, they often struggle with capturing long-range dependencies, handling irregular patterns, and adapting to evolving data distributions.

In recent years, diffusion models have emerged as a promising approach for time series forecasting, leveraging concepts from statistical physics and stochastic processes. Originally developed to model diffusion phenomena and random walks, diffusion models have garnered significant attention for their

*Work conducted while at the MIT Computer Science and Artificial Intelligence Laboratory.

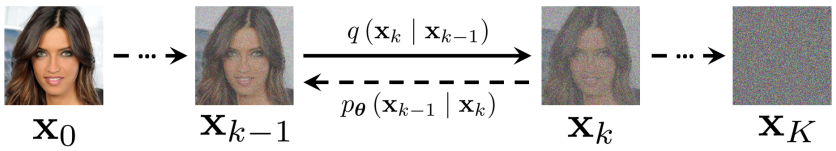

Figure 1: Directed graphical model of the forward and reverse diffusion processes as outlined in [17], illustrated on a sample from the CelebA [26] dataset.

outstanding generative power, achieving impressive results in image and video synthesis [30, 38, 15, 19]. These models offer a flexible framework to capture and sample from complex data distributions by simulating the motion of particles through a diffusion process. Building upon the foundation laid by previous research, recent advancements have extended diffusion models specifically for time series forecasting tasks [36].

Diffusion models for sequence generation and forecasting can be non-autoregressive [22] and autoregressive [36]. Non-autoregressive models in general are designed to optimize over the joint probability distribution of the values in the sequence over a fixed prediction window. While non-autoregressive models effectively reduce compounding error, as is the case in the current state-of-the-art in time series modeling [46] with a transformer architecture, it comes at the cost of losing Markovian dynamics, which prevents generalization to new trajectories. In contrast, autoregressive approaches are able to model pairwise transitions and thus capture Markovian dependencies, enabling generalization to new applications like online decision-making. Therefore, we emphasize autoregressive diffusion models as powerful and promising tools for time series forecasting.

However, we also highlight two important shortcomings that are preventing autoregressive diffusion time series models from reaching their full potential:

1. Most autoregressive diffusion models consist of a deterministic RNN state transition model, followed by a probabilistic diffusion model that acts as the emission or observation model, as depicted in Fig. 3. Thus, the generative capability of the time series prediction model is bottlenecked by the RNN, which is known to suffer from vanishing and exploding gradients [33] despite recent advancements [32]. Furthermore, the RNN state transition model neglects the probabilistic nature of the problem setting. It is only the powerful diffusion observation model that recovers stochastic information from perturbed states to decode them into valid output distributions at each step.

2. Autoregressive diffusion models for sequence forecasting are commonly trained on single-step objectives due to the prohibitive cost that Langevin sampling and even relaxations thereof [44, 40] introduce into multi-step objectives. However, at inference, generated values are fed back as input to the next step, causing a distribution shift with the ground truth observations used during training. Such a distribution mismatch between training and inference is known to degrade performance in other architectures, and "overshooting" losses that balance between single-step and multi-step predictions are often incorporated to address the issue [12, 11, 13, 14].

For the reasons above, we suggest that autoregressive diffusion time series models will benefit from architectures that are purely based on the principle of diffusion, where not only output observations but also internal states are diffused. Moreover, a tractable multi-step diffusion objective needs to be formulated to better exploit the potential of diffusion models for time series forecasting. Our proposal is still to be demonstrated in future work.

The remainder of this work is structured as follows: in Section 2, we revise the underlying principles and mathematical foundations of diffusion models. Section 3 defines the problem of time series prediction, and we present an overview of the literature in the field, focusing particularly on autoregressive methods and diffusion models. We present experimental results in Section 4 and Section 5 concludes the paper and outlines next steps.

2 Diffusion Models

Denoising diffusion probabilistic models [17, 30] are a class of generative models that have become the state-of-the-art in several downstream applications, such as image generation [38]. The general idea is to iteratively remove noise from a multivariate data point. An interpretation of diffusion modeling is that starting with isotropic noise, we use Langevin dynamics via the gradients of the data distribution approximated with score matching to sample new points that lie in the lower dimensional data manifold [45]. The main difficulty lies in the complexity of the data distribution, where even high-density areas are extremely scarce, especially in the case of high dimensionality.

Consider the D-dimensional data point $\mathbf{x}_0 \in \mathbb{R}^D$, which is sampled from the unknown data distribution $q(\mathbf{x})$ as $\mathbf{x}_0 \sim q(\mathbf{x})$. Diffusion models aim to approximate the real data distribution $q(\mathbf{x})$ via the probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$, parameterized by $\boldsymbol{\theta}$ as a neural network, such that $\mathbf{x}_0$ can be sampled along with new data points. Thus, the model is trained to minimize the negative log-likelihood

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x})} \left[-\log \left(p_{\boldsymbol{\theta}} (\mathbf{x}_0) \right) \right]. \tag{1}$$

