Learning Continuous-Time Dynamics by Stochastic Differential Networks

Yingru Liu
Stony Brook University
Stony Brook, NY 11790

Yucheng Xing
Stony Brook University
Stony Brook, NY 11790

Xuewen Yang
Stony Brook University
Stony Brook, NY 11790

Xin Wang
Stony Brook University
Stony Brook, NY 11790

Di Jin
CSAIL, MIT
Cambridge, MA 02139

Jing Shi
University of Rochester
Rochester, NY 14627

Abstract

Learning continuous-time stochastic dynamics from sparse or irregular observations is a fundamental and essential problem for many real-world applications. However, for a given system whose latent states and observed data are high-dimensional, it is generally impossible to derive a precise continuous-time stochastic process to describe the system behaviors. To solve the above problem, we apply Variational Bayesian method and propose a flexible continuous-time framework named *Variational Stochastic Differential Networks (VSDN)*, which can model high-dimensional nonlinear stochastic dynamics by deep neural networks. VSDN introduces latent states to modulate the estimated distribution and defines two practical methods to model the stochastic dependency between observations and the states. The first variant, which is called VSDN-VAE, incorporates sequential Variational Auto-Encoder (VAE) to efficiently model the distribution of the latent states. The second variant, called VSDN-SDE, further extends the model capacity of VSDN-VAE by learning a set of Stochastic Differential Equations (SDEs) to fully describe the state transitions. Through comprehensive experiments on symbolic MIDI and speech datasets, we show that VSDNs can accurately model the continuous-time dynamics and achieve remarkable performance on challenging tasks, including online prediction and sequence interpolation.

1 Introduction

Many real-world systems experience complicated stochastic dynamics over a continuous time period. The challenges on modeling the stochastic dynamics mainly come from two sources. First, the underlying state transitions of many systems are often uncertain, as they are placed in unpredictable environment with their states continuously affected by unknown disturbances. Second, the monitoring data collected may be sparse and at irregular intervals as a result of the sampling strategy or data corruption. The sparse or irregular data sequence loses a large amount of information and system behaviors hidden behind the intervals of the observed data. In order to accurately model and analyze dynamics of these systems, it is important to reliably and efficiently represent the continuous-time stochastic process based on the discrete-time observations.

In some domains, the derivation of the continuous-time stochastic model relies heavily on human knowledge and many studies focus on its inference problem [1, 2]. But in more domains (e.g., video analysis [3] and human activity detection [4]), it is difficult and sometimes intractable to derive an accurate model to capture the underlying temporal evolution from the collected sequence of data. Although some studies have been made on approximating the stochastic process from data collected,

Preprint. Under review.

the majority of these methods define the system dynamics with a linear model [5–7], which can not well represent high-dimensional data with nonlinear relationship. Recently, the Neural Ordinary Differential Equation (ODE) studies [4, 8–10] introduce deep learning models to learn an ODE and apply it to approximate continuous-time dynamics. Nevertheless, these methods generally posit simplified assumptions on the data distribution (e.g. Gaussian), which strongly limits their capability of modeling complicated continuous-time stochastic processes.

In this paper, we propose a new deep learning framework called ***Variational Stochastic Differential Network (VSDN)*** that can effectively model the continuous-time stochastic dynamics based only on sparse or irregular observations. Taking advantage of the capacity of deep neural networks, VSDNs have higher flexibility and generalizability in modeling the nonlinear stochastic dependency from high-dimensional observations. As a core design strategy, VSDN incorporates the latent state trajectory to capture the underlying factors that can track the system dynamics. The trajectory helps to more flexibly model the data distribution and more accurately generate the output data than Neural ODEs. Along with the framework, we propose two variants to efficiently learn the stochastic dependency between the observations and the underlying dynamics: 1) VSDN-VAE, which directly models the conditional distribution of the latent state through Auto-Encoding Variational Bayes [11], and 2) VSDN-SDE, which exploits deep neural networks to learn a set of Stochastic Differential Equations (SDEs) that describe the latent dynamics. The experimental studies show that VSDNs outperform state-of-the-art deep learning methods on modeling the continuous-time dynamics of symbolic MIDI and speech, and achieve remarkable performance in the tasks of prediction and interpolation. For instance, we observe a relative performance gain of 58.34% in MIDI (accuracy: $49.98\% \rightarrow 79.14\%$) and 25.49% in speech (error: $0.1165 \rightarrow 0.0868$) in the interpolation task.

