

Recurrent Interpolants for Probabilistic Time Series Prediction

Yu Chen^{*,1}Marin Biloš^{*,1}Sarthak Mittal^{†,*,2,3}Wei Deng¹Kashif Rasul¹Anderson Schneider¹¹Morgan Stanley²Mila³Université de Montréal

Abstract

Sequential models like recurrent neural networks and transformers have become standard for probabilistic multivariate time series forecasting across various domains. Despite their strengths, they struggle with capturing high-dimensional distributions and cross-feature dependencies. Recent work explores generative approaches using diffusion or flow-based models, extending to time series imputation and forecasting. However, scalability remains a challenge. This work proposes a novel method combining recurrent neural networks' efficiency with diffusion models' probabilistic modeling, based on stochastic interpolants and conditional generation with control features, offering insights for future developments in this dynamic field.

1 INTRODUCTION

Autoregression models [Box et al., 2015], such as recurrent neural networks [Graves, 2013, Sutskever et al., 2014, Hochreiter and Schmidhuber, 1997] or transformer models [Vaswani et al., 2017], have been the go-to methods for neural time series forecasting. They are widely applied in finance, biological statistics, medicine, geophysical applications, etc., effectively showcasing their ability to capture short-term and long-term sequential dependencies [Morrell et al., 2021]. These methods can also provide an assessment of prediction uncertainty through probabilistic forecasting by incorporating specific parametric probabilistic models into the output layer of the neural network. For instance, a predictor can model the Gaussian distribution by predicting both mean and covariance. However, the probabilistic

output layer is confined within a simple probability family because the density needs to be parameterized by neural networks, and the loss must be differentiable with respect to neural network parameters.

To better capture sophisticated distributions in time series modeling and learn both the temporal and cross-feature dependencies, a common strategy involves exploring the generative modeling of time series using efficient distribution transportation plans, especially via diffusion or flow-based models. For example, recent works such as Li et al. [2020] propose using latent neural SDE as latent states for modeling time series in a stochastic manner, while Spanini et al. [2022] summarize non-linear extensions of state space models using both deterministic and stochastic transformation plans. Tashiro et al. [2021], Biloš et al. [2023], Chen et al. [2023a], Miguel et al. [2022], Li et al. [2022] studied the application of diffusion models in probabilistic time series imputation and forecasting. The generative model is trained to learn the joint density of the time series window $\mathbf{X}_{\text{pred.}} \in \mathbb{R}^{D \times T_{\text{prediction}}}$ with $D > 1$ variates given $\mathbf{X}_{\text{cont.}} \in \mathbb{R}^{D \times T_{\text{context}}}$. T_{context} is the size of the context window and $T_{\text{prediction}}$ is the size of the subsequent prediction window. During inference, the model performs conditional generation given only the context, similar to the inpainting task in computer vision [Song et al., 2021]. Compared to a recurrent model, where the model size is only proportional to the number of features, but not the length of the time window, such generative model predictors may suffer from scalability issues because the model size is related to both feature dimension and the size of the window. A more computational-friendly framework is needed for large-scale generative model-based time series prediction problems.

Generative modeling excels at modeling complicated high-dimension distributions, but most models require learning a mapping from noise distribution to data distribution. If the generative procedure starts from an initial distribution proximate to the terminal data distribution, it can remarkably alleviate learning challenges, reduce inference complexity,

Preprint

* Equal Contribution

† Work done as part of an internship at Morgan Stanley

and enhance the quality of generated samples, which is also supported by previous studies [Rubanova et al., 2019, Rasul et al., 2021a,b, Chen et al., 2023b, Deng et al., 2024a,b, Chen et al., 2024]. Time series data is typically continuous and neighboring time points exhibit strong correlations, indicating that the distribution of future time points is close to that of the current time point.

These observations inspire the creation of a time series prediction model under the generative framework that maps between dependent data points: initiating the prediction of future time point’s distribution with the current time point is more straightforward and yields better quality; meanwhile, the longer temporal dependency is encoded by a recurrent neural network and the embedded history is passed to the generative model as the guidance of the prediction for the future time points. The new framework benefits from the efficient training and computation inherited from the recurrent neural network, while enjoying the high quality of probabilistic modeling empowered by the diffusion model.

Our contributions include:

- extending the theory of stochastic interpolants to a more general conditional generation framework with extra control features;
- adopting a conditional stochastic interpolants module for the sequential modeling and multivariate probabilistic time series prediction tasks, which is computational-friendly and achieves high-quality modeling of the future time point’s distribution.

2 BACKGROUND

As we formalize probabilistic time series forecasting within the generative framework in Section 4, this section is dedicated to reviewing commonly used generative methods and their extensions for conditional generation. These models will serve as baseline models in subsequent sections. For a broader overview of time series forecasting problems, refer to Salinas et al. [2019], Alexandrov et al. [2020], and the references therein.

2.1 DENOISING DIFFUSION PROBABILISTIC MODEL (DDPM)

DDPM [Sohl-Dickstein et al., 2015, Ho et al., 2020] adds Gaussian noise to the observed data point $\mathbf{x}^0 \in \mathbb{R}^D$ at different scales, indexed by n , $0 < \beta_1 < \beta_2 < \dots < \beta_N$ such that the first noisy value \mathbf{x}^1 is close to the clean data \mathbf{x}^0 , and the final value \mathbf{x}^N is indistinguishable from noise. The generative model learns to revert this process allowing sampling new points from pure noise samples.

Following previous convention, we define $\bar{\alpha}_n = \prod_{k=1}^n \alpha_k$, with $\alpha_n = 1 - \beta_n$. Then when the transition kernel is

Gaussian it can be computed directly from \mathbf{x}^0 :

$$q(\mathbf{x}^n | \mathbf{x}^0) = \mathcal{N}(\sqrt{\bar{\alpha}_n} \mathbf{x}^0, (1 - \bar{\alpha}_n) \mathbf{I}). \quad (1)$$

The *posterior* distribution is available in closed form:

$$q(\mathbf{x}^{n-1} | \mathbf{x}^n, \mathbf{x}^0) = \mathcal{N}(\tilde{\mu}_n, \tilde{\beta}_n \mathbf{I}), \quad (2)$$

where $\tilde{\mu}_n$ depends on \mathbf{x}^0 , \mathbf{x}^n and a choice of β -scheduler. The generative model $p(\mathbf{x}^{n-1} | \mathbf{x}^n) \approx q(\mathbf{x}^{n-1} | \mathbf{x}^n, \mathbf{x}^0)$ approximates the reverse process. The actual model $\epsilon_\theta(\mathbf{x}^0, n)$ is usually reparameterized to predict the noise added to a clean data point, from the noisy data point \mathbf{x}^n . The loss function can be simply written as:

$$\mathcal{L} = \mathbb{E}_{\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), n \sim \mathcal{U}(\{1, \dots, N\})} [\|\epsilon_\theta(\mathbf{x}^n, n) - \epsilon\|_2^2]. \quad (3)$$

Sampling new data is performed by first sampling a point from the pure noise $\mathbf{x}^N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and then gradually denoising it using the above model to get a sample from the data distribution via N calls of the model [Ho et al., 2020].

2.2 SCORE-BASED GENERATIVE MODEL (SGM)

SGM [Song et al., 2021], like DDPM, considers a pair of forward and backward dynamics between $s \in [0, 1]$:

$$\begin{aligned} d\mathbf{x}^s &= f(\mathbf{x}^s, s)ds + g(s)d\mathbf{w}^s \\ d\mathbf{x}^s &= [f(\mathbf{x}^s, s) - g(s)^2 \nabla_{\mathbf{x}^s} \log p(\mathbf{x}^s)]ds + g(s)d\mathbf{w}^s, \end{aligned} \quad (4)$$

where $\nabla_{\mathbf{x}^s} \log p(\mathbf{x}^s)$ is the so-called score function. The forward process usually is scheduled as simple processes, such as Brownian motion or Ornstein–Uhlenbeck process, which can transport data distribution to standard Gaussian distribution. The generative process is achieved by the backward process that walks from Gaussian prior distribution to the data distribution of interest. Now, Equation 5 gives a way to generate new points by starting at $\mathbf{x}^1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and solving the SDE backward in time giving \mathbf{x}^0 as a sample from data distribution. In practice, the only missing piece is obtaining the score. A standard approach is to approximate the score with a neural network.