However, the loss function (1) is intractable, because we define the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$ over the data point $\mathbf{x}_0$ as

$$p_{\boldsymbol{\theta}} (\mathbf{x}_0) := \int p_{\boldsymbol{\theta}} (\mathbf{x}_{0:K}) \, \mathrm{d}\mathbf{x}_{1:K} \tag{2}$$

by integrating the joint probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{0:K})$ over the latent random variables $\mathbf{x}_1, \dots \mathbf{x}_K$, where $\mathbf{x}_k \in \mathbb{R}^D \, \forall k \in \{1, \dots, K\}$ and $K \in \mathbb{N}$. In contrast to other generative deep latent-variable models, such as variational autoencoders (VAEs, [23]), the posterior probability distribution

$$q (\mathbf{x}_{1:K} \mid \mathbf{x}_0) := \prod_{k=1}^{K} q (\mathbf{x}_k \mid \mathbf{x}_{k-1})$$

over the sequence of latents has no learnable parameters. Instead, we obtain the latent variables by sequentially applying a degradation operation to the original signal $\mathbf{x}_0$. The so-called *forward diffusion process* consists of a Markov chain that gradually adds Gaussian noise to the input over K steps with the transition probability distribution

$$q (\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N} \left(\mathbf{x}_k; \sqrt{1 - \beta_k} \mathbf{x}_{k-1}, \beta_k \mathbf{I} \right). \tag{3}$$

The monotonically increasing forward noising schedule $\beta_1, \dots \beta_K$, where $\beta_k \in (0, 1) \, \forall k \in \{1, \dots, K\}$, controls the variance of the noise contamination. Using the reparametrization trick, we find a closed-form representation to efficiently sample the latent

$$\mathbf{x}_k \sim q (\mathbf{x}_k \mid \mathbf{x}_0) = \mathcal{N} \left(\mathbf{x}_k; \sqrt{\bar{\alpha}_k} \mathbf{x}_0, (1 - \bar{\alpha}_k) \, \mathbf{I} \right) \tag{4}$$

at any step $k \in \{0, \dots, K\}$ as an interpolation $\mathbf{x}_k = \sqrt{\bar{\alpha}_k} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_k} \boldsymbol{\varepsilon}$ between the clean signal $\mathbf{x}_0$ and some fixed Gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ with $\alpha_k := 1 - \beta_k$ and $\bar{\alpha}_k := \prod_{i=1}^{t} \alpha_i$. Typically, we design the noising schedule so that $\bar{\alpha}_K \approx 0$, by which

$$q (\mathbf{x}_K) := \int q (\mathbf{x}_K \mid \mathbf{x}_0) \, q (\mathbf{x}_0) \, \mathrm{d}\mathbf{x}_0 \approx \mathcal{N} (\mathbf{x}_K; \mathbf{0}, \mathbf{I})$$

can be assumed to be an isotropic Gaussian for $K \to \infty$. Nichol et al. [30] propose using a fixed cosine schedule[2] with $K = 1000$ for the forward process. Then, the *reverse diffusion process*, described by the joint probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{0:K})$ from (2), can also be modeled as a Markov chain

$$p_{\boldsymbol{\theta}} (\mathbf{x}_{0:K}) := p (\mathbf{x}_K) \prod_{k=1}^{K} p_{\boldsymbol{\theta}} (\mathbf{x}_{k-1} \mid \mathbf{x}_k) \tag{5}$$

that starts with a sample $\mathbf{x}_K$ of pure noise drawn from the prior distribution $p(\mathbf{x}_K) = \mathcal{N}(\mathbf{x}_K; \mathbf{0}, \mathbf{I})$ as $\mathbf{x}_K \sim p(\mathbf{x}_K)$. Therefore, the network must learn Gaussian transition kernels that approximate the inverse $q(\mathbf{x}_{k-1} \mid \mathbf{x}_k)$ of the forward transitions in (3) as

$$p_{\boldsymbol{\theta}} (\mathbf{x}_{k-1} \mid \mathbf{x}_k) := \mathcal{N} \left(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}} (\mathbf{x}_k, k), \boldsymbol{\Sigma}_{\boldsymbol{\theta}} (\mathbf{x}_k, k) \right). \tag{6}$$

[2] Recent work [9, 25] shows that commonly used variance schedules fail at completely removing lower-frequency information from the clean data point. Unless zero signal-to-noise ratio is enforced at the end of the forward diffusion process, the reverse process suffers from a distribution shift between the aggregate posterior $q(\mathbf{x}_K \mid \mathbf{x}_0)$ seen during training and the prior $p(\mathbf{x}_K)$ used for generation.

An illustration of the forward and backward pass can be found in Fig. 1. The choice of Gaussian transitions in (6) is valid as long as β_k is kept small [43]. This way, the model needs to learn the parameters $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{x}_k, k)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{x}_k, k)$ of the reverse transition. In this sense, we can approximate the log-likelihood objective $\mathcal{L}$ in 1 via the evidence lower bound (ELBO)

$$\mathcal{L} \leq \mathcal{L}_K + \sum_{k=2}^{K} \mathcal{L}_{k-1} + \mathcal{L}_0 \tag{7}$$

$$\mathcal{L}_K = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x})} \left[D_{\mathrm{KL}} \left(q\left(\mathbf{x}_K \mid \mathbf{x}_0\right) \parallel p\left(\mathbf{x}_K\right) \right) \right] \tag{8}$$

$$\mathcal{L}_{k-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), \mathbf{x}_k \sim q(\mathbf{x}_k \mid \mathbf{x}_0)} \left[D_{\mathrm{KL}} \left(q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k, \mathbf{x}_0\right) \parallel p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right) \right) \right] \quad \forall k \in \{2, \ldots, K\} \tag{9}$$

$$\mathcal{L}_0 = -\mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), \mathbf{x}_1 \sim q(\mathbf{x}_1 \mid \mathbf{x}_0)} \left[\log \left(p_{\boldsymbol{\theta}}\left(\mathbf{x}_0 \mid \mathbf{x}_1\right) \right) \right] \tag{10}$$

