Scalable Neural Latent Stochastic Differential Equations

Kareem Elsawah, Michael Ala, Serena Hacker, and Cordell Blanchard

Abstract—In this paper, we address the complexities of analyzing time-series medical data, characterized by irregular sampling intervals and a continuous-time nature. Traditional methods often fall short, requiring time binning, which can oversimplify the data. We propose a novel approach that blends the benefits of neural stochastic differential equations with earlier time-binning methods. This hybrid model significantly improves computational efficiency and scalability, making it more suitable for high-dimensional medical data analysis. Our approach offers a practical solution towards more accurate and comprehensive patient diagnosis and prognosis in healthcare.

Code— https://github.com/CordellBlanchard/ScalableLatentSDE

1 Introduction

The realm of healthcare presents a unique set of challenges and opportunities for machine learning, particularly in the context of time-series data. This type of data, characterized by sequences of observations over time, is prevalent in healthcare settings, where patient data often unfolds as a series of measurements and tests. The primary objective in analyzing such data can vary, ranging from forecasting future trends, diagnosing based on past data, understanding the underlying state and evolution of a patient's health, or modeling the entire generative process of health data.

Latent variable models, a significant class in this domain, aim to learn a generative process of the data. These models are particularly adept at answering diverse queries, whether they pertain to the dataset as a whole or to specific instances, by utilizing conditional queries. In healthcare, where data often follows a time-series pattern, these models have seen extensive development and application. However, a notable challenge arises from the inherently irregular nature of health data. Unlike regularly spaced measurements, healthcare data often comprises tests and readings taken sporadically, based on patient-specific needs, making the timing of these readings informative in itself.

Traditional models struggled with this irregularity, typically resorting to time binning until the advent of neural differential equations. These models represented a breakthrough by learning continuous-time functions through neural networks. Yet, they had a limitation in representing uncertainty – they assumed determinism following the initial patient state, an assumption often unrealistic in the dynamic and unpredictable context of healthcare.

To address this, the development of neural stochastic differential equations marked a significant advancement. These models excel in handling irregularly sampled data and effectively representing uncertainty, a critical aspect in medical prognostics and diagnostics. Nevertheless, they face challenges in scalability, particularly when dealing with high-dimensional data, due to the computational demands of accurate forward solvers.

In this study, we propose an innovative approach to enhance these machine learning models, specifically addressing the challenges associated with neural stochastic differential equations in healthcare data analysis. Our contribution lies in developing a hybrid method that facilitates the learning of neural stochastic differential equations without the necessity for computationally expensive and accurate forward solvers. This is achieved through the implementation of a novel network architecture that intelligently integrates checkpoints throughout the forward sampling process. These checkpoints enable the use of less precise, yet more computationally efficient solvers, without sacrificing the overall performance of the model.

2 Related Work

2.1 Deep Markov Models (DMM)

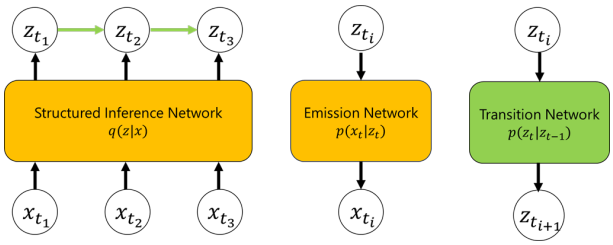

Fig. 1. Networks Used in the DMM Model

Our method originates from the model described by Krishnan et al. in "Structured Inference Networks for Nonlinear State Space Models" [1]. The aim of this model was to introduce a unified algorithm to efficiently learn a broad class of non-linear state space models, where the emission and transition distributions are modeled by deep neural networks [1]. The model has shown benefits in the healthcare domain, demonstrating promising results in predicting patient states using Electronic Health Record data for diabetic patients.

2.1.1 *Structured Inference Network*

The structured inference network describes the transition from the observation x to the latent sample z as shown in

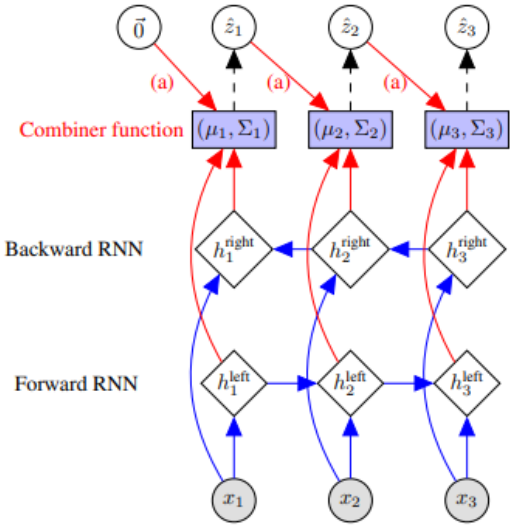

Fig. 2. Structured Inference Network (**ST-LR**) [1]

Figure 1. It is important to note that both x and z are random variables with a mean μ and variance σ^2. There are various inference networks mentioned in [1] but the one that we chose to implement for a baseline was the best performing one named ST-LR which means that the network used information from the past (denoted **L**, for left) and the future (denoted **R**, for right). Specifically this inference network is a bidirectional recurrent neural network (BRNN), as shown in Figure 2.

The BRNN takes as input the sequence $x_1, ..., x_T$ of observations and forms, initially, two sequences of intermediate states capturing the forward and backward temporal dynamics of the observations: h_t^{left} and h_t^{right} respectively. At each time step $t = 1, ..., T$ the intermediate states are combined via the "combiner function" (the arrow (a) in Figure 2), along with a hidden state z_{t-1} sampled from the previously predicted Gaussian, to produce the Gaussian at time step t (that is, it outputs μ_t, Σ_t). Formally, the output parameters μ_t, Σ_t describe the transition distribution $q(z_t \mid z_{t-1}, x)$ [1].