Since during training, we have access to clean data, the score function is available in closed form. The model $\epsilon_\theta(\mathbf{x}^s, s)$ learns to approximate the score from noisy data only, resulting in a loss function similar to Equation 3:

$$\mathcal{L} = \mathbb{E}_{s \sim \mathcal{U}(0,1), \mathbf{x}^0 \sim \text{Data}, \mathbf{x}^s \sim p(\mathbf{x}^s | \mathbf{x}^0)} \left[\|\epsilon_\theta(\mathbf{x}^s, s) - \nabla_{\mathbf{x}^s} \log p(\mathbf{x}^s | \mathbf{x}^0)\|_2^2 \right]. \quad (6)$$

2.3 FLOW MATCHING (FM)

Flow matching [Lipman et al., 2023] constructs a probability path by learning the vector field that generates it.

Given a data point \mathbf{x}^1 , the conditional probability path is denoted with $p_s(\mathbf{x}|\mathbf{x}^1)$ for $s \in [0, 1]$. We put the constraints on $p_s(\mathbf{x}|\mathbf{x}^1)$ such that $p_0(\mathbf{x}|\mathbf{x}^1) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $p_1(\mathbf{x}|\mathbf{x}^1) = \mathcal{N}(\mathbf{x}^1, \sigma^2 \mathbf{I})$, with small $\sigma > 0$. That is, the distribution $p_0(\mathbf{x}|\mathbf{x}^1)$ corresponds to the noise distribution and the distribution $p_1(\mathbf{x}|\mathbf{x}^1)$ is centered around the data point with small variance.

Then there exists a conditional vector field $u_s(\mathbf{x}|\mathbf{x}^1)$ which generates $p_s(\mathbf{x}|\mathbf{x}^1)$. Our goal is to learn the vector field with a neural network $\epsilon_\theta(\mathbf{x}, s)$ which amounts to learning the generative process. This can be done by minimizing the flow matching objective:

$$\mathcal{L} = \mathbb{E}_{s \sim \mathcal{U}(0,1), \mathbf{x}^1 \sim \text{Data}, \mathbf{x} \sim p_s(\mathbf{x}|\mathbf{x}^1)} \left[\|\epsilon_\theta(\mathbf{x}, s) - u_s(\mathbf{x}^s|\mathbf{x}^1)\|_2^2 \right]. \quad (7)$$

Going back to Equation 6 we notice that the two approaches have similarities. Flow matching differs in the path constructions and it learns the vector field directly, instead of learning the score, potentially offering a more stable alternative.

One choice for the noising function is transporting the values into noise as a linear function of transport time:

$$\mathbf{x}^s = s\mathbf{x}^1 + (1 - (1 - \sigma)s)\epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \quad (8)$$

The probability path is generated by the following conditional vector field which is available in closed form:

$$u_s(\mathbf{x}|\mathbf{x}^1) = \frac{\mathbf{x}^1 - (1 - \sigma)\mathbf{x}}{1 - (1 - \sigma)s}. \quad (9)$$

By learning the field $u_s(\mathbf{x}|\mathbf{x}^1)$ with a neural network $\epsilon_\theta(\mathbf{x}, s)$ we can sample new points by sampling an initial value \mathbf{x}^0 from the noise distribution p_0 and solve an ODE $0 \mapsto 1$ to obtain the new sample \mathbf{x}^1 .

2.4 STOCHASTIC INTERPOLANTS (SI)

Stochastic interpolants [Albergo et al., 2024] aims to model the *dependent couplings* between $(\mathbf{x}^0, \mathbf{x}^1)$ with their joint density $\rho(\mathbf{x}^0, \mathbf{x}^1)$, and establish a two-way generative SDEs mapping from one data distribution to another. The method constructs a straightforward stochastic mapping from $s = 0$ to $s = 1$ given the values at two ends $\mathbf{x}^0 \sim \rho_0$ to $\mathbf{x}^1 \sim \rho_1$, which provides a means of transport between two densities ρ_0 and ρ_1 , while maintaining the dependency between \mathbf{x}^0 and \mathbf{x}^1 .

$$\mathbf{x}^s = \alpha(s)\mathbf{x}^0 + \beta(s)\mathbf{x}^1 + \gamma(s)\mathbf{z}, \quad s \in [0, 1], \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (10)$$

where $\rho(s, \mathbf{x})$ is the marginal density of \mathbf{x}^s at diffusion time s . Such a stochastic mapping is characterized by a pair of functions: velocity function $\mathbf{b}(s, \mathbf{x})$ and score function

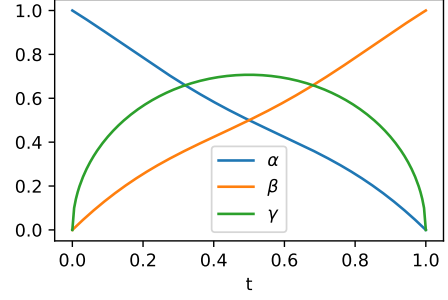


Figure 1: $\alpha(\cdot)$, $\beta(\cdot)$, and $\gamma(\cdot)$, the schedules of stochastic interpolants.

$\mathbf{s}(s, \mathbf{x})$:

$$\mathbf{s}(s, \mathbf{x}) := \nabla \log \rho(s, \mathbf{x}), \quad (11)$$

$$\mathbf{b}(s, \mathbf{x}) := \mathbb{E}_{\mathbf{x}^0, \mathbf{x}^1, \mathbf{z}} [\dot{\alpha}(s)\mathbf{x}^0 + \dot{\beta}(s)\mathbf{x}^1 + \dot{\gamma}(s)\mathbf{z} | \mathbf{x}^s = \mathbf{x}]. \quad (12)$$

$\mathbf{b}(s, \mathbf{x})$, $\rho(s, \mathbf{x})$, and $\mathbf{s}(s, \mathbf{x})$ satisfy the equality below,

$$\partial_t \rho(s, \mathbf{x}) + \nabla \cdot (\mathbf{b}(s, \mathbf{x}) \rho(s, \mathbf{x})) = 0 \quad (13)$$

$$\mathbf{s}(s, \mathbf{x}) = -\gamma^{-1}(s) \mathbb{E}_{\mathbf{z}} [\mathbf{z} | \mathbf{x}^s = \mathbf{x}], \quad (14)$$

where $\alpha(s)$ and $\beta(s)$ schedule the deterministic interpolant. We set $\alpha(0) = 1, \alpha(1) = 0, \beta(0) = 0, \beta(1) = 1$. $\gamma(s)$ schedules the variance of the stochastic component \mathbf{z} . We set $\gamma(0) = \gamma(1) = 0$, so the two ends of the interpolant are fixed at \mathbf{x}^0 and \mathbf{x}^1 . Figure 1 shows one example of the interpolant schedule, where $\alpha(s) = \sqrt{1 - \gamma^2(s)} \cos(\frac{1}{2}\pi s)$, $\beta(s) = \sqrt{1 - \gamma^2(s)} \sin(\frac{1}{2}\pi s)$, $\gamma(s) = \sqrt{2s(1-s)}$.

