Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows

Ruizhi Deng [1 2 *] **Bo Chang** [1] **Marcus A. Brubaker** [1 3] **Greg Mori** [1 2] **Andreas Lehrmann** [1]

Abstract

Normalizing flows transform a simple base distribution into a complex target distribution and have proved to be powerful models for data generation and density estimation. In this work, we propose a novel type of normalizing flow driven by a differential deformation of the Wiener process. As a result, we obtain a rich time series model whose observable process inherits many of the appealing properties of its base process, such as efficient computation of likelihoods and marginals. Furthermore, our continuous treatment provides a natural framework for irregular time series with an independent arrival process, including straightforward interpolation. We illustrate the desirable properties of the proposed model on popular stochastic processes and demonstrate its superior flexibility to variational RNN and latent ODE baselines in a series of experiments on synthetic and real-world data.

1. Introduction

Expressive models for sequential data form the statistical basis for downstream tasks in a wide range of domains, including computer vision, robotics, and finance. Meanwhile, recent deep generative modeling has enabled vastly increased flexibility while keeping generation and inference tractable (Kingma & Welling, 2014; Chung et al., 2015; Rezende & Mohamed, 2015; Kingma & Dhariwal, 2018; Chen et al., 2018; Li et al., 2020).

In this work, we approach the modeling of continuous and irregular time series with a *reversible generative model for stochastic processes*. Our approach builds upon ideas from normalizing flows; however, instead of a static base distribution, we transform a dynamic base process into an observable one. In particular, we introduce the *continuous-time flow process (CTFP)*, a novel type of generative model that decodes the base continuous Wiener process into a complex observable process using a dynamic instance of normalizing flows. The resulting observable process is thus continuous in time. In addition to the appealing properties of static normalizing flows (e.g., efficient sampling and exact likelihood), this also enables a series of inference tasks that are typically unattainable in time series models with complex dynamics, such as interpolation and extrapolation at arbitrary timestamps. Furthermore, to overcome the simple covariance structure of the Wiener process, we augment the reversible mapping with latent variables and optimize this latent CTFP variant using variational optimization.

Contributions. In summary, we propose the *continuous-time flow process (CTFP)*, a novel generative model for continuous stochastic processes. It has the following appealing properties: (1) it induces flexible and consistent joint distributions on arbitrary and irregular time grids, with easy-to-compute density and an efficient sampling procedure; (2) the stochastic process generated by CTFP is guaranteed to have continuous sample paths, making it a natural fit for data with continuously-changing dynamics; (3) CTFP can perform interpolation and extrapolation conditioned on given observations. We validate our model and its latent variant on various common stochastic processes and show superior performance than the state-of-the-art methods including the variational recurrent neural network (VRNN) (Chung et al., 2015) and latent ordinary differential equation (latent ODE) (Rubanova et al., 2019).

2. Background: Wiener Processes

A stochastic process can be defined as a collection of random variables that are indexed by time. An example of continuous stochastic processes is the *Wiener process*. The d-dimensional Wiener process $\boldsymbol{W}_\tau$ can be characterized by the following properties: (1) $\boldsymbol{W}_0 = 0$; (2) $\boldsymbol{W}_t - \boldsymbol{W}_s \sim \mathcal{N}(0, (t-s)\boldsymbol{I}_d)$ for $s \leq t$, and $\boldsymbol{W}_t - \boldsymbol{W}_s$ is independent of past values of $\boldsymbol{W}_{s'}$ for all $s' \leq s$. The joint density of $(\boldsymbol{W}_{\tau_1}, \ldots, \boldsymbol{W}_{\tau_n})$ can be written as the product of the conditional densities: $p_{(\boldsymbol{W}_{\tau_1},\ldots,\boldsymbol{W}_{\tau_n})}(\boldsymbol{w}_{\tau_1}, \ldots, \boldsymbol{w}_{\tau_n}) = \prod_{i=1}^{n} p_{\boldsymbol{W}_{\tau_i}|\boldsymbol{W}_{\tau_{i-1}}}(\boldsymbol{w}_{\tau_i}|\boldsymbol{w}_{\tau_{i-1}})$ for $0 \leq \tau_1 < \cdots < \tau_n \leq T$.