Below, the equations marked (1) explicitly show how the parameters μ_t, Σ_t are calculated. Note that we assume $\Sigma_t = \sigma_t I$ is diagonal, resulting in an isotropic Gaussian. Equation (2) shows the combiner function.

$$
\begin{aligned}
\mu_r &= W_{\mu_r}^{right} h_t^{right} + b_{\mu_r}^{right}; \\
\sigma_r^2 &= \text{softplus}\left(W_{\sigma_r^2}^{right} h_t^{right} + b_{\sigma_r^2}^{right}\right) \\
\mu_l &= W_{\mu_l}^{left} h_t^{left} + b_{\mu_l}^{left}; \\
\sigma_l^2 &= \text{softplus}\left(W_{\sigma_l^2}^{left} h_t^{left} + b_{\sigma_l^2}^{left}\right) \\
\mu_t &= \frac{\mu_r \sigma_l^2 + \mu_l \sigma_r^2}{\sigma_r^2 + \sigma_l^2};\ \sigma_t^2 = \frac{\sigma_r^2 \sigma_l^2}{\sigma_r^2 + \sigma_l^2} \qquad (1)
\end{aligned}
$$

$$
h_{combined} = \frac{1}{3}(tanh(W z_{t-1} + v) + h_t^{left} + h_t^{right} \qquad (2)
$$

2.1.2 Emission Network

The emission network is used to describe the reconstruction from the latent sample z to the observation x. This network is a two-layer multi-layer perceptron (MLP). In our re-implementation of this we used 2 feed-forward networks with ReLU as the non-linearities. The following equations were used to calculate the mean and variance from the output of the MLP.

$$
\mu_t = \frac{1}{2} MLP(z_t);\ \sigma_t^2 = 20 MLP(z_t)
$$

2.1.3 Transition Network

The transition network is used to describe the temporal dynamics of the latent samples z ($z_t \rightarrow z_{t+1}$). This network uses a network inspired by the Gated Recurrent Unit (GRU). What makes this different than a typical GRU is that it is not conditional on any of the input observations. This gives the model flexibility to choose a linear transition for some dimensions and non-linear transition for others [1]. Here is the description of the model, where **I** is the identity function and $\odot$ for element-wise multiplication:

$$
\begin{aligned}
&g_t = MLP(z_{t-1}, ReLU, sigmoid) \text{ (Gating Unit)} \\
&h_t = MLP(z_{t-1}, ReLU, I) \text{ (Proposed Mean)} \\
&\text{(Transition Mean and Variance)} \\
&\mu_t(z_{t-1}) = (1 - g_t) \odot (W_{\mu_p} z_{t-1} + b_{\mu_p}) + g_t \odot h_t \\
&\sigma_t^2(z_{t-1}) = \text{softplus}(W_{\sigma_p^2} \text{ReLU}(h_t) + b_{\sigma_p^2})
\end{aligned}
$$

Note that the mean and covariance functions both share the use of h_t. In the papers experiments, they initialize W_{μ_p} to be the identity function and b_{μ_p} to 0 [1]. We did similar in our implementation of their paper.

It should be noted that this specific network is the one that we will be changing in our method described in the Formal Description section.

2.1.4 Evidence Lower Bound (ELBO) Loss

Equation 3 shows the loss that is used to train all 3 of the mentioned above networks. The first term represents the loss for the reconstruction of x_t from z_t it is a maximum log likelihood function of the emission network $p_\theta(x_t|z_t)$. The second and third term are to measure the similarity between the transition network $p_\theta(z_t|z_{t-1}$ and structured inference network $q_\phi(z_t|z_{t-1}, \vec{x})$. We make changes to this function in our method mentioned in the formal description.

$$
\begin{aligned}
\mathcal{L}(\vec{x}; \theta, \phi) = &\sum_{t=1}^{T} \mathbb{E}_{q_\phi(z_t|\vec{x})} \left[\log p_\theta(x_t|z_t)\right] \\
&- \text{KL}(q_\phi(z_1|\vec{x}) \parallel p_\theta(z_1)) \\
&- \sum_{t=2}^{T} \mathbb{E}_{q_\phi(z_{t-1}|\vec{x})} \left[\text{KL}(q_\phi(z_t|z_{t-1}, \vec{x}) \parallel p_\theta(z_t|z_{t-1}))\right]
\end{aligned} \qquad (3)
$$

2.2 Latent Ordinary Differential Equations (ODE)

Rubanova et al. [2] propose a method for probabilistically modeling time-series data sampled at irregular time intervals using ODEs to model latent variable transition dynamics. Given an observed time series $\{x_i, t_i\}_{i=1}^N$, an encoder first converts the entire time series into parameters μ_{z_0}, σ_{z_0}, which are used to sample $z_0 \sim q_\phi(z_0 \mid \{x_i, t_i\}_{i=1}^N)$. The

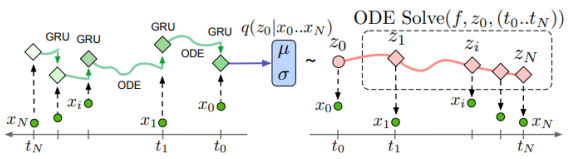

Fig. 3. Latent ODE Model [2]

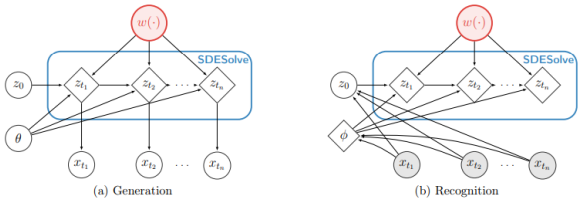

Fig. 4. Transition and Inference Networks in SDE Model [3]

initial state z_0 is used as input to an initial value problem, and samples $z_1, ..., z_N$ are drawn from a learned ODE:

$$z_0 \sim q_\phi(z_0 \mid \vec{x}) \approx p(z_0 \mid \vec{x})$$
$$z_0, z_1, ..., z_N = \text{ODESolve}(f_\theta, z_0, (t_0, t_1, ..., t_N)$$
$$\text{each } x_i \overset{\text{indep.}}{\sim} p(x_i|z_i) \;\; i = 0, 1, ...N$$

The encoder converting $\{x_i, t_i\}_{i=1}^N$ into the posterior distribution estimate ($q_\phi(z_0 \mid \vec{x})$) is an ODE-RNN, an RNN-like architecture where the transition from h_{i-1} to h_i is also modeled by a separate ODE:

$$h_i' := \text{ODESolve}(f_\phi, h_{i-1}, (t_{i-1}, t_i))$$
$$h_i := \text{RNNCell}(h_i', x_i)$$
$$o_i := \text{NN}(h_i)$$

By modeling the transition of the hidden states h_i as a continuous variable, the model is able to take into account the time interval (t_{i-1}, t_i) more effectively than a standard RNN. To go from $\{x_i, t_i\}_{i=1}^N$ to μ_{z_0}, σ_{z_0}, the final observation o_N is passed through a final MLP to obtain the distribution parameters.