The velocity function $\mathbf{b}(s, \mathbf{x})$ and the score function $\mathbf{s}(s, \mathbf{x})$ can be modeled by a rich family of functions, such as deep neural networks. The model is trained to match the above equality by minimizing the mean squared error loss functions,

$$\mathcal{L}_b = \int_0^1 \mathbb{E} \left[\frac{1}{2} \|\hat{\mathbf{b}}(s, \mathbf{x}^s)\|^2 - (\dot{\alpha}(s)\mathbf{x}^0 + \dot{\beta}(s)\mathbf{x}^1 + \dot{\gamma}(s)\mathbf{z})^T \hat{\mathbf{b}}(s, \mathbf{x}^s) \right] ds \quad (15)$$

$$\mathcal{L}_s = \int_0^1 \mathbb{E} \left[\frac{1}{2} \|\hat{\mathbf{s}}(s, \mathbf{x}^s)\|^2 + \gamma^{-1}(s) \mathbf{z}^T \hat{\mathbf{s}}(s, \mathbf{x}^s) \right] ds. \quad (16)$$

More details of training will be shown in section 4.

During inference, usually, one side of the diffusion trajectory at $s = 0$ or $s = 1$ is given, the goal is to infer the sample distribution on the other side. The interpolant in equation 10 results in elegant forward and backward SDEs and corresponding Fokker-Planck equations, which offer convenient tools for inference. The SDEs are composed of $\mathbf{b}(s, \mathbf{x}^s)$ and $\mathbf{s}(s, \mathbf{x}^s)$, which are learned from the data. For

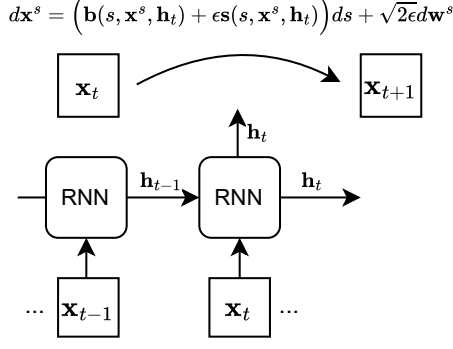


Figure 2: Stochastic interpolants for time series prediction using forward SDE in equation 22.

any $\epsilon(s) \geq 0$, define the forward and backward SDEs

$$dx^s = [b(s, x) + \epsilon(s)s(s, x)]ds + \sqrt{2\epsilon(s)}dw^s \quad (17)$$

$$dx^s = [b(s, x) - \epsilon(s)s(s, x)]ds + \sqrt{2\epsilon(s)}dw_B^s, \quad (18)$$

where w_B^s is the backward Brownian motion. The SDEs satisfy the forward and backward Fokker-Plank equations,

$$\partial_s \rho + \nabla \cdot (b_F \rho) = \epsilon(s) \Delta \rho, \rho(0) = \rho_0 \quad (19)$$

$$\partial_s \rho + \nabla \cdot (b_B \rho) = -\epsilon(s) \Delta \rho, \rho(1) = \rho_1. \quad (20)$$

These properties imply that one can draw samples from the conditional density $\rho(x^1|x^0)$ following the forward SDE in equation 17 starting from x_0 at $s = 0$. It can also draw samples from the joint density $\rho(x^0, x^1)$ by initially drawing a sample $x^0 \sim \rho_0$ (if feasible, for example, pick one sample from the dataset), then using the forward SDE to generate samples x^1 at $s = 1$. The method guarantees that x^1 follows marginal distribution ρ_1 and the sample pair (x^0, x^1) satisfies the joint density $\rho(x^0, x^1)$. Drawing samples using the backward SDE is similar: one can draw samples from $\rho(x^0|x^1)$ and the joint density $\rho(x^0, x^1)$ as well. Details of inference will be shown in section 4.

3 CONDITIONAL GENERATION WITH EXTRA FEATURES

All the aforementioned methods can be adapted for conditional generation with additional features. The conditions may range from simple categorical values [Song et al., 2021] to complex prompts involving multiple data types, including partial observations of a sample’s entries (e.g., image inpainting, time series imputation) [Tashiro et al., 2021, Song et al., 2021], images [Zheng et al., 2023, Rombach et al., 2022], text [Rombach et al., 2022, Zhang et al.], etc. A commonly employed technique to handle diverse conditions is to integrate condition information through feature embedding, where the embedding is injected into various layers of neural networks [Song et al., 2021, Rombach et al.,

2022]. For instance, conditional SGM can be trained with

$$\mathcal{L}_{\text{cond}} = \mathbb{E}_{s \sim \mathcal{U}(0,1), (x^0, \xi) \sim \text{Data}, x^s \sim p(x^s|x^0)} \left[\|\epsilon_\theta(s, x^s, \xi) - \nabla_{x^s} \log p(x^s|x^0)\|_2^2 \right]. \quad (21)$$

where the data is given by pairs of a sample x^0 and the corresponding condition ξ . This simple scheme approach showcases its effectiveness in various tasks, achieving state-of-the-art performance [Rombach et al., 2022, Zhang et al.].

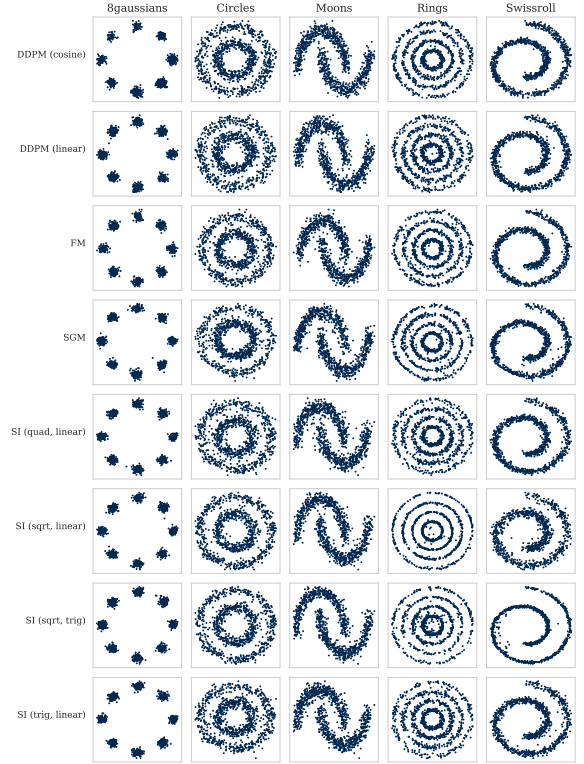


Figure 3: Examples of model generated samples for synthetic two-dimensional ($D = 2$) datasets.

Likewise, SI can be expanded for conditional generation by substituting the velocity function and score function with $b(x^s, s, \xi)$ and $s(x^s, s, \xi)$ [Albergo et al., 2024]. The model is trained using samples of tuples (x^0, x^1, ξ) , where ξ is the extra condition feature. Consequently, the inference using forward or backward SDEs becomes

$$dx^s = [b(s, x^s, \xi) + \epsilon(s)s(s, x^s, \xi)]ds + \sqrt{2\epsilon(s)}dw^s \quad (22)$$

$$dx^s = [b(s, x^s, \xi) - \epsilon(s)s(s, x^s, \xi)]ds + \sqrt{2\epsilon(s)}dw_B^s, \quad (23)$$

where both velocity and score functions depend on the condition ξ . The loss functions are similar to equation 15 and equation 16.

Regarding the time series prediction task, we will encode a large context window as the conditional information, and

Algorithm 1 Training algorithm.

Input: Sample $\mathbf{x}_{1:C+P}$ from training split. Interpolant schedules: $\alpha(s), \beta(s), \gamma(s)$. Models: velocity $\hat{\mathbf{b}}$, score $\hat{\mathbf{s}}$, RNN.

for iteration $t = C$ **to** $C + P - 1$ **do**
 $s \sim \text{Beta}(0.1, 0.1)$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
 $\mathbf{x}^s = \alpha(s)\mathbf{x}_t + \beta(s)\mathbf{x}_{t+1} + \gamma(s)\mathbf{z}$
 $\mathbf{h}_t = \text{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1})$

$$\begin{aligned}\mathcal{L}_b &= \frac{1}{p_{\text{Beta}}(s)} \left[\frac{1}{2} \|\hat{\mathbf{b}}(s, \mathbf{x}^s, \mathbf{h}_t)\|^2 \right. \\ &\quad \left. - (\dot{\alpha}(s)\mathbf{x}_t + \dot{\beta}(s)\mathbf{x}_{t+1} + \dot{\gamma}(s)\mathbf{z})^T \hat{\mathbf{b}}(s, \mathbf{x}^s, \mathbf{h}_t) \right] \\ \mathcal{L}_s &= \frac{1}{p_{\text{Beta}}(s)} \left[\frac{1}{2} \|\hat{\mathbf{s}}(s, \mathbf{x}^s)\|^2 + \gamma^{-1} \mathbf{z}^T \hat{\mathbf{s}}(s, \mathbf{x}^s) \right]\end{aligned}$$

Perform back-propagation by minimizing \mathcal{L}_b and \mathcal{L}_s .
end for

the prediction or generation of future time points will rely on such a conditional generation mechanism.