The conditional distribution of $p_{\boldsymbol{W}_t|\boldsymbol{W}_s}$, for $s < t$, is multi-

[*] Work done during an internship at Borealis AI. [1]Borealis AI [2]Simon Fraser University [3]York University. Correspondence to: Ruizhi Deng <ruizhid@sfu.ca>.

Second workshop on *Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models* (ICML 2020), Virtual Conference

variate Gaussian; its conditional density is

$$p_{\boldsymbol{W}_t|\boldsymbol{W}_s}(\boldsymbol{w}_t|\boldsymbol{w}_s) = \mathcal{N}(\boldsymbol{w}_t; \boldsymbol{w}_s, (t-s)\boldsymbol{I}_d), \tag{1}$$

where $\boldsymbol{I}_d$ is a d-dimensional identity matrix. This equation also provides a way to sample from $(\boldsymbol{W}_{\tau_1}, \ldots, \boldsymbol{W}_{\tau_n})$. Furthermore, given $\boldsymbol{W}_{t_1} = \boldsymbol{w}_{t_1}$ and $\boldsymbol{W}_{t_2} = \boldsymbol{w}_{t_2}$, the conditional distribution of $\boldsymbol{W}_t$ for $t_1 \leq t \leq t_2$ is also Gaussian:

$$p_{\boldsymbol{W}_t|\boldsymbol{W}_{t_1},\boldsymbol{W}_{t_2}}(\boldsymbol{w}_t|\boldsymbol{w}_{t_1}, \boldsymbol{w}_{t_2}) = \mathcal{N}\left(\boldsymbol{w}_t; \boldsymbol{w}_{t_1} + \frac{t-t_1}{t_2-t_1}(\boldsymbol{w}_{t_2} - \boldsymbol{w}_{t_1}), \frac{(t_2-t)(t-t_1)}{t_2-t_1}\boldsymbol{I}_d\right). \tag{2}$$

This is known as the Brownian bridge. An important property of the Wiener process is that the sample paths are continuous in time with probability one. This property allows our models to generate continuous sample paths and perform interpolation and extrapolation tasks.

3. Model

We define our proposed *continuous-time flow process (CTFP)* in Section 3.1. In Section 3.2, a generative variant of ANODE is presented as a component to implement CTFP. Since the proposed stochastic process is continuous in time, it enables interpolation and extrapolation to arbitrary time points, as described in Section 3.3. Finally, richer covariance structures are enabled by the latent CTFP model presented in Section 3.4.

3.1. Continuous-Time Flow Process

Let $\{(\boldsymbol{x}_{\tau_i}, \tau_i)\}_{i=1}^n$, denote a sequence of irregularly spaced time series data. We assume the time series to be an (incomplete) realization of a continuous stochastic process $\{\boldsymbol{X}_\tau\}_{\tau\in[0,T]}$. In other words, this stochastic process induces a joint distribution of $(\boldsymbol{X}_{\tau_1}, \ldots, \boldsymbol{X}_{\tau_n})$. Our goal is to model $\{\boldsymbol{X}_\tau\}_{\tau\in[0,T]}$ such that the log-likelihood on the observations

$$\mathcal{L} = \log p_{\boldsymbol{X}_{\tau_1},\ldots,\boldsymbol{X}_{\tau_n}}(\boldsymbol{x}_{\tau_1}, \ldots, \boldsymbol{x}_{\tau_n}) \tag{3}$$

is maximized. We define the continuous-time flow process (CTFP) $\{F_{\boldsymbol{\theta}}(\boldsymbol{W}_\tau; \tau)\}_{\tau\in[0,T]}$ such that

$$\boldsymbol{X}_\tau = F_{\boldsymbol{\theta}}(\boldsymbol{W}_\tau; \tau), \quad \forall \tau \in [0, T], \tag{4}$$

where $F_{\boldsymbol{\theta}}(\cdot;\tau) : \mathbb{R}^d \to \mathbb{R}^d$ is an invertible mapping parametrized by the learnable parameters $\boldsymbol{\theta}$ for every $\tau \in [0, T]$, and $\boldsymbol{W}_\tau$ is a d-dimensional Wiener process.