The term (8) has no learnable parameters $\boldsymbol{\theta}$ and is thus ignored during training, although it is close to zero if the requirement $q\left(\mathbf{x}_K \mid \mathbf{x}_0\right) \approx \mathcal{N}\left(\mathbf{x}_K; \mathbf{0}, \mathbf{I}\right)$ is fulfilled. Also, (10) represents a variational reconstruction term. Notably, the forward process posteriors $q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)$ become tractable when conditioned on $\mathbf{x}_0$ via Bayes theorem, such that we can use the KL divergence in (9) to compute the variational gap between $p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)$ and

$$q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{k-1}; \tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right), \tilde{\beta}_k \mathbf{I}\right),$$

where the mean $\tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right)$ and the variance schedule $\tilde{\beta}_k$ are given by

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right) &:= \frac{\sqrt{\bar{\alpha}_{k-1}} \beta_k}{1-\bar{\alpha}_k} \mathbf{x}_0 + \frac{\sqrt{\alpha_k}\left(1-\bar{\alpha}_{k-1}\right)}{1-\bar{\alpha}_k} \mathbf{x}_k, \\ \tilde{\beta}_k &:= \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k} \beta_k. \end{aligned} \tag{11}$$

The variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ in the reverse transition kernel (6) can be either fixed [17] to $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) := \sigma_t^2 \mathbf{I}$ or made learnable [30]. The mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ could be directly predicted, but we obtain better results by substituting the explicit form of $\mathbf{x}_k$ from (4) into the formulation of the mean in (11) to obtain

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) = \frac{1}{\sqrt{\alpha_k}} \left(\mathbf{x}_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}} \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) \right)$$

with the neural network $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ that predicts the noise $\boldsymbol{\varepsilon}$ present in $\mathbf{x}_k$. The ELBO in (7) can then be reweighted by sampling k from the uniform distribution $k \sim \mathcal{U}\left(1, K\right)$ and computing the denoising term $\mathcal{L}_{k-1}$ from (9) for $k \in \{1, \ldots, K\}$, which can be simplified to

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), k \sim \mathcal{U}(1,K), \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})} \left[\left\| \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) \right\|^2 \right] \tag{12}$$

in the case of fixed variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) := \sigma_t^2 \mathbf{I}$.

The neural network $\boldsymbol{\theta}$ is commonly parameterized as a U-Net [39] with self-attention [1, 27]. Rombach et al. [38] propose to denoise the data points in the latent space of a VAE to alleviate the high computational and time cost of diffusion models both during training and at sampling time. Moreover, more efficient sampling techniques, modified formulations of the simplified objective, and distillation approaches have been proposed in [44, 40, 34] and widely adopted by the research community. Classifier-free guidance [18] has become a popular method to condition diffusion models on feature vectors, such as CLIP [35] embeddings of text descriptions. The guidance parameter can be tuned to achieve a balance between perceptual quality of the generations (often measured in terms of the FID score [16]) and variation between samples. Furthermore, the ControlNet [49] architecture allows to spatially condition the diffusion model and fine-tune large pre-trained models. There have been claims made that Gaussian noise is not necessary for denoising diffusion models to work and that the model can work even with deterministic perturbations [2], showing the robustness of the diffusion framework. However, non-Gaussian perturbations have been found to hurt the generative performance both in terms of quality and variability.

3 Autoregressive Probabilistic Time Series Forecasting Models

In this section, we define the problem setting of probabilistic time series forecasting and provide an overview of autoregressive approaches in the literature.

3.1 Problem Formulation

Let $\boldsymbol{X} = \{\mathbf{x}_t\}_{t=1}^{T}$ be a sequence (multivariate time series) of D-dimensional observations $\mathbf{x}_t \in \mathbb{R}^D$ of some underlying dynamical process, sampled in discrete time steps $t \in \{1, \dots, T\}$, where $T \in \mathbb{N}$. In the problem setting of probabilistic time series forecasting, the sequence $\boldsymbol{X} = \{\boldsymbol{X}_c, \boldsymbol{X}_p\}$ is split into two subsequences at time step $t_0 \in \mathbb{N}$ with $1 < t_0 \leq T$: the context window $\boldsymbol{X}_c := \{\mathbf{x}_t\}_{t=1}^{t_0-1}$ (also called history or evidence) of length $t_0 - 1$, and the prediction window $\boldsymbol{X}_p := \{\mathbf{x}_t\}_{t=t_0}^{T}$ of length $T - t_0 + 1$ (also known as prediction horizon). Then, the task is to model the conditional joint probability distribution

$$q(\mathbf{x}_{t_0:T} \mid \mathbf{x}_{1:t_0-1}) := \prod_{t=t_0}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{1:t-1}) \tag{13}$$

over the samples in the prediction window. If we know the distribution in (13), we can sample forecast prediction sequences given some initial context from the evidence sequence. However, most time-dependent data generation processes in nature have complex dynamics and no tractable formulation of $q(\mathbf{x}_{t_0:T} \mid \mathbf{x}_{1:t_0-1})$. Instead, we generally construct statistical models that approximate the generative process in (13) and estimate quantiles via Monte Carlo sampling of simulated trajectories. In this way, confidence levels or uncertainty measures can be calculated, and point forecasts can be produced as the mean or median trajectory [21].