Next, we demonstrate that the probability distribution of \mathbf{x}^s as simulated by equation 24, results in a dynamic density function. This density serves as a solution to a transport equation 25, which smoothly transitions between ρ_0 and ρ_1 .

Theorem 1. (*Extension of Stochastic Interpolants to Arbitrary Joint Distributions*). Let ρ_{01} be the joint distribution $(\mathbf{x}^0, \mathbf{x}^1) \sim \rho_{01}$ and let the stochastic interpolant be

$$\mathbf{x}^s = \alpha_s \mathbf{x}^0 + \beta_s \mathbf{x}^1 + \gamma_s \mathbf{z}, \quad (24)$$

where $\alpha_0 = \beta_1 = 1, \alpha_1 = \beta_0 = \gamma_0 = \gamma_1 = 0$, and $\alpha_s^2 + \beta_s^2 + \gamma_s^2 > 0$ for all $s \in [0, 1]$. We define ρ_s to be the noise-dependent density of \mathbf{x}^s , which satisfies the boundary conditions at $s = 0, 1$ and the transport equation follows that

$$\dot{\rho}_s + \nabla \cdot (\mathbf{b}_s \rho_s) = 0 \quad (25)$$

for all $s \in [0, 1]$ with the velocity defined as

$$\mathbf{b}_s(\mathbf{x}|\xi) = \mathbb{E} \left[\dot{\alpha}_s \mathbf{x}^0 + \dot{\beta}_s \mathbf{x}^1 + \dot{\gamma}_s \mathbf{z} | \mathbf{x}^s = \mathbf{x}, \xi \right], \quad (26)$$

where the expectation is based on the density ρ_{01} given $\mathbf{x}^s = \mathbf{x}$ and the extra information ξ .

The score function follows the relation such that

$$\nabla \log \rho_s(\mathbf{x}) = -\gamma_s^{-1} \mathbb{E} [\mathbf{z} | \mathbf{x}^s = \mathbf{x}, \xi].$$

The proof is in a spirit similar to Theorem 2 in Albergo et al. [2024] and detailed in section B. The key difference is that we consider a continuous-time interpretation and avoid using characteristic functions, which makes the analysis more friendly to users. Additionally, the score function $\nabla \log \rho_s(\mathbf{x})$ is optimized in a simple quadratic objective function as indicated in Theorem 2 in the Appendix.

Algorithm 2 Inference algorithm.

Input: Last context $\mathbf{x}_{1:C}$. Trained models: Velocity $\hat{\mathbf{b}}$, score $\hat{\mathbf{s}}$, RNN. Diffusion variance $\epsilon(s)$.

for iteration $t = C$ **to** $C + P - 1$ **do**
Set $\hat{\mathbf{x}}^0 = \mathbf{x}_t$ and $\mathbf{h}_t = \text{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1})$.
Run SDE integral for $s \in [0, 1]$ following

$$d\hat{\mathbf{x}}^s = [\hat{\mathbf{b}}(s, \hat{\mathbf{x}}^s, \mathbf{h}_t) + \epsilon(s)\hat{\mathbf{s}}(s, \hat{\mathbf{x}}^s, \mathbf{h}_t)]ds + \sqrt{2\epsilon}d\mathbf{w}^s$$

Output: $\hat{\mathbf{x}}^1$ as prediction: \mathbf{x}_{t+1} .
end for

4 STOCHASTIC INTERPOLANTS FOR TIME SERIES PREDICTION

We formulate the multivariate probabilistic time series prediction tasks through the conditional probability $\Pi_{t=C+1}^{C+P} p(\mathbf{x}_t | \mathbf{x}_{1:t-1})$ for some chosen context length C and prediction length P . The model diagram is illustrated in Figure 2. Here, $\mathbf{x}_t \in \mathbb{R}^D$ represents the multivariate time series at *date-time index* t with $D > 1$ variates. $\mathbf{x}_{1:C} = \mathbf{X}_{\text{cont.}}$ is the context window and during training the subsequent prediction window $\mathbf{x}_{C+1:C+P} = \mathbf{X}_{\text{pred.}}$ is available, typically sampled randomly from within the train split of a dataset.

For this problem, we employ the conditional Stochastic Interpolants (SI) method as follows. In the *training phase*, the generative model learns the joint distribution $p(\mathbf{x}_{t+1}, \mathbf{x}_t | \mathbf{x}_{1:t-1})$ of the pair $(\mathbf{x}_{t+1}, \mathbf{x}_t)$ given the past observations $\mathbf{x}_{1:t-1}$, where $\mathbf{x}_t \sim \rho_0$ and $\mathbf{x}_{t+1} \sim \rho_1$ for all t , so the marginal distributions are equal: $\rho_0 = \rho_1$. The model aims to learn the coupling relation between \mathbf{x}_{t+1} and \mathbf{x}_t conditioning on the history $\mathbf{x}_{1:t}$. This is achieved by training the conditional velocity and score functions in equation 22.

	8gaussians	Circles	Moons	Rings	Swissroll
DDPM (cosine)	2.58	0.20	0.20	0.12	0.24
DDPM (linear)	0.70	0.18	0.12	0.11	0.14
SGM	1.10	0.30	0.35	0.32	0.14
FM	0.58	0.10	0.11	0.09	0.15
SI (quad, linear)	0.52	0.15	0.32	0.12	0.16
SI (sqrt, linear)	0.59	0.29	0.51	0.22	0.37
SI (sqrt, trig)	0.75	0.25	0.50	0.48	0.36
SI (trig, linear)	0.52	0.13	0.29	0.21	0.16

Table 1: Wasserstein distance between the generated samples and true data.

As the sample spaces of ρ_0 and ρ_1 must be the same, the generative model can not directly map the whole context window $\mathbf{x}_{1:t}$ to the target \mathbf{x}_{t+1} due to different tensor sizes. Instead, a recurrent neural network is used to encode the context $\mathbf{x}_{1:t}$ into a *history prompt* $\mathbf{h}_t \in \mathbb{R}^H$ vector. Subsequently, the score function and velocity function perform conditional generation diffusing from \mathbf{x}_t with the condition

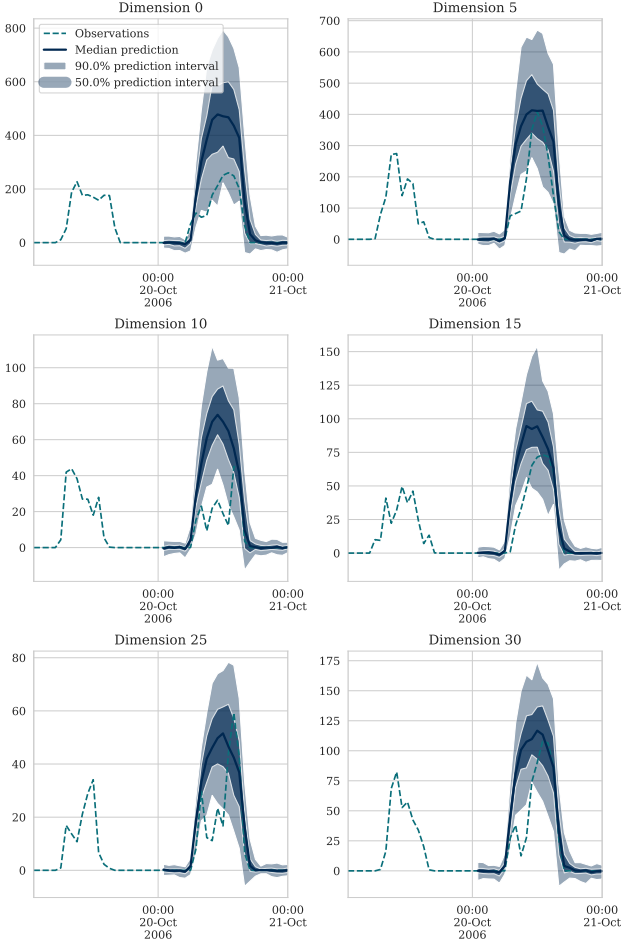


Figure 4: Forecast paths for SI on Solar dataset showing median prediction, 50th and 90th confidence intervals calculated from model samples, on 6 / 137 variate dimensions.

input \mathbf{h}_t following equation 22.