The log-likelihood in Equation 3 can be rewritten using the change of variables formula. Let $\boldsymbol{w}_{\tau_i} = F_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x}_{\tau_i}; \tau_i)$, then

$$\mathcal{L} = \sum_{i=1}^n \left[\log p_{\boldsymbol{W}_{\tau_i}|\boldsymbol{W}_{\tau_{i-1}}}(\boldsymbol{w}_{\tau_i}|\boldsymbol{w}_{\tau_{i-1}}) - \log\left|\det \frac{\partial \boldsymbol{x}_{\tau_i}}{\partial \boldsymbol{w}_{\tau_i}}\right|\right], \tag{5}$$

where $\tau_0 = 0$, $\boldsymbol{W}_0 = 0$, and $p_{\boldsymbol{W}_{\tau_i}|\boldsymbol{W}_{\tau_{i-1}}}$ is defined in Section 2. Figure 1a shows an example of the likelihood calculation. Sampling from a CTFP is straightforward. Given the timestamps τ_i, we first sample a realization of the Wiener process $\{\boldsymbol{w}_{\tau_i}\}_{i=1}^n$, then map them to $\boldsymbol{x}_{\tau_i} = F_{\boldsymbol{\theta}}(\boldsymbol{w}_{\tau_i})$. Figure 1b illustrates this procedure.

The normalizing flow models $F_{\boldsymbol{\theta}}(\cdot;\tau)$ transform simple base distribution induced by $\boldsymbol{W}_\tau$ on arbitrary time grid into more complex shapes in the observation space. It is worth noting that given a continuous realization of $\boldsymbol{W}_\tau$, as long as $F_{\boldsymbol{\theta}}(\cdot;\tau)$ is implemented as a continuous mapping, the resulting trajectory $\boldsymbol{x}_\tau$ is also continuous.

3.2. Generative ANODE

In principle, any normalizing flow model (Rezende & Mohamed, 2015; Dinh et al., 2014; Kingma et al., 2016; Dinh et al., 2017; Papamakarios et al., 2017; Kingma & Dhariwal, 2018; Behrmann et al., 2019; Chen et al., 2019; Kobyzev et al., 2019; Papamakarios et al., 2019) indexed by time τ could be used as $F_{\boldsymbol{\theta}}(\cdot;\tau)$ in Equation 4. We proceed with the continuous normalizing flow (Chen et al., 2018) because it has free-form Jacobian and efficient trace estimator (Grathwohl et al., 2019). In particular, we consider the following instantiation of augmented neural ordinary differential equation (ANODE) (Dupont et al., 2019) as a generative model: For any $\tau \in [0, T]$ and $\boldsymbol{w}_\tau \in \mathbb{R}^d$, we map $\boldsymbol{w}_\tau$ to $\boldsymbol{x}_\tau$ by solving the following initial value problem:

$$\frac{d}{dt}\begin{pmatrix}\boldsymbol{h}_\tau(t)\\ a_\tau(t)\end{pmatrix} = \begin{pmatrix}f_{\boldsymbol{\theta}}(\boldsymbol{h}_\tau(t), a_\tau(t), t)\\ g_{\boldsymbol{\theta}}(a_\tau(t), t)\end{pmatrix}, \quad \begin{pmatrix}\boldsymbol{h}_\tau(t_0)\\ a_\tau(t_0)\end{pmatrix} = \begin{pmatrix}\boldsymbol{w}_\tau\\ \tau\end{pmatrix}, \tag{6}$$

where $\boldsymbol{h}_\tau(t) \in \mathbb{R}^d, t \in [t_0, t_1]$, $f_{\boldsymbol{\theta}} : \mathbb{R}^d \times \mathbb{R} \times [t_0, t_1] \to \mathbb{R}^d$, and $g_{\boldsymbol{\theta}} : \mathbb{R} \times [t_0, t_1] \to \mathbb{R}$. Then $F_{\boldsymbol{\theta}}$ in Equation 4 is defined as the solution of $\boldsymbol{h}_\tau(t)$ at $t = t_1$:

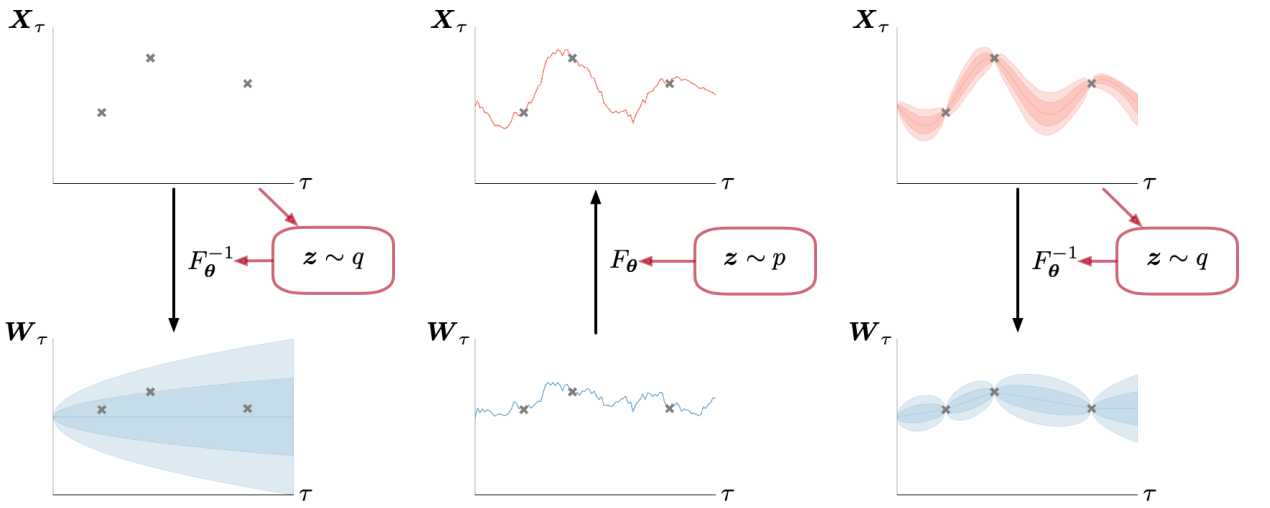
$$F_{\boldsymbol{\theta}}(\boldsymbol{w}_\tau; \tau) := \boldsymbol{h}_\tau(t_1) = \boldsymbol{h}_\tau(t_0) + \int_{t_0}^{t_1} f_{\boldsymbol{\theta}}\left(\boldsymbol{h}_\tau(t), a_\tau(t), t\right) dt. \tag{7}$$

Note that the index t represents the independent variable in the initial value problem and should not be confused with τ, the timestamp of the observation.

Using the Instantaneous Change of Variable theorem(Chen et al., 2018), the log-likelihood $\mathcal{L}$ can be calculated as follows:

$$\mathcal{L} = \sum_{i=1}^n \left[\log p_{\boldsymbol{W}_{\tau_i}|\boldsymbol{W}_{\tau_{i-1}}}\left(\boldsymbol{h}_{\tau_i}(t_0)|\boldsymbol{h}_{\tau_{i-1}}(t_0)\right) - \int_{t_0}^{t_1} \operatorname{tr}\left(\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{h}_{\tau_i}(t), a_{\tau_i}(t), t)}{\partial \boldsymbol{h}_{\tau_i}(t)}\right) dt\right], \tag{8}$$

where $\boldsymbol{h}_{\tau_i}(t_0)$ is obtained by solving the ODE in Equation 6 backwards from $t = t_1$ to $t = t_0$, and the trace of the



(a) Likelihood calculation. (b) Sampling. (c) Interpolation and extrapolation.

Figure 1. **(Latent) Continuous-Time Flow Processes (CTFPs).** (**a**) Likelihood calculation. Given an irregular time series $\{\boldsymbol{x}_{\tau_i}\}$, the inverse flow $F_{\boldsymbol{\theta}}^{-1}$ maps the observations to a set of Wiener points $\{\boldsymbol{w}_{\tau_i}\}$ for which we can compute the likelihood according to Equation 5. (**b**) Sampling. Given a set of timestamps $\{\tau_i\}$, we sample a Wiener process and use the forward flow $F_{\boldsymbol{\theta}}$ to obtain an observed process. (**c**) Interpolation and extrapolation. In order to compute the density at an unobserved point $\boldsymbol{x}_\tau$, we compute the left-sided (extrapolation; Equation 1) or two-sided (interpolation; Equation 2) conditional density of its Wiener point $\boldsymbol{w}_\tau$ and adjust for the flow (Equation 9). **Notes:** The effect of the latent variables $\boldsymbol{Z}$ in our latent CTFP model is indicated by red boxes. The shaded areas represent 70% and 95% confidence intervals.