3.2 Covariates

Often, such statistical models that approximate (13) benefit from manually curated features as additional input to the observations. A sequence of covariates $\boldsymbol{C} = \{\mathbf{c}_t\}_{t=1}^{T}$ can be constructed to help the model recognize seasonal patterns and other temporal dependencies. Covariates may be composed of lagged inputs, as well as learned embeddings or handcrafted temporal features that encode information such as the hour of the day or the day of the month, depending on the sampling rate of the particular time series that is being modeled. Therefore, covariates are known for the entire interval $[1, T]$, even at inference. We can easily incorporate covariates into the probabilistic framework as

$$q(\mathbf{x}_{t_0:T} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:T}) := \prod_{t=t_0}^{T} q(\mathbf{x}_t \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:T}). \tag{14}$$

The benefit obtained from covariates is highly dependent on the characteristics of both the time series and the model used, as well as the feature engineering practices followed. In fact, covariates may even hurt performance under certain circumstances. Usually, designing covariates is a highly empirical process. Thus, we remind the reader that covariates may optionally be left out of the future discussion by not using them as a conditioning signal.

3.3 Variational State-space Models

We can model the time series $\boldsymbol{X}$ via a latent state-space model [7, 21]:

$$\mathbf{h}_t \sim p(\mathbf{h}_t \mid \mathbf{h}_{t-1}), \tag{15}$$

$$\mathbf{x}_t \sim p(\mathbf{x}_t \mid \mathbf{h}_{t-1}), \tag{16}$$

where the sequence $\{\mathbf{h}_t\}_{t=1}^{T}$ of S-dimensional hidden states $\mathbf{h}_t \in \mathbb{R}^S$ that governs the dynamics follows the transition model (15), and the observations are generated from the latent states via the observation model (16). Note that we can update the transition model (15) with the observation $\mathbf{x}_t$ and the covariate $\mathbf{c}_t$ at time step t to obtain an approximate state posterior

$$q(\mathbf{h}_{1:T} \mid \mathbf{x}_{1:T}, \mathbf{c}_{1:T}) = \prod_{t=1}^{T} q(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t) \tag{17}$$

via filtering, given some initial state $\mathbf{h}_0$. The described variational state-space model is recurrent, because the state transitions $q(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t)$ depend on the previous state $\mathbf{h}_{t-1}$. Furthermore, the model is also autoregressive, as the output variable $\mathbf{x}_t$ is fed back as input in the following step.

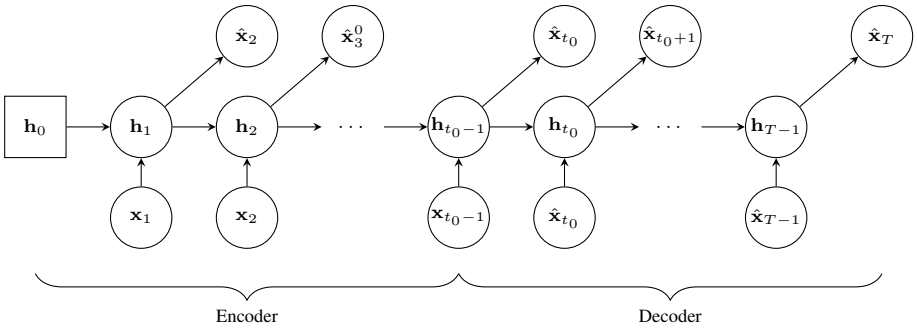

Figure 2: Full schematics of the recurrent encoder-decoder. Square nodes indicate deterministic variables, while circles represent stochastic variables. The generative process is reflected by the solid arrows.

3.4 Recurrent Encoder-Decoders

One of the most successful approaches for autoregressive sequence-to-sequence prediction has been the encoder-decoder architecture [4], which consists of an RNN that encodes the history sequence into an internal state representation that is then decoded by a second RNN into the prediction sequence. The application of recurrent encoder-decoder networks to time series forecasting was first inspired by the initial success of the encoder-decoder in neural machine translation [3].