The training loss consists of tuples $(\mathbf{x}_{t+1}, \mathbf{x}_t, \mathbf{h}_t)$ for each time step t . It is worth noting that the loss values become larger when s is close to the two ends. To address this, importance sampling is leveraged to better handle the integral over diffusion time in the loss functions equation 15 and equation 16 to stabilize the training, where we use Beta distribution for our proposal distribution. The algorithm is outlined in Algorithm 1. Additional details can be found in Appendix C.

In the *inference phase*, the RNN first encodes the context $\mathbf{x}_{1:t}$ into the history prompt \mathbf{h}_t , then SI transports the context vector \mathbf{x}_t to the target distribution with the condition \mathbf{h}_t , following the forward SDE. Regarding the multiple-step prediction, we recursively run the step-by-step prediction in an autoregressive manner as outlined in Algorithm 2. By repeating the inference loop (in the batch dimension) we can obtain empirical samples from the predicted distribution which are used to quantify uncertainty.

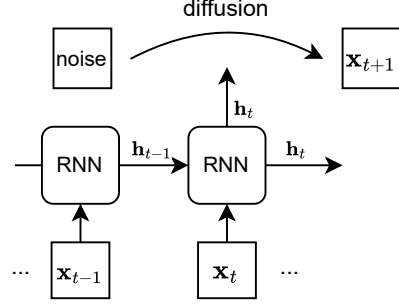


Figure 5: Time-Grad [Rasul et al., 2021a] model for conditional time series prediction as a comparison.

5 EXPERIMENTS

We first verify the method on synthetic datasets and then apply it to the time series forecasting tasks with real data.

Baseline models such as DDPM, SGM, FM, and SI all involve modeling field functions, where the inputs are the state vector (in the same space of the data samples), diffusion time, and condition embedding, and the output is the generated sample. The field functions correspond to the noise prediction function in DDPM; the score function in SGM; the vector field in FM; and the velocity and score functions in SI. To make a fair comparison between these models, we use the same neural networks for these models. Details of the models are discussed in Appendix C.

5.1 SYNTHETIC DATASETS

We synthesize several two-dimensional datasets with regular patterns, such as Circles, Moons, etc. Details can be found in Chen et al. [2022], Lipman et al. [2023] and their published code repositories. Models introduced in section 2 are compared to the SI as baselines. For diffusion-like models, we implement DPPM with a linear or cosine noise scheduler. We explore on synthetic datasets to determine a *good* range of hyperparameters which will be used in later time series experiments. This experiment is used to investigate the properties with respect to the varying data sizes, model sizes, and training lengths.

To fairly compare the generation quality, all models are assigned to generate data in the same setting by mapping from standard Gaussian to the target distribution. The neural networks and hyperparameters are also set as the same, such as batch size, training epochs, etc. The generated samples from different methods are shown in Figure 3. Table 1 measures the sample quality with Wasserstein distance [Ramdas et al., 2017]. It shows that all the models can capture the true distribution. The same holds when we use different metrics such as Sliced Wasserstein Distance (SWD) [Rabin et al., 2012] and Maximum Mean Discrepancy (MMD) [Gretton

	Exchange rate	Solar	Traffic	Wiki
Vec-LSTM	0.008 \pm 0.001	0.391 \pm 0.017	0.087 \pm 0.041	0.133 \pm 0.002
DDPM	0.009 \pm 0.004	0.359 \pm 0.061	0.058 \pm 0.014	0.084 \pm 0.023
FM	0.009 \pm 0.001	0.419 \pm 0.027	0.038 \pm 0.002	64.256 \pm 62.596
SGM	0.008 \pm 0.002	0.364 \pm 0.029	0.071 \pm 0.05	0.108 \pm 0.026
SI	0.007 \pm 0.001	0.359 \pm 0.06	0.083 \pm 0.005	0.080 \pm 0.007

Table 2: CRPS-sum metric on multivariate probabilistic forecasting tasks. A smaller number indicates better performance.

et al., 2012].

We also test out different schedulers for a stochastic interpolant model. For example, “SI (sqrt, linear)” means we use square root gamma-function $\gamma(s) = \sqrt{2s(1-s)}$ and a linear interpolant. Other gamma-functions that we consider are *quad*: $\gamma(s) = s(1-s)$, and *trig*: $\gamma(s) = \sin^2(\pi s)$. We show that most of the gamma-interpolant function combinations achieve good results in modeling the target distribution.

5.2 MULTIVARIATE PROBABILISTIC FORECASTING

In this section, we will empirically verify that: 1) SI is a suitable generative module for the prediction compared with other baselines with different generative methods under the same framework; 2) the whole framework can achieve competitive performance in time series forecasting.

Models. The baseline models include DDPM, SGM, and FM-based generative models adopted for step-by-step (autoregressive) prediction. DDPM and SGM-based models can only generate samples by transporting Gaussian noise distribution to data distribution. So we modify the framework by replacing the context time point \mathbf{x}_t with Gaussian noise, as shown in Figure 5. Flow matching can easily fit into this framework by replacing the denoising objective with the flow matching objective. The modified framework is shown in Figure 2. We model the map from the previous time series observation to the next (forecasted) value. We argue this is a more natural choice than mapping from noise for each time series prediction step. Finally, Vec-LSTM from Salinas et al. [2019] is compared as a pure recurrent neural network model whose probabilistic layer is a multivariate Gaussian.

Setup. The real-world time series datasets include Solar [Lai et al., 2018], Exchange [Lai et al., 2018], Traffic¹, and Wiki² which have been commonly used for probabilistic forecasting tasks. We follow the preprocessing steps as in

¹<https://archive.ics.uci.edu/ml/datasets/PEMS-SF>

²https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets

Salinas et al. [2019]. The probabilistic forecasting is evaluated by Continuous Ranked Probability Score (CRPS-sum) [Koochali et al., 2022], normalized root mean square error via the median of the samples (NRMSE), and point-metrics normalized deviance (ND). The metrics calculation is provided by `gluonts` package [Alexandrov et al., 2020]. In all of the cases, smaller values indicate better performance.

Results. The results for CRPS-sum are shown in Table 2. The results for other metrics are consistent with CRPS-sum and are shown in Tables 4 and 5, in Appendix C. We outperform or match other models on three out of four datasets, only on Traffic FM model achieves better performance. Note that on Wiki data FM cannot capture the data distribution. We ran a search over flow matching hyperparameters without being able to get satisfying results. Therefore, we conclude that stochastic interpolants are a strong candidate for conditional generation, in particular for multivariate probabilistic forecasting. By comparing to the RNN-based model Vec-LSTM, our model and other baselines such as SGM and DDPM get better performance, which implies that carefully modeling the probability distribution is critical for large dimension time series prediction. Figure 4 demonstrates the quality of the forecast on the Solar dataset. We can see that our model can make precise predictions and capture the uncertainty, even when the scale of the different dimensions varies considerably.