The full model is demonstrated in figure 5 - the ODE-RNN encoder being the first half of the left, and the latent ODE decoder being the second half on the right.

One disadvantage to this approach is that the only probabilistic element of the latent variable is the initial sample z_0 - all other latent samples $z_1, ..., z_N$ are determined from z_0. In other words, the model doesn't intrinsically encode any uncertainty in the transition dynamics. Moreover, the observations x_i are only related to the corresponding latent states z_i via the initial sample z_0, creating an bottleneck in information flow.

It should also be noted that they tried different inference networks (encoders) using models other than the ODE-RNN. In our implementation of this paper as a baseline we only used a RNN similar to the DMM as the inference network.

2.3 Scalable Gradients for Stochastic Differential Equations (SDE)

The paper presented by Xuechen et al. [3] follows a similar approach to our method in the sense that they opted to use SDEs within a structure resembling the DMM. What distinguishes this approach is the utilization of two SDE solutions to replace both the transition network and the inference network. It should also be mentioned that they maintained a similar approach for the emission network as the DMM.

As shown in Figure 4, they have a generation model (a) and a recognition model (b), which are similar to the transition network and inference network mentioned earlier. In the generation approach, they replace their transition network with a method that treats all given latent samples as a solution to an SDE and uses this solution to predict further latent samples into the future. In the recognition approach, they replace the inference network ($x_t \rightarrow z_t$) with a solution to an SDE as well. However, they train all observations x_t to emit to the first latent sample z_0 and then predict further latent samples using the SDE solution. Our method differs in that we retain the DMM inference network and use a generation model similar to this one in our transition network.

3 Proposed Method

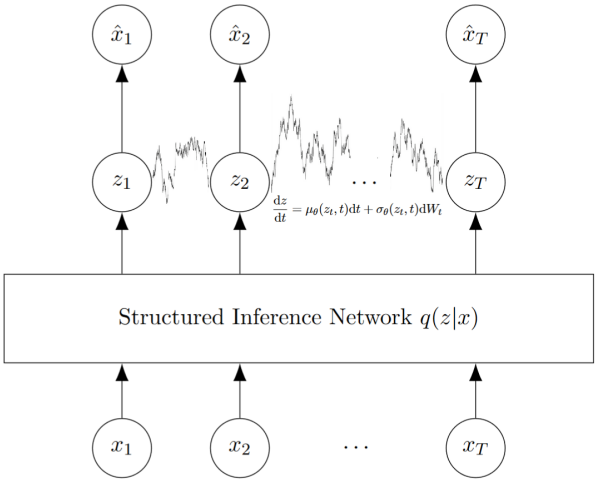

Fig. 5. Our proposed architecture. Utilizes the same structured inference network and emission network as proposed in [1]. Replaces their transition network with a neural stochastic differential equations represented using the drift μ_θ and diffusion σ_θ terms

Our method builds primarily off the work of Krishnan et al. [1] and Xuechen et al. [3]. We assume we are given a time series $\vec{x} := \{x_i, t_i\}_{i=1}^N$ of multivariate observations ($x_i \in \mathbb{R}^n$). As in figure 2, we first pass the observations, along with their time intervals $\Delta t_i := t_i - t_{i-1}$ ($\Delta t_1 := t_1$) through a bidirectional LSTM to obtain two sequences of hidden states h_i^{left} and h_i^{right}. At $i = 1$ the combiner function takes h_1^{left} and h_1^{right} and outputs μ_1, Σ_1, the mean and covariance of a Gaussian from which we sample z_i. From here, we progress iteratively by combining h_i^{left}, h_i^{right} and z_{i-1} to obtain μ_i, Σ_i, and sample z_i, repeating until we reach

z_N. The parameters μ_i, Σ_i model the *inference distribution* estimates, denoted $q(z_t \mid z_{t-1}, \vec{x})$.

Next, we wish to model the temporal dynamics of the latent state samples z_i with a stochastic differential equation

$$\mathrm{d}Z_t = \mu_\theta(Z_t, t)\mathrm{d}t + \sigma_\theta(Z_t, t)\mathrm{d}W_t \tag{4}$$

Here μ_θ and σ_θ are neural networks representing the *drift* and *diffusion* terms respectively. In our experiments these are represented as standard multi-layer perceptrons. The goal is to learn the parameters θ such that this SDE accurately models the temporal dynamics of the latent states. In other words, we wish to use the SDE to model $p(z_t|z_{t-1})$.

In order to do this, at each timestep t_i we start with the initial value $Z(t_i) := z_i$ and use the Euler-Maruyama method to estimate samples of the SDE at timestep t_{i+1}, then compare these estimates $\hat{z}_{i+1}$ against the inferred z_{i+1}. Put simply, we're comparing the SDE against the structured inference network and training both in tandem. For our specific implementation of the Euler-Maruyama method, we partition the intervals $[t_i, t_{i+1}]$ into K equal subintervals of length $\Delta t_{i+1}/K$, denoted $t_i = \tau_0 < \tau_1 < \cdots < \tau_K = t_{i+1}$, then recursively define approximations

$$\begin{aligned} y_0 &:= z_{\tau_0} \\ y_{k+1} &:= y_k + \frac{\Delta t_{i+1}}{K}\mu_\theta(y_k, \tau_k) + \sigma_\theta(y_k, \tau_k)w_l \end{aligned}$$

where $w_k \sim N(0, \Delta t_{i+1}/K)$ is a series of samples drawn iid from a zero-mean Gaussian, modeling the noise introduced by the Weiner process W_t. From this process we obtain $\widehat{z}_{i+1} := y_K$, an estimate of the next latent z_{i+1}.

Finally, we pass the latent variable samples z_t through an emission network to obtain parameters for the distribution $p(x_t \mid z_t)$, which we again assume is an isotropic Gaussian.

There are a couple of advantages to this approach over the latent ODE model in [2] or the SDE model in [3]. For one, unlike [3], the observations x_t are more closely linked with the latent variable ditsributions, only having to pass through the structured inference network as opposed to being used only to estimate an initial sample z_0. Additionally, the propagation of the samples z_i forward through the SDE can be done in parallel. In our testing, we found our approximate method to be at least an order of magnitude faster than the explicit method given in [3].