3.4.1 General Formulation

Due to the aforementioned recurrence in the state transition $q(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t)$, RNNs have widely been used in the literature to approximate the filtering state posterior in (17) during the conditioning phase as a sequential encoder

$$q_{\boldsymbol{\psi}}(\mathbf{h}_{1:t_0-1} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:t_0-1}) = \prod_{t=1}^{t_0-1} q_{\boldsymbol{\psi}}(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t) \tag{18}$$

with parameters $\boldsymbol{\psi}$ and initial state $\mathbf{h}_0 := \mathbf{0}$ without loss of generality. In time series forecasting, the sequence of evidence $\{\mathbf{x}_t\}_{t=1}^{t_0-1}$ is passed through the RNN sequence encoder (18) to infer the approximate final context latent $\mathbf{h}_{t_0-1}$ that models the internal state of the generative process at the end of the evidence window at step $t_0 - 1$. The encoding process is thus given by the marginal

$$q_{\boldsymbol{\psi}}(\mathbf{h}_{t_0-1} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:t_0-1}) = \int q_{\boldsymbol{\psi}}(\mathbf{h}_{1:t_0-1} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:t_0-1}) \, \mathrm{d}\mathbf{h}_{1:t_0-2}. \tag{19}$$

Then, an unseen sample $\hat{\mathbf{x}}_{t_0}$ can be estimated from $\mathbf{h}_{t_0-1}$ by sampling from

$$\hat{\mathbf{x}}_t \sim p_{\boldsymbol{\psi}}(\hat{\mathbf{x}}_t \mid \mathbf{h}_{t-1}), \tag{20}$$

which approximates the observation model (16) as a neural network parameterized by $\boldsymbol{\psi}$. The estimate $\hat{\mathbf{x}}_{t_0}$ is then fed back autoregressively to the transition model to unroll the predictions for further time steps. Formally, the transition forecasting part of the process is expressed via the recursive decoder[3]

$$\begin{aligned} &p_{\boldsymbol{\psi}}(\mathbf{h}_{t_0-1:T-1} \mid \mathbf{x}_{1:t_0-1}, \hat{\mathbf{x}}_{t_0:T-1}, \mathbf{c}_{1:T-1}) = \\ &\underbrace{q_{\boldsymbol{\psi}}(\mathbf{h}_{t_0-1} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:t_0-1})}_{\text{Encoder}} \underbrace{\prod_{t=t_0}^{T-1} p_{\boldsymbol{\psi}}(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \hat{\mathbf{x}}_t, \mathbf{c}_t)}_{\text{Decoder}}, \end{aligned} \tag{21}$$

where $\hat{\mathbf{x}}_{t_0:T-1}$ is a trajectory of estimations that we assume to be given by sampling from (20) for $t \in \{t_0, \ldots, T-1\}$. Importantly, the RNN $q_{\boldsymbol{\psi}}(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t)$ models the posterior of the state transitions in the encoder during the context window, while its decoder counterpart

[3]We omit the prediction of $\mathbf{h}_T$, because the final target $\hat{\mathbf{x}}_T$ can already be estimated from $\mathbf{h}_{T-1}$ with (20).

$p_{\boldsymbol{\psi}}(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \hat{\mathbf{x}}_t, \mathbf{c}_t)$ represents a prior probability during the prediction phase, since only an estimation $\hat{\mathbf{x}}_t$ is available. Both RNNs do not necessarily need to be the same networks, but it is common practice to use the same architecture and weights for the two [42]. Finally, we can approximate the target distribution $q(\mathbf{x}_{t_0:T} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:T})$ from (14) as

$$q_{\boldsymbol{\psi}}(\hat{\mathbf{x}}_{t_0:T} \mid \mathbf{x}_{1:t_0-1}, \mathbf{c}_{1:T}) = \int \underbrace{p_{\boldsymbol{\psi}}(\hat{\mathbf{x}}_{t_0:T} \mid \mathbf{h}_{t_0-1:T-1})}_{\text{Observation model}} \underbrace{p_{\boldsymbol{\psi}}(\mathbf{h}_{t_0-1:T-1} \mid \mathbf{x}_{1:t_0-1}, \hat{\mathbf{x}}_{t_0:T-1}, \mathbf{c}_{1:T})}_{\text{Transition model}} \mathrm{d}\mathbf{h}_{t_0-1:T-1}, \tag{22}$$

according to the law of total probability. The unrolling scheme of the probabilistic encoder-decoder in its general formulation is depicted in Fig. 2.

3.4.2 Deep Learning Methods

Over the last decade, deep learning methods have become the state-of-the-art in time series forecasting, as can be concluded from Table 1. Starting with early successes as in [6], neural networks have become a common way to model the time series prediction problem. Most algorithms follow a recurrent encoder-decoder architecture and apply different approaches to deal with uncertainty, inaccuracies, data dimensionality, and long-term dependencies. Given that the encoder-decoder framework usually involves alternating state transitions from the transition model with output generations from the observation model, errors can quickly compound. While we use ground truth observations in the encoding phase, yielding approximate state posteriors, the decoder consumes previous predictions as input, causing errors to compound during decoding. These errors cause the states, and ultimately individual outputs or the output trajectory as a whole to fall out of distribution over time. Thus, research usually focuses on making either the transition or the observation model (or both simultaneously) more powerful.