6 CONCLUSIONS

This study presents an innovative method that effectively merges the computational efficiency of recurrent neural networks with the high-quality probabilistic modeling of the diffusion model, specifically applied to probabilistic time series forecasting. Grounded in stochastic interpolants and an expanded conditional generation framework featuring control features, the method undergoes empirical evaluation on both synthetic and real datasets, showcasing its compelling performance.

References

- Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic Interpolants: A Unifying Framework for Flows and Diffusions. In *arXiv:2303.08797v3*, 2023.
- Michael Samuel Albergo, Mark Goldstein, Nicholas Matthew Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=FFILRGD0jG>.
- Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. Gluonts: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL <http://jmlr.org/papers/v21/19-820.html>.
- Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling Temporal Data as Continuous Functions with Process Diffusion. In *Proc. of the International Conference on Machine Learning (ICML)*, 2023.
- George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. WILEY, 2015.
- Tianrong Chen, Guan-Hong Liu, and Evangelos Theodorou. Likelihood training of schrödinger bridge using forward-backward SDEs theory. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=nioAdKCedXB>.
- Yifan Chen, Mark Goldstein, Mengjian Hua, Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Probabilistic Forecasting with Stochastic Interpolants and Föllmer Processes. In *Proc. of the International Conference on Machine Learning (ICML)*, 2024.
- Yu Chen, Wei Deng, Shikai Fang, Fengpei Li, Nicole Tianjiao Yang, Yikai Zhang, Kashif Rasul, Shandian Zhe, Anderson Schneider, and Yuriy Nevmyvaka. Provably Convergent Schrödinger Bridge with Applications to Probabilistic Time Series Imputation. In *International Conference on Machine Learning (ICML)*, 2023a.
- Zehua Chen, Guande He, Kaiwen Zheng, Xu Tan, and Jun Zhu. Schrodinger Bridges Beat Diffusion Models on Text-to-Speech Synthesis. *arXiv preprint arXiv:2312.03491*, 2023b.
- Zonglei Chen, Minbo Ma, Tianrui Li, Hongjun Wang, and Chongshou Li. Long Sequence Time-Series Forecasting with Deep Learning: A Survey. In *Information Fusion*, 2023c.
- Wei Deng, Yu Chen, Nicole Tianjiao Yang, Hengrong Du, Qi Feng, and Ricky T. Q. Chen. Reflected Schrödinger Bridge for Constrained Generative Modeling. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024a.
- Wei Deng, Weijian Luo, Yixin Tan, Marin Biloš, Yu Chen, Yuriy Nevmyvaka, and Ricky T. Q. Chen. Variational Schrödinger Diffusion Models. In *Proc. of the International Conference on Machine Learning (ICML)*, 2024b.
- Alex Graves. Generating Sequences with Recurrent Neural Networks. 2013.
- Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 1997.
- Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. Random noise vs. state-of-the-art probabilistic forecasting methods: A case study on crps-sum discrimination ability. *Applied Sciences*, 12(10):5104, 2022.
- Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.
- Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable Gradients for Stochastic Differential Equations. In *Proc. of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.
- Yan Li, Xinjiang Lu, Yaqing Wan, and Dejing Do. Generative Time Series Forecasting with Diffusion, Denoise, and Disentanglement. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=PqvMRDCJT9t>.

- Juan Miguel, Lopez Alcaraz, and Nils Strodthoff. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. In *Transactions on Machine Learning Research*, 2022.
- James Morrill, Cristopher Salvi, Patrick Kidger, James Foster, and Terry Lyons. Neural Rough Differential Equations for Long Time Series. In *Proc. of the International Conference on Machine Learning (ICML)*, 2021.
- Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. Wasserstein barycenter and its application to texture mixing. In *Scale Space and Variational Methods in Computer Vision: Third International Conference, SSVN 2011, Ein-Gedi, Israel, May 29–June 2, 2011, Revised Selected Papers 3*, pages 435–446. Springer, 2012.
- Aaditya Ramdas, Nicolás García Trillos, and Marco Cuturi. On wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47, 2017.
- Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive Denoising Diffusion Models for Multivariate Probabilistic Time Series Forecasting. In *International Conference on Machine Learning*, pages 8857–8868. PMLR, 2021a.
- Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs Bergmann, and Roland Vollgraf. Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In *Proc. of the International Conference on Learning Representation (ICLR)*, 2021b.
- Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.
- Yulia Rubanova, Ricky T. Q. Chen, and David Duvenaud. Latent ODEs for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- David Salinas, Michael Bohlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional Multivariate Forecasting with Low-rank Gaussian Copula Processes. *Advances in neural information processing systems*, 32, 2019.
- Vladislav Shishkov. Time Diffusion. In <https://github.com/timetoai/TimeDiffusion>, 2023.
- Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.
- Alessio Spantini, Ricardo Baptista, and Youssef Marzouk. Coupling Techniques for Nonlinear Ensemble Filtering. In *SIAM Review*, volume 64:4, page 10.1137, 2022.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *NIPS*, 2014.
- Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in Time Series: a Survey. *Thirty-Second International Joint Conference on Artificial Intelligence*, 2023.
- Chenshuang Zhang, Chaoning Zhang, Mengchun Zhang, and In So Kweon. Text-to-image diffusion models in generative ai: A survey. *arXiv preprint arXiv:2303.07909*, 2023.
- Guangcong Zheng, Xianpan Zhou, Xuwei Li, Zhongang Qi, Ying Shan, and Xi Li. Layoutdiffusion: Controllable diffusion model for layout-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22490–22499, 2023.

A RELATED WORKS

A plethora of papers focus on auto-regression models, particularly transformer-based models. For a more comprehensive review, we refer to Wen et al. [2023], Chen et al. [2023c]. While our work does not aim to replace RNN- or transformer-based architectures, we emphasize that one of the main motivations behind our work is to develop a probabilistic module building upon these recent advancements. Due to limited resources, we did not extensively explore all underlying temporal architectures but instead selected relatively simpler models as defaults.

The authors were aware of other diffusion-based probabilistic models, as highlighted in the introduction. Unlike our lightweight model, which models the transition between adjacent time points, these selected works model the entire time window, requiring both high memory and computational complexity. With our computation budget restricted to a 32 GB GPU device, effectively training these diffusion models on large datasets with hundreds of features is challenging.

Additionally, several relevant works are related to our idea. For instance, Rasul et al. [2021b] incorporates the DDPM structure, aligning with our DDPM baseline structure. During inference, the prediction diffuses from pure noise to the target distribution. TimeDiff [Shishkov, 2023] introduces two modifications to the established diffusion model: during training it mixes target and context data and it adds an AR model for more precise initial prediction. Both of these can be incorporated into our model as well.

The existing probabilistic forecasters model the distribution of the next value from scratch, meaning they start with normal distribution in the case of normalizing flows and diffusion models or output parametric distribution in the case of transformers and deep AR. We propose modeling the transformation between the previously observed value and the next value we want to predict. We believe this is a more natural way to forecast which can be seen from requiring fewer solver steps to reach the target distribution.

The second row (DDPM) in Table 2 is an exact implementation of Rasul et al. [2021b]. The results might be different due to slightly different training setups but all the models share the same training parameters so the rank should remain the same. We also include ND-sum and NRMSE-sum in the appendix for completeness.

Discussion of Vec-LSTM baseline In terms of the neural network architecture, we use a similar architecture for the LSTM encoder. But to be clear, Vec-LSTM [Salinas et al., 2019] and our SI framework are not the same mainly due to different ways of probabilistic modeling. Vec-LSTM considers the multivariate Gaussian distribution for the time points, where the mean and covariance matrices are modeled using separate LSTMs. Especially, the covariance matrix is modeled through a low-dimensional structure $\Sigma(\mathbf{h}_t) = \mathbf{D}_t(\mathbf{h}_t) + \mathbf{V}_t(\mathbf{h}_t)\mathbf{V}_t(\mathbf{h}_t)^\top$, where \mathbf{h}_t is the latent variable from LSTM. The SI framework does not explicitly model the output distribution in any parametric format. Instead, the latent variable from RNN output is used as the condition variable to guide the diffusion model in Eq.22. Thus, the architectures of RNNs in the two frameworks are not quite strictly comparable.