A disadvantage, however, is that we're not precisely solving the SDE to obtain distributions at each timestep t - rather we're only obtaining sample estimates from the SDE. This gives us only a sample estimate of $p(z_{t+1} \mid z_t)$ rather than an explicit parameterization of the distribution. The distribution $p(z, t)$ of the solution Z to the SDE in equation (4) is governed by a partial differential equation known as the *Fokker-Planck equation*:

$$\frac{\partial}{\partial t}p(z,t) = -\frac{\partial}{\partial z}\left[\mu_\theta(z,t)p(z,t)\right] + \frac{1}{2}\frac{\partial}{\partial z^2}\left[\sigma_\theta^2(z,t)p(z,t)\right]$$

This PDE allows for $p(z, t)$ to have highly complex and potentially non-normal distributions, meaning any attempt to model $p(z_{t+1} \mid z_t)$ as a Gaussian and estimating the parameters would result in a highly restrictive and potentially inaccurate transition density estimate. Thus, we're limited to using sample estimates z_{t+1} from the SDE, and in turn we won't be able to calculate an explicit ELBO loss to train with. To handle this, we introduce what we call a *pseudo-ELBO*.

3.1 Pseudo-ELBO

In [1] the ELBO is factored as

$$\begin{aligned} \text{ELBO} = &\sum_{t=1}^{T} \mathbb{E}_{z_t \sim q(z_t|\vec{x})}[\log p_\theta(x_t \mid z_t)] \\ &- \text{KL}[q(z_0 \mid \vec{x}) \parallel p(z_0)] \\ &- \sum_{t=2}^{T} \mathbb{E}_{z_{t-1} \sim q(z_{t-1}|z_{t-2},\vec{x})}[\text{KL}(q(z_t \mid z_{t-1}, \vec{x}) \parallel p(z_t \mid z_{t-1}))] \end{aligned}$$

In practice the expectations are evaluated using a single sample z_t, and the distributions $q(z_t \mid z_{t-1}, \vec{x})$ and $p(z_t \mid z_{t-1})$ are assumed to have a fixed, parametric form with parameters given by the output of a neural network. As discussed above however, we only have access to sample estimates of z_t from $p(z_t|z_{t-1})$, and we would like to avoid oversimplifying the distribution by parameterizing it as something like a Gaussian. In light of this, we need to make some modifications to the ELBO. Consider the following quantity

$$\begin{aligned} &\sum_{t=1}^{T} \mathbb{E}_{z_t \sim q(z_t|\vec{x})}[\log p_\theta(x_t \mid z_t)] - \text{KL}[p(z_0) \parallel q(z_0 \mid \vec{x})] \\ &- \sum_{t=2}^{T} \mathbb{E}_{z_{t-1} \sim q(z_{t-1}|z_{t-2},\vec{x})}[\text{KL}(p(z_t \mid z_{t-1}) \parallel q(z_t \mid z_{t-1}, \vec{x}))] \end{aligned}$$

The KL divergence is a non-symmetric quantity, and moreover there is no closed formula for $\text{KL}(p \parallel q) - \text{KL}(q \parallel p)$, so the difference between the ELBO and the expression above is non-trivial. By considering the reverse KL, we can use the identity $\text{KL}(p \parallel q) = H(p, q) - H(p)$, where $H(p, q)$ is the *cross-entropy*, evaluated as $\mathbb{E}_p[-\log q]$. Then

$$\begin{aligned} &\text{KL}(p(z_t|z_{t-1}) \mid q(z_t|z_{t-1}, \vec{x})) \\ &\quad = \mathbb{E}_{z_t|z_{t-1}}[q(z_t|z_{t-1}, \vec{x})] - H(p(z_t|z_{t-1})) \\ &\quad \approx q(\hat{z}_t|z_{t-1}, \vec{x}) - H(p(z_t|z_{t-1})) \end{aligned}$$

Since we have a sample estimate of $z_t|z_{t-1}$, namely $\hat{z}_t$, we can estimate the cross-entropy term above, leaving only the entropy term to deal with. We approximate this term with the entropy of a Gaussian with mean and standard deviation equal to that of $p(z_t \mid z_{t-1})$, which can be inferred from the Euler-Maruyama method, giving

$$\approx q(\hat{z}_t|z_{t-1}, \vec{x}) - H\left(N(\mu_{z_t|z_{t-1}}, \sigma^2_{z_t|z_{t-1}})\right)$$

We replace each KL term in the ELBO with this approximation, yielding what we call the *pseudo-ELBO*. While we do not present any formal proof of our derivation of the pseudo-ELBO, our experiments lead us to conjecture that optimizing the pseudo-ELBO is equivalent in some sense to optimizing the traditional ELBO, however further investigation is required.

3.2 Summary

In summary, the architecture we propose is a sort of variational autoencoder where the latent variable is modeled by a neural SDE. We use a structured inference network to obtain rough estimates of the SDE sampled at distinct timesteps, then fine-tune the SDE to capture the temporal dynamics of the latent state distributions. After decoding with a relatively straightforward MLP, we train the model end-to-end using the pseudo-ELBO, an approximation to the traditional ELBO which allows us to train the model far faster than other neural SDE approaches.

4 Experiments

This section will describe how we evaluated our method including the datasets, baselines, and training procedure we used.

4.1 Datasets

4.1.1 Linear Synthetic Dataset

Similar to the paper on DMMs [1], we generated synthetic sequences of observations and latent states using a fixed linear one-dimensional GSSM defined by:

$$z_t \sim \mathcal{N}(z_{t-1} + 0.05, 10)$$
$$x_t \sim \mathcal{N}(0.5 z_t, 20)$$

The training set consisted of 5000 datapoints with a length of 25 time-steps for each.

4.1.2 Non-Linear Synthetic Dataset

We also generated data from a fixed non-linear one-dimensional GSSM defined by:

$$z_t \sim \mathcal{N}(\sin(t), 0.1)$$
$$x_t \sim \mathcal{N}(0.5 z_t, 0.1)$$

Again, the training set contained 5000 datapoints, each with 25 time-steps.

4.1.3 Pharmacodynamic Synthetic Dataset

We generated a synthetic disease progression dataset according to [4]. This dataset has two biomarkers, which follow second-order polynomial trajectories over time. There are two lines of therapy. For the first line of therapy, the drug is withheld, and for the second, the drug is administered. The disease has 4 subtypes, and the subtype influences the patient's untreated progression of the disease.

4.1.4 PhysioNet Dataset

The real healthcare dataset we used is the PhysioNet Computing in Cardiology Challenge data from 2012 [5]. It contains time series data, including measurements of vital signs and laboratory results, for ICU patients over the first 48 hours of their ICU stay. There are about 40 dimensions for each observation for a patient. We use data from 6400 out of the 8000 patients from set A and set B for training.