As previously indicated, RNNs tend to be the network used to model the state transition model, due to the recurrence of the state in the problem setting. Specifically, the LSTM [20] and more recently the GRU [4] have become the RNN architectures of choice, particularly due to their ability to remember long-range dependencies. Recently, structured deep state-space models [8] have also shown an outstanding capability to capture long dependencies by following a principled approach to simulate the state-space model as a deterministic linear time-invariant system. Concurrently, a simple yet powerful linear recurrent unit has been proposed in [32] that can compete with structured deep state-space models in long range tasks. Despite their remarkable performance in time series forecasting and similar problem settings, all these architectures share the problem that they are entirely deterministic. The state transition probability density function $\mathbf{h}_t \sim q(\mathbf{h}_t, \mid \mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t)$ is instead approximated by a deterministic state transition difference equation:

$$\mathbf{h}_t = f_{\boldsymbol{\psi}}(\mathbf{h}_{t-1}, \mathbf{x}_t, \mathbf{c}_t),$$

parameterized by the RNN $\boldsymbol{\psi}$. Such a state transition function can suffer from mean collapse, where the average transition is learned. While learning the mean transition might be desirable when the true transition distribution is nearly a Dirac delta distribution, such as a Gaussian with low variance, it is problematic when the transition density is more complicated. A simple example could be any bimodal distribution (or multimodal in the more general case) where the mean value has almost zero probability (think of a ball falling through a Galton board). In the best case, the deterministic network could still learn to predict the peak density of the dominating mode, which would still not be an accurate representation of the true transition distribution. For this reason, Salinas et al. [42] incorporate a probabilistic RNN architecture that is able to better capture the probabilistic nature of time series data. Previous works [10, 12] had already formulated either purely stochastic, or mixed deterministic and probabilistic transition models. Note that the authors additionally design the observation model in a variational setting as a VAE, instead of using a simple MLP or the transition RNN itself to produce the outputs. Furthermore, observations are not necessarily decoded with the observation model at every step, but can be kept in an lower-dimensional intermediate latent representation for efficiency, and still be fed autoregressively as input to the next step. This line of work has culminated in the successful Dreamer reinforcement learning agent [11, 13, 14] that operates with observations in image space. One could easily reformulate the problem setting of world modeling in reinforcement learning as a time series prediction task by considering actions, rewards, and observation frames as mere features of the time series measurements $\mathbf{x}_t$.

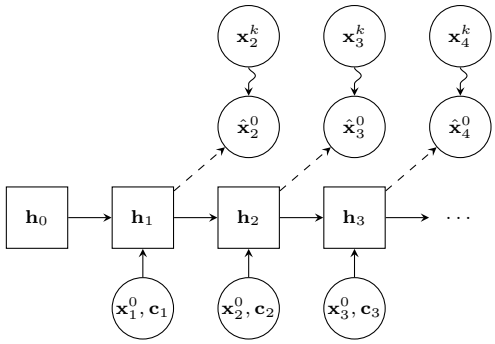

Figure 3: Unrolling scheme of TimeGrad [36]. Square nodes indicate deterministic variables, while circles represent stochastic variables. The RNN generative process is reflected by the solid arrows and the reverse diffusion process by the curved arrows. Dashed arrows imply conditioning signals to the diffusion model. TimeGrad rolls out the states deterministically via an RNN and decodes them probabilistically via diffusion.

Finally, time series prediction can also be modeled with autoregressive diffusion models. To date, TimeGrad [36] is the most performant autoregressive diffusion model. TimeGrad learns to model

$$q\big(\mathbf{x}^0_{t_0:T} \mid \mathbf{x}^0_{1:t_0-1}, \mathbf{c}_{1:T}\big) = \prod_{t=t_0}^{T} q\big(\mathbf{x}^0_t \mid \mathbf{x}^0_{1:t-1}, \mathbf{c}_{1:T}\big),$$

which is an extension of (13) where the superscript refers to the diffusion time step k and the subscript to the time series time step t. The time dynamics are modeled by an RNN transition model $q_{\boldsymbol{\psi}}\big(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \mathbf{x}^0_t, \mathbf{c}_t\big)$ in the encoder and $p_{\boldsymbol{\psi}}\big(\mathbf{h}_t \mid \mathbf{h}_{t-1}, \hat{\mathbf{x}}^0_t, \mathbf{c}_t\big)$ in the decoder, and the observation model is approximated by a diffusion model $p_{\boldsymbol{\psi}}\big(\hat{\mathbf{x}}^0_t \mid \mathbf{h}_{t-1}\big)$. The unrolling scheme of TimeGrad is depicted in Fig. 3. At this point, we want to once again highlight the great potential of autoregressive diffusion models for time series forecasting, if used to model the variational state-space model in its entirety, and not only the observation model. Limiting the state transitions to an RNN undermines the probabilistic and multimodal nature of the problem setting, which might require to foresee multiple futures. As we explained before, the RNN state transitions can collapse towards the mean of the true transition distribution and produce single-point Dirac state transition approximations that may be out of distribution. While purely deterministic RNNs can be extender with variational modules to produce probabilistic transitions, such as via VAEs with Gaussian or categorical transition kernels [11, 13], it is known that VAEs have limited expressive power and have been recently outperformed by other generative models (GANs, diffusion models). In the context of modeling highly stochastic state transitions, VAEs are limited optimistically only by the mismatch between the chosen kernel and the real transition distribution, or by the quantization error in the case of categorical kernels. Thus, we suggest that a diffusion would be able to better capture the true time dynamics and ensure that the trajectory, and not only the individual trajectory tokens, is consistent with the training data. Moreover, recurrent encoder-decoders that share the same architecture and weights for prior and posterior state transitions suffer from a distribution shift between the estimate $\hat{\mathbf{x}}_t$ and the ground truth measurement $\mathbf{x}_t$.