Jacobian can be estimated by Hutchinson's trace estimator (Hutchinson, 1990; Grathwohl et al., 2019).

3.3. Interpolation and Extrapolation with CTFP

Time-indexed normalizing flows and the Brownian bridge allow us to define conditional distributions on arbitrary timestamps. They also permit the CTFP model to perform interpolation and extrapolation given partial observations, which are of great importance in time series modeling.

Interpolation means that we can model the conditional distribution $p_{\boldsymbol{X}_\tau|\boldsymbol{X}_{\tau_i},\boldsymbol{X}_{\tau_{i+1}}}(\boldsymbol{x}_\tau|\boldsymbol{x}_{\tau_i},\boldsymbol{x}_{\tau_{i+1}})$ for all $\tau \in [\tau_i, \tau_{i+1}]$ and $i = 1, \ldots, n-1$. This can be done by mapping the values $\boldsymbol{x}_\tau$, $\boldsymbol{x}_{\tau_i}$ and $\boldsymbol{x}_{\tau_{i+1}}$ to $\boldsymbol{w}_\tau$, $\boldsymbol{w}_{\tau_i}$ and $\boldsymbol{w}_{\tau_{i+1}}$, respectively. After that, Equation 2 can be applied to obtain the conditional density of $p_{\boldsymbol{W}_\tau|\boldsymbol{W}_{\tau_1},\boldsymbol{W}_{\tau_2}}(\boldsymbol{w}_\tau|\boldsymbol{w}_{\tau_1},\boldsymbol{w}_{\tau_2})$. Finally, we have

$$\begin{aligned}&\log p_{\boldsymbol{X}_\tau|\boldsymbol{X}_{\tau_i},\boldsymbol{X}_{\tau_{i+1}}}(\boldsymbol{x}_\tau|\boldsymbol{x}_{\tau_i},\boldsymbol{x}_{\tau_{i+1}})\\ &= \log p_{\boldsymbol{W}_\tau|\boldsymbol{W}_{\tau_i},\boldsymbol{W}_{\tau_{i+1}}}(\boldsymbol{w}_\tau|\boldsymbol{w}_{\tau_i},\boldsymbol{w}_{\tau_{i+1}}) - \log\left|\det\frac{\partial \boldsymbol{x}_\tau}{\partial \boldsymbol{w}_\tau}\right|.\end{aligned} \tag{9}$$

Extrapolation can be done in a similar fashion using Equation 1. This allows the model to predict continuous trajectories into the future, given past observations. Figure 1c shows a visualization of interpolation and extrapolation using CTFP.

3.4. Latent Continuous-Time Flow Process

The CTFP model inherits the independent increment property from the Wiener process, which is a strong assumption and limits its ability to model stochastic processes with complex temporal dependence. In order to enhance the expressive power of the CTFP model, we augment it with a latent variable $\boldsymbol{Z} \in \mathbb{R}^m$, whose prior distribution is an isotropic Gaussian $p_{\boldsymbol{Z}}(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; 0, \boldsymbol{I}_m)$. As a result, the data distribution can be approximated by a diverse collection of CTFP models conditioned on sampled latent variables $\boldsymbol{z}$.

The generative model in Equation 4 is augmented to $\boldsymbol{X}_\tau = F_{\boldsymbol{\theta}}(\boldsymbol{W}_\tau; \boldsymbol{Z}, \tau), \forall \tau \in [0, T]$, which induces the conditional distribution $\boldsymbol{X}_{\tau_1}, \ldots, \boldsymbol{X}_{\tau_n}|\boldsymbol{Z}$. Similar to the initial value problem in Equation 6, we define $F_{\boldsymbol{\theta}}(\boldsymbol{w}_\tau; \boldsymbol{z}, \tau) = \boldsymbol{h}_\tau(t_1)$, where

$$\frac{d}{dt}\begin{pmatrix}\boldsymbol{h}_\tau(t)\\ \boldsymbol{a}_\tau(t)\end{pmatrix} = \begin{pmatrix}f_{\boldsymbol{\theta}}(\boldsymbol{h}_\tau(t), \boldsymbol{a}_\tau(t), t)\\ g_{\boldsymbol{\theta}}(\boldsymbol{a}_\tau(t), t)\end{pmatrix}, \begin{pmatrix}\boldsymbol{h}_\tau(t_0)\\ \boldsymbol{a}_\tau(t_0)\end{pmatrix} = \begin{pmatrix}\boldsymbol{w}_\tau\\ (\boldsymbol{z}, \tau)^\top\end{pmatrix}. \tag{10}$$