The dataset is missing many values, and patients' observations are not evenly spaced in time. So, we do significant pre-processing of the data before using it. We first remove invalid values from the data (e.g. Heart Rate > 300 bpm) or convert the value to the correct unit when possible (e.g. convert Temperature measured in degrees Fahrenheit to Celsius). Next, we remove the general descriptors, which are variables that are collected only once on admission including Record ID, Gender, and ICU Unit Type. In order to make the observations for a patient evenly spaced in time, we implemented a discretization method that resamples the time series data at a specified frequency by taking the mean of the samples in each discretized time interval. We normalize each variable across all patients in the dataset to be between 0 and 1. Finally, we impute the missing values using an imputation method such as mean or forward imputation.

TABLE 1
Comparison of Our DMM With Kalman Smoother on Linear Synthetic Dataset

	ELBO	Latent Variable RMSE
Our DMM Implementation	3.193	4.23
Kalman Smoother	3.083 ± 0.0087	4.061 ± 0.0359

TABLE 2
Comparison of Our DMM With Original DMM in [1] on Polyphonic Music Dataset

	ELBO
Our DMM Implementation	6.7355 ± 0.129500
Original DMM	6.926

4.2 Baselines

We compare our SDE method to 3 baseline models.

We implemented an ARIMA(2,1,0) model. This is a classical time-series model known as an AutoRegressive Integrated Moving Average model, where the number of time lags is 2, meaning the current value is based on the previous two values, and the number of differences is 1, meaning that the data is replaced with the difference between the current values and the previous values once in order to make the time series stationary with respect to the mean.

We re-implemented the DMM from Krishnan et al. [1] to use as a baseline, and verified the correctness of our implementation in two ways. First, for the Linear Synthetic dataset, we verified that the DMM's ELBO and latent variable RMSE approach the ELBO and RMSE of a Kalman smoother, which performs exact inference (Table 1).

Second, we tested the DMM on a polyphonic music dataset [6] with 88 features at each timestep corresponding to the notes of a piano. Our implementation of the DMM achieved an ELBO close to that of the implementation in the DMM paper (Table 2).

We also implemented the Latent ODE model from Rubanova et al. [2] as a baseline.

4.3 Parameter Setup and Training Procedure

For the DMM, ODE, and SDE models, we include our training parameters in YAML configuration files in the code. See Table 4 in the Appendix for definitions of the various parameters. In order to improve convergence in training, we used annealing for the KL term in the ELBO. The weight

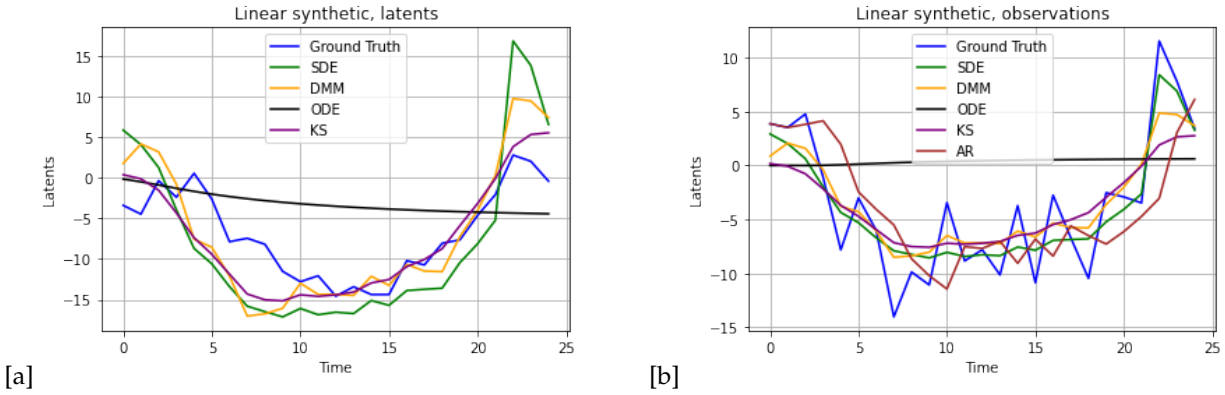

Fig. 6. Comparing ground truth, SDE, DMM, ODE, Kalman Smoother (KS), and ARIMA(2,1,0) (AR) [a] Latent space variables z and [b] Observations x for linear synthetic dataset. Note that there is no latent space representation for AR.

of the KL term in the ELBO starts at 0 during the warm-up period. Then it increases linearly to 1 over a number of epochs specified in the configuration. We also added a weight hyperparameter for the cross entropy term in the KL portion of the pseudo-ELBO in order to make the training more stable.

TABLE 3
Training time for Non-Linear Synthetic Dataset

	Time (seconds)
DMM	145.942
ODE	308.729
SDE	188.066

5 Results

5.1 Evaluation Metrics

We compare the DMM, ODE, and SDE using Rolling Window RMSE and training time. The Rolling Window RMSE is the RMSE for the observations at varying time-steps into the future. Lower RMSE is better. Note that since we are using a pseudo-ELBO for our SDE method, we cannot compare the exact ELBO across methods.

5.2 Results and Analysis

For SDE and each of the baselines, we plotted the latent space variables and observations for the linear synthetic dataset (Figure 6) and non-linear synthetic dataset (Figure 8). The rolling window RMSEs for the linear and non-linear synthetic datasets are shown in Figure 7 and Figure 9 respectively.

For the linear synthetic dataset, the SDE and DMM latents and observations follow the pattern of the ground truth quite well. ARI is our simplest baseline. It has no latent space representation, but still performs well, even with high noise. However, the ODE method fails due to the high noise. The SDE achieves the lowest rolling window RMSE for window shifts further into the future. This aligns with our findings that SDEs can generalize well on out-of-distribution data. See Appendix A for details on the generalization experiments we performed.

For the non-linear synthetic dataset, there is no ground truth emission function, so the latents across the different methods don't necessarily need to match, but the observations should match across methods. In Figure 8, we observe that although the latents for ODE do not match the ground truth, the observations do match the general pattern of the ground truth. ARI has the lowest rolling window RMSE at time steps in the future. This is because we used an AR(2) model, which can fit the underlying non-linear GSSM perfectly. We included the training time for DMM, ODE, and SDE on the non-linear synthetic data in Table 3. ODE is significantly slower than our implementation of the SDE, which is expected because the ODE method requires using complicated solvers, whereas for our implementation of SDE, we use less complex solvers that are augmented with checkpoints to help them stay on track.