3.5 Non-Recursive Deep Learning Approaches

While the preceding sections delved deeply into the encoder-decoder approach, primarily focusing on RNN-based models, it is essential to recognize that the landscape of time series prediction is vast and varied. There exist alternative methodologies that, though not grounded in the encoder-decoder or even in the state-space paradigm, still offer valuable insights and solutions to the problem at hand. For the sake of comprehensiveness and to provide a holistic understanding of the available techniques, this section presents an overview over non-recursive methods. These alternative approaches, while deviating from our primary modeling strategy, hold their own merits and can be effectively applied to the time series prediction setting.

Transformers [48] are a family of neural network architectures that leverage the attention mechanism [1, 27] to perform sequence-to-sequence prediction. Recently, transformer-based models have

caught the interest of the general public due to their remarkable performance in natural language processing [31]. Although transformers are generally not applied autoregressively, nor do they offer the flexibility of RNNs that allow for variable-length input and output sequences, the probabilistic transformer presented in [46] (ProTran) sets the current state-of-the-art in time series modeling, as can be read from Table 1. The architecture combines a state-space model with a transformer architecture of hierarchical stochastic variables. The design of the transformer architecture allows it to attend equally over all elements in the fixed-sized input sequence. Therefore, the transformer does not suffer from memory loss within its receptive field, in contrast to most RNN architectures. This argument is used in [46] to claim that ProTran can capture long-term dependencies better than recurrent models, but no experiment has demonstrated better performance on a dataset that requires long-range reasoning. One possible explanation is that the length of the context window of the transformer has a fixed size and cannot be arbitrarily increased, as the computational cost scales drastically with the number of input tokens.

Another interesting approach to sequence prediction, presented in the context of video prediction, is that of frame inpainting [15, 19]. Instead of diffusing a single image, one can concatenate subsequent frames (or frames that follow more flexible or even random schedules) from a video stream for a fixed-size operating window, selectively mask out certain frames during training, and inpaint the missing frames. At test time, this process can be rolled out autoregressively to generate hour-long videos where no single frame falls out of distribution. The authors additionally propose different schedules with different trade-offs, such as which frames to use from the context window and the number of time steps to predict ahead. Even though the output at each individual time step is reasonable, the trajectories a whole are often clearly out of distribution, similar to the claims we make about TimeGrad [36]. For instance, the predicted video can get stuck for unreasonably long periods of time on static moments (e.g., car parked at a traffic light), since the model has no sense of state that encodes waiting times. While such models are restricted to the dynamics present within the operating window (like transformers), they still enable seemingly infinite unrolling of the sequence prediction. Furthermore, most video prediction models can be easily adapted to time series by treating individual frames as time series tokens.

4 Experiments

In this section we briefly present a comparison between some of the presented methods in the setting of time series forecasting.

4.1 Data sets

The models are evaluated in [36, 46] on the public data sets `SOLAR`, `ELECTRICITY`, `TRAFFIC`, `TAXI`, and `WIKIPEDIA`. These data sets have different dimensionality, domains, sampling frequency, and capture seasonal patterns of different lengths. The context and prediction windows are also adapted to each data set, depending on the frequency of the seasonality to be modeled.

4.2 Metric

The Continuous Ranked Probability Score (CRPS) [29] is a scoring function that measures how good the forecast distribution matches the ground truth distribution:

$$\mathrm{CRPS}(F, x) = \int_{\mathbb{R}} \left(F(z) - \mathbb{I}\{x \leq z\}\right)^2 \mathrm{d}z \,,$$

where $F(z)$ is the univariate cumulative distribution function (CDF) over the predicted value, x is a ground truth observation, and $\mathbb{I}\{x \leq z\}$ is the indicator function that is one if $x \leq z$ and zero otherwise. By summing the D-dimensional time series along the dimension axis for simulated samples (resulting in $\hat{F}_{\text{sum}}(t)$) and ground truth data (as $\sum_i x^0_{i,t}$), we can report the $\mathrm{CRPS}_{\text{sum}}$

$$\mathrm{CRPS}_{\text{sum}} = \mathbb{E}_{t\sim\mathcal{U}(t_0,T)}\left[\mathrm{CRPS}\left(\hat{F}_{\text{sum}}(t), \sum_i x^0_{i,t}\right)\right]$$

as the average over the prediction window. The CDF $\hat{F}_{\text{sum}}(t)$ can be estimated via quantiles at each time step t. The lower the $\mathrm{CRPS}_{\text{sum}}$ value, the better does the predicted distribution match the data distribution.

4.3 Quantitative Results

Table 1: Test set CRPS_{sum} (the lower, the better) of different methods on six time series data sets. Means and standard deviations are reported for runs with different seeds.