B PROOF: CONDITIONAL STOCHASTIC INTERPOLANT

The proof is in spirit similar to Theorem 2 in Albergo et al. [2024]. The key difference is that we consider a continuous-time interpretation, which makes the analysis more friendly to users.

Proof [Proof of Theorem 1]

Given the conditional information ξ and $\mathbf{x}^s = \mathbf{x}$ simulated from equation 24, the conditional stochastic interpolant for equation 24 follows that (where the index t is over the noise index and *not* date-times):

$$\mathbb{E}[\mathbf{x}^t | \mathbf{x}^s = \mathbf{x}, \xi] = \mathbb{E}[\alpha_t \mathbf{x}^0 + \beta_t \mathbf{x}^1 + \gamma_t \mathbf{z} | \mathbf{x}^s = \mathbf{x}, \xi], \quad (27)$$

where the expectation takes over the density for $(\mathbf{x}^0, \mathbf{x}^1) \sim \rho(\mathbf{x}_0, \mathbf{x}_1 | \xi)$, $\xi \sim \eta(\xi)$, and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

We next show equation 27 is a solution of a stochastic differential equation as follows

$$d\mathbb{E}[\mathbf{x}^t | \mathbf{x}^s = \mathbf{x}, \xi] = \mathbf{f}_t(\mathbf{x})dt + \sigma_t d\mathbf{w}^t, \quad (28)$$

where $\mathbf{f}_t(\mathbf{x}) = \mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 | \mathbf{x}^s = \mathbf{x}, \xi]$ and $\sigma_t = \sqrt{2\gamma_t \dot{\gamma}_t}$.

To prove the above argument, we proceed to verify the *drift* and *diffusion* terms respectively:

- *Drift*: It is straightforward to verify the drift \mathbf{f}_t by taking the gradient of the conditional expectation $\mathbb{E}[\alpha_t \mathbf{x}^0 + \beta_t \mathbf{x}^1 | \mathbf{x}^s = \mathbf{x}, \xi]$ with respect to t .
- *Diffusion*: For the diffusion term, the proof hinges on showing $\sigma_t = \sqrt{2\gamma_t \dot{\gamma}_t}$, which boils down to prove the stochastic calculus follows that $\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} d\mathbf{w}^s = \gamma_t \mathbf{z}$. Note that $\mathbb{E}[\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} d\mathbf{w}^s] = 0$. Invoking the Itô isometry, we have $\text{Var}(\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} d\mathbf{w}^s) = \int_0^t 2\gamma_s \dot{\gamma}_s ds = \int_0^t (\gamma_s^2)' ds = \gamma_t^2$ (given $\gamma_0 = 0$). In other words, $\int_0^t \sqrt{2\gamma_s \dot{\gamma}_s} d\mathbf{w}^s$ is a normal random variable with mean 0 and variance γ_t^2 , which proves that equation 27 is a solution of the stochastic differential equation 28.

Define $\Sigma_t = 2\gamma_t \dot{\gamma}_t$, we know the Fokker-Planck equation associated with equation 28 follows that

$$\begin{aligned}
0 &= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\mathbf{f}_t \rho_t - \frac{1}{2} \Sigma_t \nabla \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\left(\mathbf{f}_t - \frac{1}{2} \Sigma_t \nabla \log \rho_t \right) \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot \left(\left(\mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 | \mathbf{x}^s = \mathbf{x}, \xi] - \gamma_t \dot{\gamma}_t \nabla \log \rho_t \right) \rho_t \right) \\
&= \frac{\partial \rho_t}{\partial t} + \nabla \cdot (b_{t|s}(\mathbf{x}, \xi) \rho_t),
\end{aligned} \tag{29}$$

where $b_{t|s}(\mathbf{x}|\xi) = \mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathbf{x}^s = \mathbf{x}, \xi]$.

Further setting $s = t$ and rewrite $b_t \equiv b_{t|t}$, we have $b_t(\mathbf{x}|\xi) = \mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathbf{x}^t = \mathbf{x}, \xi]$.

Further define $g_t^{(i)}(\mathbf{x}|\xi) = \mathbb{E}[\mathbf{x}^i | \mathbf{x}^t = \mathbf{x}, \xi]$, where $i \in \{0, 1\}$ and $g_t^{(z)}(\mathbf{x}|\xi) = \mathbb{E}[\mathbf{z} | \mathbf{x}^t = \mathbf{x}, \xi]$. We have that

$$\begin{aligned}
b_t(\mathbf{x}|\xi) &= \mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 - \gamma_t \dot{\gamma}_t \nabla \log \rho_t | \mathbf{x}^t = \mathbf{x}, \xi] \\
&= \dot{\alpha}_t g^{(0)} + \dot{\beta}_t g^{(1)} + \dot{\gamma}_t g^{(z)} \\
&= \mathbb{E}[\dot{\alpha}_t \mathbf{x}^0 + \dot{\beta}_t \mathbf{x}^1 + \dot{\gamma}_t \mathbf{z} | \mathbf{x}^t = \mathbf{x}, \xi],
\end{aligned}$$

where the first equality follows by equation 29 and the last one follows by taking derivative to equation 27 w.r.t. the index t .

We also observe that $\nabla \log \rho_t = -\gamma_t^{-1} \mathbb{E}[\mathbf{z} | \mathbf{x}^t = \mathbf{x}]$.

□

Theorem 2. *The loss functions used for estimating the vector field follow that*

$$L_i(\hat{g}^{(i)}) = \int_0^1 \mathbb{E}[|\hat{g}^{(i)}|^2 - 2\mathbf{x}^i \cdot \hat{g}^{(i)}] dt,$$

where $i \in \{0, 1, z\}$, the expectation takes over the density for $(\mathbf{x}^0, \mathbf{x}^1) \sim \rho(\mathbf{x}^0, \mathbf{x}^1 | \xi)$, $\xi \sim \eta(\xi)$, and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Proof To show the loss is effective to estimate $g^{(0)}$, $g^{(1)}$, and $g^{(z)}$. It suffices to show

$$\begin{aligned}
L_0(\hat{g}^{(0)}) &= \int_0^1 \mathbb{E}[|\hat{g}^{(0)}|^2 - 2\mathbf{x}^0 \cdot \hat{g}^{(0)}] dt, \\
&= \int_0^1 \int_{\mathbb{R}^D} \left[|\hat{g}^{(0)}|^2 - 2\mathbb{E}[\mathbf{x}^0 | \mathbf{x}^t = \mathbf{x}, \xi] \cdot \hat{g}^{(0)} \right] d\mathbf{x} dt, \\
&= \int_0^1 \int_{\mathbb{R}^D} \left[|\hat{g}^{(0)}|^2 - 2g^{(0)} \cdot \hat{g}^{(0)} \right] d\mathbf{x} dt,
\end{aligned}$$

where the last equality follows by definition. The unique minimizer is attainable by setting $\hat{g}^{(0)} = g^{(0)}$.

The proof of $g^{(1)}$ and $g^{(z)}$ follows a similar fashion.

C EXPERIMENT DETAILS

C.1 TIME SERIES DATA

The time series datasets include: Solar [Lai et al., 2018], Exchange [Lai et al., 2018], Traffic³, and Wikipedia⁴. We follow the preprocessing steps as in Salinas et al. [2019]. Details of the datasets are listed in Table 3.

Table 3: Properties of the datasets.