Both the SDE and DMM match the observations well for biomarker 1 and 2 of the pharmacodynamic synthetic dataset, as shown in Figure 10. The SDE has lower RMSE than DMM across all window shifts (Figure 11).

We present a plot of the rolling window RMSE of DMM and SDE on the PhysioNet dataset in Figure 12. The RMSE for SDE is lower than that of DMM. However, the SDE RMSE is almost constant across the window shifts, which may indicate that the SDE is finding a trivial solution. We will need to do further investigation to see why this could be occurring. Additional pre-processing for the PhysioNet dataset might be required. Once we achieve more meaningful results for PhysioNet, we would also like to study the impact of using different imputation methods on the performance of our SDE method.

6 Limitations

There are three primary limitations with our approach. For one, we use a structured inference network to obtain initial estimates of the distributions $q(z_{t+1}|z_t, \vec{x})$, but structured inference networks suffer from the same problem as traditional RNNs in that they *ignore* temporality - samples are assumed to be equally spaced throughout time. We mitigate this by including Δt_i as input to the structured inference network, but there are better approaches to this problem. For instance, one could use an ODE-RNN as in [2] as the encoder instead of a structured inference network, as the

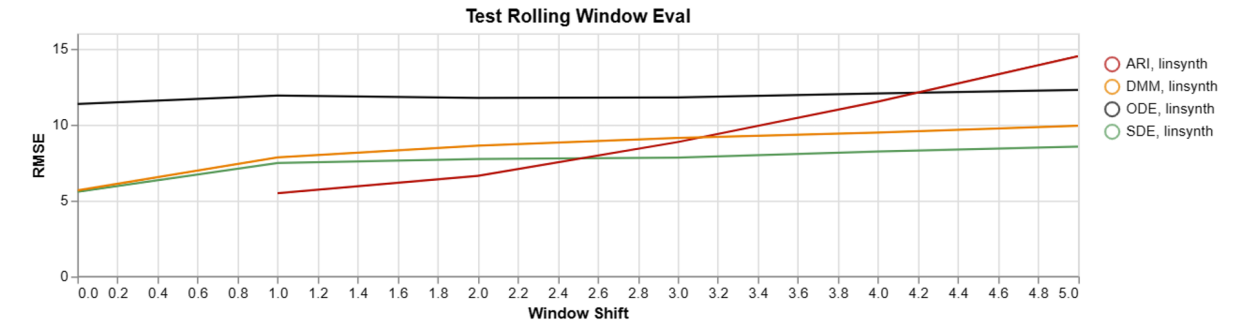

Fig. 7. Rolling window RMSE for the Linear Synthetic dataset using ARIMA(2,1,0) (ARI), DMM, ODE, and SDE. Window shifts between 0 to 5 are plotted.

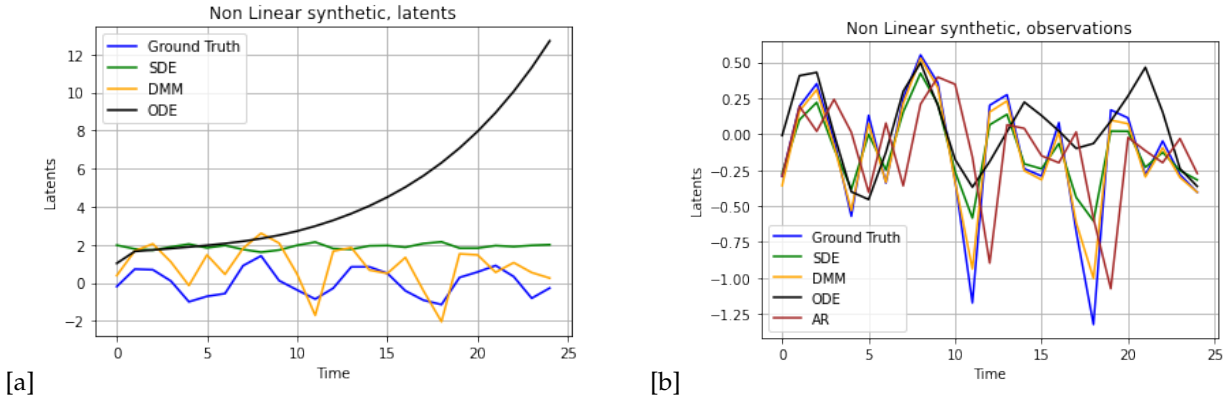

Fig. 8. Comparing ground truth, SDE, DMM, ODE, and ARIMA(2,1,0) (AR) [a] Latent space variables z and [b] Observations x for non-linear synthetic dataset. Note that there is no latent space representation for AR.

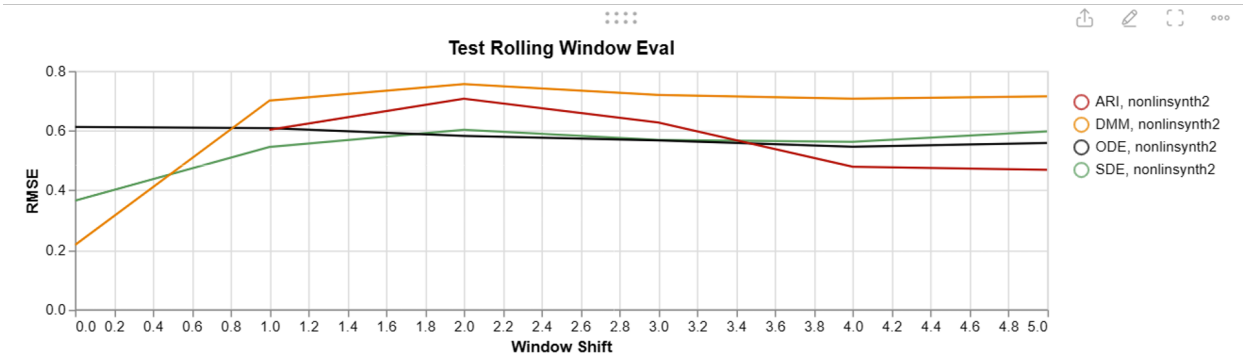

Fig. 9. Rolling window RMSE for the Non-Linear Synthetic dataset using ARIMA(2,1,0) (ARI), DMM, ODE, and SDE. Window shifts between 0 to 5 are plotted.