Method	Exchange	Solar	Electricity	Traffic	Taxi	Wikipedia
VES [21]	$\mathbf{0.005 \pm 0.000}$	0.900 ± 0.003	0.880 ± 0.004	0.350 ± 0.002	-	-
VAR [28]	$\mathbf{0.005 \pm 0.000}$	0.830 ± 0.006	0.039 ± 0.001	0.290 ± 0.001	-	-
VAR-Lasso [28]	0.012 ± 0.000	0.510 ± 0.006	0.025 ± 0.000	0.150 ± 0.002	-	3.100 ± 0.004
GARCH [47]	0.023 ± 0.000	0.880 ± 0.002	0.190 ± 0.001	0.370 ± 0.001	-	-
DeepAR [42]	-	0.336 ± 0.014	0.023 ± 0.001	0.055 ± 0.003	-	0.127 ± 0.042
LSTM-Copula [41]	0.007 ± 0.000	0.319 ± 0.011	0.064 ± 0.008	0.103 ± 0.006	0.326 ± 0.007	0.241 ± 0.033
GP-Copula [41]	0.007 ± 0.000	0.337 ± 0.024	0.025 ± 0.002	0.078 ± 0.002	0.208 ± 0.183	0.086 ± 0.004
KVAE [24]	0.014 ± 0.002	0.340 ± 0.025	0.051 ± 0.019	0.100 ± 0.005	-	0.095 ± 0.012
NKF [5]	-	0.320 ± 0.020	0.016 ± 0.001	0.100 ± 0.002	-	0.071 ± 0.002
Transformer-MAF [37]	$\mathbf{0.005 \pm 0.00}3$	0.301 ± 0.014	0.021 ± 0.000	0.056 ± 0.001	0.179 ± 0.002	0.063 ± 0.003
TimeGrad [36]	0.006 ± 0.001	0.287 ± 0.020	0.021 ± 0.001	0.044 ± 0.006	0.114 ± 0.020	$\mathbf{0.049 \pm 0.002}$
ProTran [46]	-	$\mathbf{0.194 \pm 0.030}$	$\mathbf{0.016 \pm 0.001}$	$\mathbf{0.028 \pm 0.001}$	$\mathbf{0.084 \pm 0.003}$	$\mathbf{0.047 \pm 0.004}$

The quantitative comparison between a selection of the cited methods can be found in Table 1. The probabilistic transformer ProTran sets the current state-of-the-art, followed closely by the autoregressive diffusion model TimeGrad and other transformer architectures like Transformer-MAP. For reference, the RNN model DeepAR is one of the most widely adopted methods in the field of time series forecasting. NKF is a normalizing flow with a Kalman Filter, and KVAE is a VAE that uses a linear state-space to model to describe dynamics. GP-Copula nad LSTM-Copula are both LSTM models, while GARCH is a multivariate heteroskedastic model. VES is an innovation state-space model. VAR-Lasso is a multivariate linear autoregressive model like VAR, but with Lasso regularization.

5 Conclusion

As we have seen, the field of time series forecasting is vast and there are multiple different approaches to the problem. While traditional research has focused mostly on recurrent models that fit into the framework of state-space models and the recurrent encoder-decoder architecture, recent works have explored alternatives such as transformers or diffusion inpainting. Diffusion models can also be used to model state-space models, but current implementations still still use RNNs for time dynamics. A tractable multi-step training objective that allows using diffusion models as state transition models is yet to be formulated.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[2] Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold diffusion: Inverting arbitrary image transforms without noise, 2022.

[3] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 9 2014.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 6 2014.

[5] Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[6] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial neural networks applied to taxi destination prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDC'15, page 40–51, Aachen, DEU, 2015. CEUR-WS.org.

[7] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, 05 2012.

[8] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

[9] Nicholas Guttenberg. Diffusion with offset noise, 2023.

[10] David Ha and Jürgen Schmidhuber. World models, 2018.

[11] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[12] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565, 2019.

[13] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[14] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[15] William Harvey, Saeid Naderiparizi, Vaden Masrani, Christian Weilbach, and Frank Wood. Flexible diffusion modeling of long videos, 2022.

[16] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[19] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[21] Rob Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. *Forecasting with Exponential Smoothing: The State Space Approach*. Springer Science & Business Media, 2008.

[22] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis, 2022.

[23] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[24] Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, 2017.

[25] Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed, 2023.

[26] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[27] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[28] H. Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer Science & Business Media, 2005.

[29] James E. Matheson and Robert L. Winkler. Scoring rules for continuous probability distributions. *Management Science*, 22(10):1087–1096, 1976.

[30] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021.

[31] OpenAI. Gpt-4 technical report, 2023.

[32] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences, 2023.

[33] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[34] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[36] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 2021.

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

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[40] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[41] David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, Jan Gasthaus, and Roberto Medico. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *NeurIPS*, 2019.

[42] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, 2020.

[43] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2256–2265, Lille, France, 07–09 Jul 2015. PMLR.

[44] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[45] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020.

[46] Binh Tang and David S. Matteson. Probabilistic transformer for time series analysis. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[47] Roy van der Weide. Go-garch: A multivariate generalized orthogonal garch model. *Journal of Applied Econometrics*, 17(5):549–564, 2002.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[49] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.