For simplicity of notation, the subscripts of density functions are omitted from now on. For the augmented generative model, the log-likelihood becomes $\mathcal{L} = \log \int_{\mathbb{R}^m} p(\boldsymbol{x}_{\tau_1}, \ldots, \boldsymbol{x}_{\tau_n}|\boldsymbol{z})p(\boldsymbol{z})\ d\boldsymbol{z}$, which is intractable to evaluate. Following the variational autoencoder approach (Kingma & Welling, 2014), we introduce an approximate posterior distribution of $\boldsymbol{Z}|\boldsymbol{X}_{\tau_1}, \ldots, \boldsymbol{X}_{\tau_n}$, denoted by $q(\boldsymbol{z}|\boldsymbol{x}_{\tau_1}, \ldots, \boldsymbol{x}_{\tau_n})$. The implementation of the approximate posterior distribution is an ODE-RNN en-

Table 1. **Quantitative Evaluation (Synthetic Data)**. We show test negative log-likelihood on three synthetic stochastic processes across different models. Below each process, we indicate the intensity of the Poisson point process from which the timestamps for the test sequences were sampled. "Ground Truth" refers to the closed-form negative log-likelihood of the true underlying data generation process. [GBM: geometric Brownian motion; OU: Ornstein–Uhlenbeck process; M-OU: mixture of OUs.]

Model	GBM		OU		M-OU
	$\lambda_{\text{test}} = 2$	$\lambda_{\text{test}} = 20$	$\lambda_{\text{test}} = 2$	$\lambda_{\text{test}} = 20$	$\lambda_{\text{test}} = (2, 20)$
Latent ODE (Rubanova et al., 2019)	3.826	5.935	3.066	3.027	2.690
VRNN (Chung et al., 2015)	3.762	3.492	**2.729**	1.939	1.415
CTFP (**ours**)	3.107	1.931	2.903	1.942	1.432
Latent CTFP (**ours**)	**3.106**	**1.929**	2.902	**1.936**	**1.391**
Ground Truth	3.106	1.928	2.722	1.888	1.379

coder (Rubanova et al., 2019). With the approximate posterior distribution, we can derive an importance-weighted autoencoder (IWAE) (Burda et al., 2016) lower bound of the log-likelihood on the right-hand side of the inequality:

$$\mathcal{L}_{\text{IWAE}} = \mathbb{E}_{\boldsymbol{z}_k \sim q}\left[\log\left(\frac{1}{K}\sum_{k=1}^{K}\frac{p(\boldsymbol{x}_{\tau_1},\ldots,\boldsymbol{x}_{\tau_n}|\boldsymbol{z}_k)p(\boldsymbol{z}_k)}{q(\boldsymbol{z}_k|\boldsymbol{x}_{\tau_1},\ldots,\boldsymbol{x}_{\tau_n})}\right)\right], \tag{11}$$

where K is the number of samples from the approximate posterior distribution.

4. Experiments

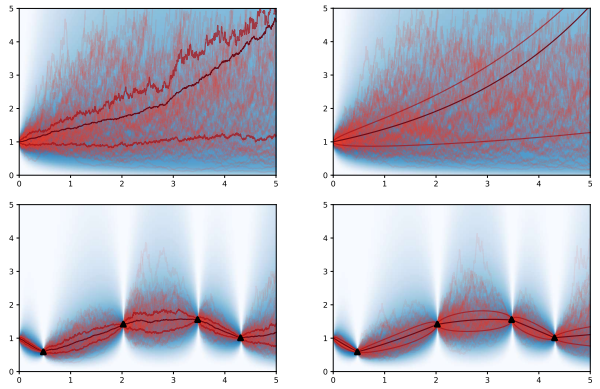
We apply our models on synthetic data generated from common continuous-time stochastic processes. The proposed CTFP and latent CTFP models are compared against two baseline models: latent ODE (Rubanova et al., 2019) and variational RNNs (VRNNs) (Chung et al., 2015). For VRNNs, we append the time gap between two observations as an additional input to the neural network. Both latent CTFP and latent ODE models use ODE-RNN (Rubanova et al., 2019) as the inference network; GRU (Cho et al., 2014) is used as the RNN cell in latent CTFP, latent ODE, and VRNN models.