ODE-RNN is explicitly designed to mitigate this problem. This approach requires further investigation.

The second main limitation with our approach is the pseudo-ELBO. We unfortunately were unable to give any formal justification as to why optimizing the pseudo-ELBO is in any way equivalent to a traditional ELBO, despite our experiments demonstrating that to be the case. Nonetheless, there may be scenarios in which the pseudo-ELBO fails to produce a well-performing model, which we have yet to encounter in testing.

Additionally, in approximating solutions to the latent SDE with the Euler-Maruyama method we use a fixed number of timesteps K irrespective of the length of the interval. This reduces the computational cost of the forward pass (as it allows every application of the Euler-Maruyama method to be calculated in parallel), but may also introduce error into the approximation for larger intervals. It may be worth investigating how a variable number of timesteps K (or perhaps a fixed timestep size, $\Delta\tau$) affects the efficacy of the model.

A limitation of all the approaches mentioned in this paper is how to handle *missing* time-series data, not simply irregularly sampled data. At each sample time we assume we are given a vector of data in $\mathbb{R}^n$, however in most real

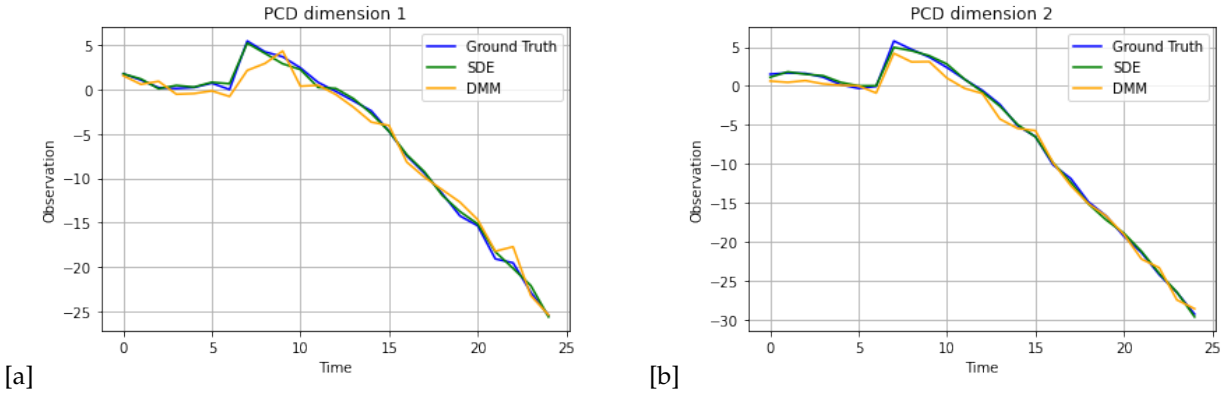

Fig. 10. Comparing ground truth, SDE, and DMM Observations for [a] biomarker 1 and [b] biomarker 2 for Pharmacodynamic synthetic dataset.

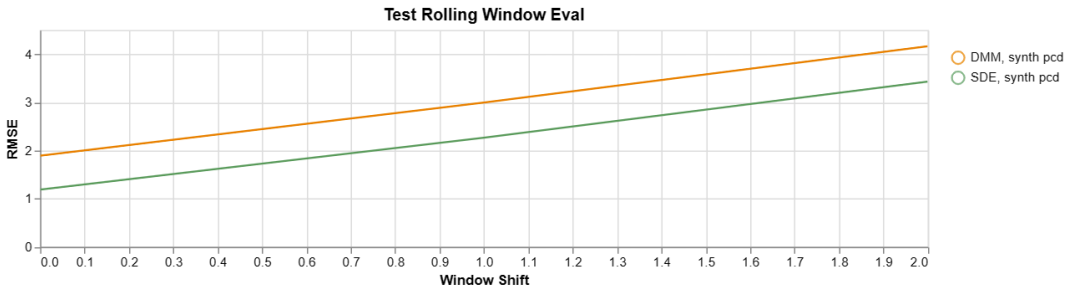

Fig. 11. Rolling window RMSE for the Pharmacodynamic synthetic dataset using DMM and SDE. Window shifts between 0 to 5 are plotted.

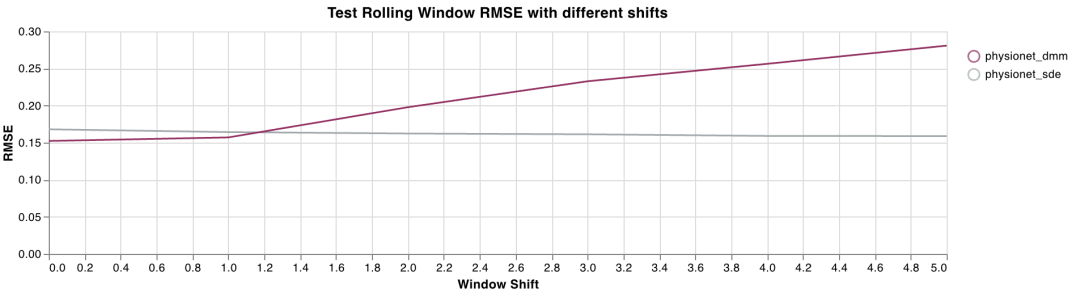

Fig. 12. Rolling window RMSE for the PhysioNet dataset using DMM and SDE. Window shifts between 0 to 5 are plotted.

world settings the data is better modeled as an element of $(\mathbb{R} \cup \{\text{N/A}\})^n$), with "N/A" used to denote a missing measurement. While there is a vast literature on imputing time series, and one could even impute using a pretrained neural latent SDE architecture, it is worth assessing if this assumption about the data can be "baked in" to the training process, so that this data can be used for further training/fine-tuning.

7 Conclusion

In this paper, we have explored a new method for learning latent neural stochastic differential equations, aimed at enhancing scalability compared to existing approaches. Our technique leverages structured inference networks, which serve to predict checkpoints across time intervals. This method represents a shift from the conventional direct solving from start to end time, employing a continuous-time stochastic transition function for integration between these checkpoints. This strategy enables the use of more computationally efficient solvers without significantly compromising the model's performance.

We tested our method on a variety of synthetic datasets, including one specifically designed to simulate healthcare-related data. The results were promising, indicating the potential applicability of our approach in real-world healthcare scenarios.