Datasets	Dimension D	Frequency	Total time points	Prediction length P
Exchange	8	Daily	6,071	30
Solar	137	Hourly	7,009	24
Traffic	963	Hourly	4,001	24
Wiki	2000	Daily	792	30

The probabilistic forecasting is evaluated by Continuous Ranked Probability Score (CRPS-sum) [Koochali et al., 2022], normalized root mean square error via the median of the samples (NRMSE), and point-metrics normalized deviance (ND). The metrics calculation is provided by `gluonts` package [Alexandrov et al., 2020] by calling module `gluonts.evaluation.MultivariateEvaluator`.

C.2 MODELS AND HYPERPARAMETERS

Baseline models such as DDPM, SGM, FM, and SI all involve modeling field functions, where the inputs are the state vector (in the same space as the data samples), diffusion time, and condition embedding, and the output is the generated sample. The field functions correspond to the “noise prediction” function in DDPM; the score function in SGM; the vector field in FM; the velocity and score functions in SI. To make a fair comparison between these models, we use the same neural networks for these models.

In the synthetic datasets experiments, we model the field functions with a 4-layer ResNet, each layer has 256 intermediate dimensions. The batch size is 10,000 for all models and a model is trained with 20,000 iterations. The learning rate is 10^{-3} .

In the time series forecasting experiments, the RNN for the history encoder has 1 layer and 128 latent dimensions; The field function is modeled with a Unet-like structure [Ronneberger et al., 2015] with 8 residual blocks, and each block has 64 dimensions. To stabilize the training, we also use paired sampling for the stochastic interpolants introduced by [Albergo et al., 2023, Appendix C]:

$$\begin{aligned}\mathbf{x}^s &= \alpha(s)\mathbf{x}^0 + \beta(s)\mathbf{x}^1 + \gamma(s)\mathbf{z} \\ \mathbf{x}^{s'} &= \alpha(s)\mathbf{x}^0 + \beta(s)\mathbf{x}^1 + \gamma(s)(-\mathbf{z}) \\ s &\in [0, 1], \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).\end{aligned}$$

The baseline models are trained with 200 epochs and 64 batch sizes with a learning rate 10^{-3} . The SI model is trained with 100 epochs and 128 batch sizes with a learning rate 10^{-4} . We find if the learning rate is too large, SI may not converge properly.

C.3 IMPORTANCE SAMPLING

The loss functions for training the velocity and score functions are

$$\begin{aligned}\mathcal{L}_b &= \int_0^1 \mathbb{E} \left[\frac{1}{2} \|\hat{\mathbf{b}}(s, \mathbf{x}^s)\|^2 - (\dot{\alpha}(s)\mathbf{x}^0 + \dot{\beta}(s)\mathbf{x}^1 + \dot{\gamma}(s)\mathbf{z})^T \hat{\mathbf{b}}(s, \mathbf{x}^s) \right] ds, \\ \mathcal{L}_s &= \int_0^1 \mathbb{E} \left[\frac{1}{2} \|\hat{\mathbf{s}}(s, \mathbf{x}^s)\|^2 + \gamma^{-1} \mathbf{z}^T \hat{\mathbf{s}}(s, \mathbf{x}^s) \right] ds.\end{aligned}\tag{30}$$

³<https://archive.ics.uci.edu/ml/datasets/PEMS-SF>

⁴https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets

Both loss functions involve the integral over diffusion time $s \in [0, 1]$ in the form of

$$\mathcal{L} = \int_0^1 l(s)ds \approx \sum_i l(s_i), \quad s_i \sim \text{Uniform}[0, 1]. \quad (31)$$

However, the loss value $l(s)$ has a large variance, especially when s is near 0 or 1. Figure 6 shows an example of the distribution of $l(s)$ across multiple s . The large variance slows down the convergence of training. To overcome this issue, we apply importance sampling, a similar technique used by [Song et al., 2021, Sec. 5.1], to stabilize the training. Instead of drawing diffusion time from a uniform distribution, importance sampling considers,

$$\mathcal{L} = \int_0^1 l(s)ds \approx \sum_i \frac{l(s_i)}{\tilde{q}(s_i)}, \quad s_i \sim \tilde{q}(s). \quad (32)$$

Ideally, one wants to keep $l(s_i)/\tilde{q}(s_i)$ as constant as possible such that the variance of the estimation is minimum. The loss value $l(s)$ is very large when s is close to 0 or 1, and $l(s)$ is relatively flat in the middle, and the domain of s is $[0, 1]$, so we choose Beta distribution $\text{Beta}(s; 0.1, 0.1)$ as the proposal distribution \tilde{q} . As shown in Figure 6, the values of $l(s_i)/\tilde{q}(s_i)$ are plotted against their s , which becomes more concentrated in a small range.

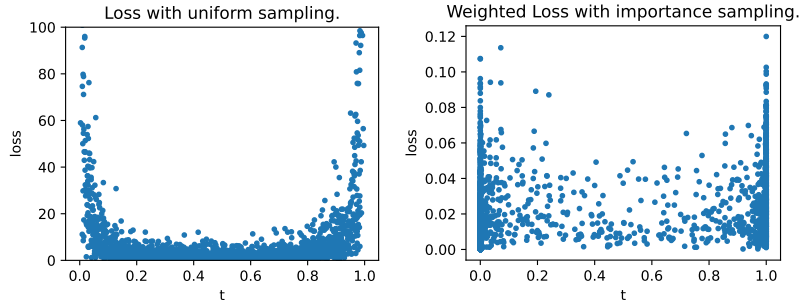


Figure 6: Comparison between uniform sampling and importance sampling. Each dot represents the loss of one sample with respect to the diffusion time.

C.4 ADDITIONAL FORECASTING RESULTS

	Exchange rate	Solar	Traffic	Wiki
DDPM	0.011 ± 0.004	0.377 ± 0.061	0.064 ± 0.014	0.093 ± 0.023
FM	0.011 ± 0.001	0.445 ± 0.031	0.041 ± 0.002	80.624 ± 89.804
SGM	0.01 ± 0.002	0.388 ± 0.026	0.08 ± 0.053	0.122 ± 0.026
SI	0.008 ± 0.002	0.399 ± 0.065	0.089 ± 0.006	0.091 ± 0.011

Table 4: ND-sum. A smaller number indicates better performance.

	Exchange rate	Solar	Traffic	Wiki
DDPM	0.013 ± 0.005	0.72 ± 0.08	0.094 ± 0.029	0.123 ± 0.026
FM	0.014 ± 0.002	0.849 ± 0.072	0.059 ± 0.007	165.128 ± 147.682
SGM	0.019 ± 0.004	0.76 ± 0.066	0.109 ± 0.064	0.164 ± 0.03
SI	0.01 ± 0.003	0.722 ± 0.132	0.127 ± 0.003	0.117 ± 0.011

Table 5: NRMSE-sum. A smaller number indicates better performance.

C.5 BASELINE MODEL USING UNCONDITIONAL SI

Additionally, we introduced a new experiment to verify the necessity of conditional SI over unconditional SI. The unconditional SI diffuses from pure noise and does not utilize the prior distribution from the previous time point. In this case, the context for the prediction is provided exclusively by the RNN encoder. The new results are shown in the following tables. When compared with the conditional SI framework, the unconditional model shows slightly inferior performance.

	Exchange rate	Solar	Traffic
SI	0.007 ± 0.001	0.359 ± 0.06	0.083 ± 0.005
Vanilla SI	0.010 ± 0.001	0.383 ± 0.010	0.082 ± 0.006

Table 6: CRPS-sum metric on multivariate probabilistic forecasting. A smaller number indicates better performance.

	Exchange rate	Solar	Traffic
SI	0.008 ± 0.002	0.399 ± 0.065	0.089 ± 0.006
Vanilla SI	0.010 ± 0.003	0.430 ± 0.113	0.093 ± 0.007

Table 7: ND-sum. A smaller number indicates better performance.

	Exchange rate	Solar	Traffic
SI	0.010 ± 0.003	0.722 ± 0.132	0.127 ± 0.003
Vanilla SI	0.012 ± 0.003	0.815 ± 0.135	0.132 ± 0.015

Table 8: NRMSE-sum. A smaller number indicates better performance.