We simulate three irregularly-sampled time series datasets; all of them are univariate. **Geometric Brownian motion** (GBM) is a continuous-time stochastic process widely used in mathematical finance. It satisfies the following stochastic differential equation: $dX_\tau = \mu X_\tau d\tau + \sigma X_\tau dW_\tau$. The timestamps of the observations are in the range between 0 and $T = 30$ and are sampled from a homogeneous Poisson point process with an intensity of $\lambda_{\text{train}} = 2$. To further evaluate the model's capacity to capture the dynamics of GBM, we test the model with observation time-steps sampled from Poisson point processes with intensities of $\lambda_{\text{test}} = 2$ and $\lambda_{\text{test}} = 20$. **Ornstein–Uhlenbeck process** (OU Process) is another type of widely used continuous-time stochastic process. The OU process satisfies the following stochastic differential equation: $dX_\tau = \theta(\mu - X_\tau)d\tau + \sigma dW_\tau$. We use the same set of observation intensity values as GBM to sample observation timestampes in the training and test sets. **Mixture of OUs.** To demonstrate the latent CTFP's capability to model sequences sampled from different continuous-time stochastic processes, we train the models on a dataset generated by mixing the sequences sampled from two different OU processes with different values of θ, μ, σ and different observation intensities. We defer more details of model implementation, the parameters of the synthetic dataset, and experiment settings to the supplementary material.

Results. The results are presented in Table 1. We report the exact negative log-likelihood (NLL) per observation for CTFP. For latent ODE, latent CTFP, and VRNN, we report the (upper bound of) NLL estimated by the IWAE bound (Burda et al., 2016) in Equation 11, using $K = 25$ samples of latent variables.

The results on the test set sampled from GBM indicate that the CTFP model can recover the true data generation process as the NLL estimated by CTFP is close to the ground truth. On the M-OU dataset, latent CTFP shows better performance than other models and outperforms CTFP by 0.04 nats, indicating its ability to leverage the latent variables. Although trained on samples with an observation intensity of $\lambda_{\text{train}} = 2$, CTFP can better adapt to samples with a bigger observation intensity (and thus denser time grid) of $\lambda_{\text{test}} = 20$. We hypothesize that the superior performance of CTFP models when $\lambda_{\text{test}} = 20$ is due to their capability to model continuous stochastic processes, whereas the baseline models do not have the property of continuity.

Figure 2 provides qualitative samples from CTFP model trained on the GBM data, on the generation task (upper panels) and the interpolation task (lower panels). In additional to samples, we also show the the marginal density (blue) of CTFP for each time stamp, and sample-based estimates of the inter-quantile range (dark red) and mean (brown) of the marginal density for both tasks. We compare the results of CTFP with the samples and ground truth marginal density, inter-quantile range, and mean of GBM. The comparison shows the results of CTFP is consistent with the ground truth in terms of both uncertainty estimate and point estimate.



(a) CTFP (b) GBM

Figure 2. **Comparison between CTFP and GBM.** We consider the generation and interpolation tasks for (a) CTFP, and (b) GBM. In each subfigure, the upper panel shows unconditional generation samples and the lower panel shows samples for interpolation. The observed points for interpolation are marked by black triangles. In addition to the sample trajectories (red) and the marginal density (blue), we also show the sample-based estimates (closed-form for ground truth) of the inter-quantile range (dark red) and mean (brown) of the marginal density.

References

Behrmann, J., Grathwohl, W., Chen, R. T., Duvenaud, D., and Jacobsen, J.-H. Invertible residual networks. In *International Conference on Machine Learning*, pp. 573–582, 2019.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in neural information processing systems*, pp. 6571–6583, 2018.

Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pp. 9913–9923, 2019.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural ODEs. In *Advances in Neural Information Processing Systems*, pp. 3134–3144, 2019.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., and Duvenaud, D. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

He, J., Spokoyny, D., Neubig, G., and Berg-Kirkpatrick, T. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

Kobyzev, I., Prince, S., and Brubaker, M. A. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

Li, X., Wong, T.-K. L., Chen, R. T., and Duvenaud, D. Scalable gradients for stochastic differential equations. *arXiv preprint arXiv:2001.01328*, 2020.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pp. 2338–2347, 2017.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Rubanova, Y., Chen, T. Q., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pp. 5321–5331, 2019.

A. Experiment Setup and Model Architecture Details

We describe all the important details on synthetic dataset generation, model architecture, as well as training and evaluation settings in this section.

A.1. Synthetic Dataset Details

For Geometric Brownian Motion (GBM), we sample 10000 trajectories from a GBM with the parameters of $\mu = 0.2$ and a variance of $\sigma = 0.5$ in the interval of [0, 30]. The timestamps of the observations are sampled from a homogeneous Poisson point process with an intensity of $\lambda_{train} = 2$. We evaluate the model on the observations timestamps sampled from two homogeneous Poisson processes separately with intensity values of $\lambda_{test} = 2$ and $\lambda_{test} = 20$.

For Ornstein–Uhlenbeck (OU) process, the parameters of the process we sample trajectories from are $\theta = 2, \mu = 1$ and $\sigma = 10$. We also sample 10000 trajectories and use the same set of observation intensity values, λ_{train} and λ_{test}, to sample observation timestamps from homogeneous Poisson processes for training and test.

For Mixture of OU processes (MOU), we sample 5000 sequences from each of two different OU processes and mix them. One OU process has the parameters of $\theta = 2, \mu = 1$, and $\sigma = 10$ and the observation timestamps are sampled from a homogeneous Poisson process with $\lambda = 2$. The other OU process has the parameters of $\theta = 1.0, \mu = 2.0$ and $\sigma = 5.0$ with observation timestamps sampled with $\lambda = 20$.

For the 10000 trajectories of each dataset, we use 7000 trajectories for training and 1000 trajectories for validation. We test the model on 2000 trajectories for each value of λ_{test}.

A.2. Model Architecture Details

To ensure a fair comparison, we use the same values for important hyper-parameters like the latent variable and hidden state dimensions across all models. Likewise, we keep the underlying architectures as similar as possible and use the same experimental protocol across all models.

For CTFP and Latent CTFP, we use a one-block augmented neural ODE module that maps the base process to the observation space. In the augmented neural ODE model, we use an MLP model consisting of 4 hidden layers of size 32–64–64–32 for both f and g in Equation 6 and 10.

For latent CTFP and latent ODE models, we use the same ODE-RNN model as the recognition network. The ODE-RNN model consists of a one-layer GRU cell with a hidden dimension of 20 (the rec-dims parameter in its original implementation) and a one-block neural ODE module that

has a single hidden layer of size 100, and it outputs a 10-dimensional latent variable. The same architecture is used by both latent ODE and latent CTFP models.

For the generation network of the latent ODE model, we use an ODE function with one hidden layer of size 100. The decoder network has 4 hidden layers of size 32–64–64–32; it maps a latent trajectory to parameters of Gaussian distributions at different time steps.

The VRNN model is implemented using a GRU network. The hidden state of the GRU cell is 20-dimensional. The dimension of the latent variable is 10. We use an MLP of 4 hidden layers of size 32–64–64–32 for the decoder network, an MLP with one hidden layer that has the same dimension as the hidden state for the prior proposal network and an MLP with two hidden layers for the posterior proposal network.

For data sampled from Geometric Brownian Motion, we apply an exponential activation function to the output of all models. Therefore the distribution precited by latent ODE and VRNN at each timestamp is a log-normal distribution.

A.3. Training and Evaluation Settings

We train all models using the IWAE bound with 5 samples and a flat learning rate of 5×10^{-4} for all models. We also consider models trained with or without the aggressive training scheme proposed by He et al. (2019) for latent ODE and latent CTFP. We choose the best-performing model among the ones trained with or without the aggressive scheme based IWAE bound, estimated with 25 samples on the validation set for evaluation. The training batch size is 100 for CTFP models and 25 for all the other models.