We also recognize and discuss certain limitations of our approach, understanding that there is always room for improvement and refinement. The exploration of these limitations is crucial for the ongoing development of our method and its future applications in the complex and ever-changing field of machine learning in healthcare.

REFERENCES

[1] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," 2016.

[2] Y. Rubanova, R. T. Q. Chen, and D. Duvenaud, "Latent odes for irregularly-sampled time series," 2019.

[3] X. Li, T.-K. L. Wong, R. T. Q. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 3870–3882. [Online]. Available: https://proceedings.mlr.press/v108/li20i.html

[4] Z. Hussain, R. G. Krishnan, and D. Sontag, "Neural pharmacodynamic state space modeling," 2021.

[5] I. Silva, G. Moody, R. Mark, and L. A. Celi, "Predicting mortality of icu patients: The physionet/computing in cardiology challenge 2012," 2012. [Online]. Available: https://physionet.org/content/challenge-2012/1.0.0/

[6] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," 2012.

APPENDIX

Training Parameters

TABLE 4
Description of Training Parameters

Parameter Name	Description
batch_size	Batch size for each iteration
lr	Learning rate for Adam optimizer
n_epochs	Number of training epochs
model_params	
st_net_n_layers / n_inference_layers	Number of hidden layers in inference network
st_net_hidden_dim / inference_hidden_size	Dimension of hidden layers in inference network
transition_hidden_dim / transition_hidden_size	Dimension of hidden layers in transition network
emission_hidden_size / emission_hidden_dim	Dimension of hidden layers in emission network
n_euler_steps	Number of steps used in the euler-maruyama
loss_params	
entropy_weight	Weight for the entropy of the transition function term in the Pseudo-ELBO
entropy_q_weight	Weight for the entropy of the inference network (as a regularizer)
elog_weight	Weight for the cross-entropy term in the Pseudo-ELBO
n_entropy_samples	Number of samples to use for estimating the entropy of $p(z_t \mid z_{t-1})$
loss_params > annealing_params	
warm_up	Number of epochs before starting annealing
n_epochs_for_full	Number of epochs after warm-up until reaching weight of 1 for KL term

Generalization Experiments

We conducted experiments to demonstrate that SDEs generalize well on data that is out-of-distribution. For one experiment, we train an SDE to transform 2-dimensional samples that look like a cosine at time $t = 0$ into two-dimensional samples that look like a sine at time $t = 1$ using maximum likelihood (Figure 13). Given the distribution at $t = 0$, the SDE is able to produce the distribution that looks like a sine at $t = 1$. The SDE can continue the pattern at future time-steps by continuing to shift the distribution to the right, despite only being trained on the data going from cosine to sine. Compared to normalizing flows which can transform between distributions, the SDE generalizes better for time-steps further into the future, as shown in Figure 14. We also demonstrated this generalization property of SDEs on higher dimensional data. These experiments allowed us to hypothesize that SDEs would be better than previous methods especially in the task of prediction. We successfully observed this behavior in the various rolling window RMSE metrics presented in the evaluation section.

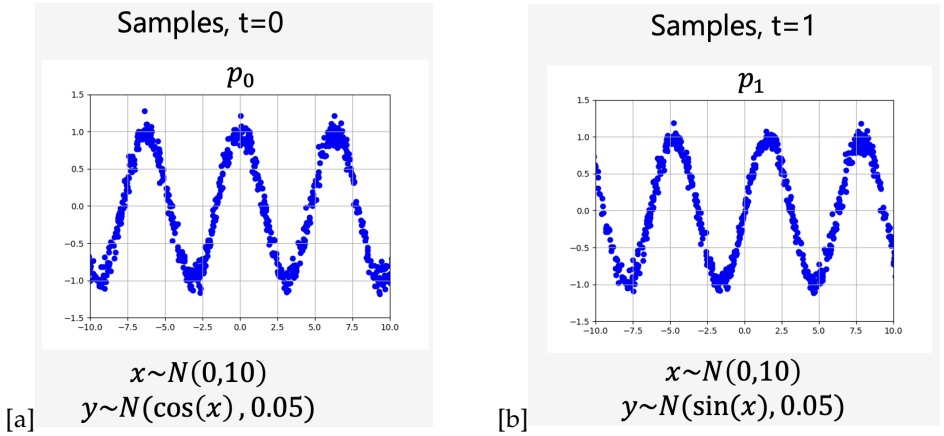

Fig. 13. Generalization Experiment for SDEs: Samples at [a] t=0 and [b] t=1

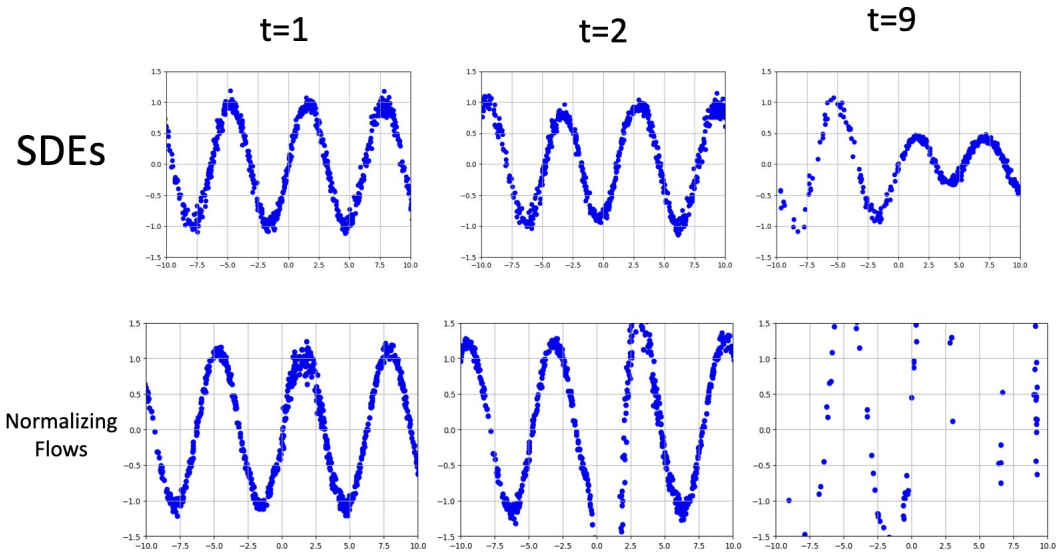

Fig. 14. Generalization Experiment for SDEs: Comparing SDEs with Normalizing flows at time t=1, t=2, t=9