The rest of this paper is organized as follows. We first introduce the related works in Section 2. In section 3, we propose the architecture of VSDN-VAE. In Section 4, we extend VSDN-VAE into VSDN-SDE and define a variational inference method to train our model. The experiments are presented in section 5 and conclusion is given in section 6.

2 Related Works

2.1 Neural Ordinary Differential Equations

Neural ODE is first proposed in [8], where the hierarchical structure of residual network [12] is replaced by an ordinary differential equation. An extended model called Latent ODE is further proposed for sequential modeling. The significant drawback of latent ODE is that it encodes the whole observation sequence into the initial condition of the ODE, which is not expressive enough to capture all the information of the data. To solve this problem, ODE-RNN [4] and GRU-ODE [10] integrate ODE into Recurrent Neural Network (RNN) and encode the data sequence by the hidden feature trajectory. Nevertheless, ODE-RNN and GRU-ODE posit simple parameterization of the data distribution (e.g. Gaussian and Bernoulli distributions), and do not include the continuous-time latent states. Therefore, they are not able to represent multi-modal distributions, which are present frequently in high-dimensional time series. They are also unable to extract explanatory latent factors from data, which is one of the main applications of generative models. In Neural Jump Stochastic Differential Equation (NJSDE) [9], temporal point process is integrated into Neural ODE to capture the effect of stochastic events. NJSDE is parallel to our study, as it is proposed to solve the event detection problem, while we aim to model the stochastic dynamics that generates the observation data.

3 Variational Stochastic Differential Network

In this section, we present in details our first design of the Variational Stochastic Differential Network (VSDN), which is called VSDN-VAE. An extended models called VSDN-SDE will be introduced in Section 4. VSDN introduces the latent state to capture the underlying unobservable factors that generate the observed data. As an additional benefit, the latent state empowers VSDN to learn a flexible parameterization of the data distribution.

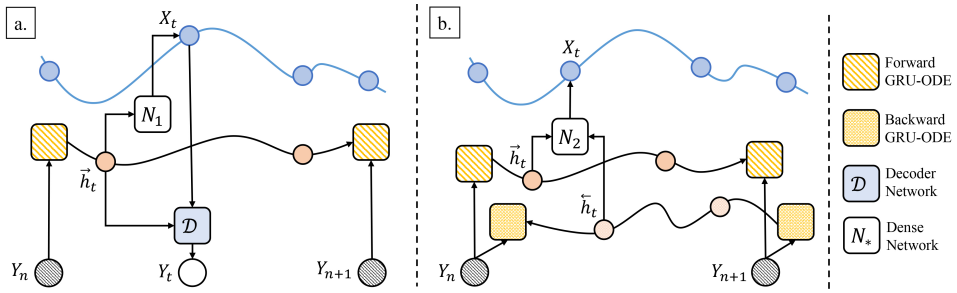


Figure 1: Model Architectures of VSDN: (a). Generative Model $\mathcal{G}$; (b). Inference Model $\mathcal{Q}$.

3.1 Basic Notations and Problem Formulation

Throughout this paper, we define $X_t \in \mathbb{R}^{d_1}$ as the continuous-time latent state at time t and $Y_n \in \mathbb{R}^{d_2}$ as the n_{th} discrete-time observed data at time t_n. d_1 and d_2 are the dimensions of the latent state and observation, respectively. $X_{<t}$ is the continuous trajectory before time t and $X_{\leq t}$ is the path up to time t. $Y_{n_1:n_2}$ is the sequence of data points and $X_{t_{n_1}:t_{n_2}}$ is the continuous-time state trajectory from t_{n_1} to t_{n_2}. $\mathcal{Y}_t = \{Y_n | t_n < t\}$ is the historical observations before t and $\mathbb{Y}_t = \{Y_n | t_n \geq t\}$ is the current and future observations. For simplicity, we also assume that the initial value of the latent state is constant. The results in this paper can be easily extended to the situation that the initial states are also random variables. Given K data sequences $\{y^{(i)}_{1:n_i}\}, i = 1, \cdots, K$, the target of our study is to learn an accurate continuous-time generative model $\mathcal{G}$ that maximizes the log-likelihood:

$$\mathcal{G} = \arg\max_{\mathcal{G}} \frac{1}{K} \sum_{i=1}^{K} \log P_{\mathcal{G}}(y^{(i)}_{1:n_i}). \tag{1}$$

For high-dimensional sequential data, there exists a complicated nonlinear relationship between the observed data and the unobservable latent state, which can be either the physical state of a dynamic system or the low-dimensional manifold of data. In our study, the latent state evolves in the continuous time domain and generates the observation through some transformation.

3.2 Framework Architecture

The whole framework is shown in Figure 1. VSDN consists of a generative model $\mathcal{G}$ and an inference model $\mathcal{Q}$. The generative model $\mathcal{G}$ captures the generating process of the data sequence to perform online ***prediction***. It exploits both latent states and the historical observations to predict the observation at a specific time t. The inference model $\mathcal{Q}$ works offline to infer the latent state trajectory based on the overall observation sequence. Theoretically, $\mathcal{Q}$ is defined to train $\mathcal{G}$ by variational inference methods, because the exact log-likelihood of $\mathcal{G}$ is intractable [11]. The inferred latent states of $\mathcal{Q}$ embeds the information of the whole sequence and can be applied in the generative model to generate observation data as well. Therefore, $\mathcal{Q}$ is itself an accurate and efficient model for ***interpolation***.

The architectures of $\mathcal{G}$ and $\mathcal{Q}$ are constructed by the subsets of five deep learning components: forward GRU-ODE [10], backward GRU-ODE, decoder network and two encoder networks that synthesize the latent states. The detailed descriptions are given as follows.

Generative Model $\mathcal{G}$: The generative model describes the stochastic dependency of the latent state on the historical observations and the conditional distribution that generates the data (i.e $P(X_t|\mathcal{Y}_t)$ and $P(Y_t|X_t, \mathcal{Y}_t)$). We first define a forward GRU-ODE to explore the information of historical observations and extract a hidden feature trajectory. We then incorporate deep neural networks to directly learn these two distributions:

$$P_{\mathcal{G}}(X_t|\mathcal{Y}_t) = \mathcal{N}(\mu_{0,t}, \sigma^2_{0,t}), \quad [\mu_{0,t}, \sigma_{0,t}] = N_1(\overrightarrow{h_t}), \tag{2}$$

$$P_{\mathcal{G}}(Y_t|X_t, \mathcal{Y}_t)) = \mathcal{P}(Y_t|\theta), \quad \theta = \mathcal{D}(X_t, \overrightarrow{h_t}), \tag{3}$$

where $\overrightarrow{h_t}$ is the extracted feature of the forward GRU-ODE. $\mathcal{N}(\cdot, \cdot)$ denotes a Gaussian distribution and its parameters are learned by an encoder network N_1. $\mathcal{P}$ is the parameterization of the conditional

distribution, which is determined by the type of data. The decoder network $\mathcal{D}$ is introduced to compute the parameters for $\mathcal{P}$.

Although the generative model of VSDN-VAE has a strong assumption that the latent state at any time follows a Gaussian distribution, which seems to be limited, our experimental results show that the flexibility of deep learning models can alleviate this constraint to achieve good performance in real-world applications.

Inference Model $\mathcal{Q}$: the inference model depicts the posterior distribution of the latent trajectory after observing the complete data sequence that are taken from both past and future. According to d-separation [13], the latent state X_t is dependent on both the historical data $\mathcal{Y}_t$ and future observations $\mathbb{Y}_t$. Therefore, we take a forward-backward bidirectional scheme in $\mathcal{Q}$. While the historical information is embedded by the forward GRU-ODE, we further define a backward GRU-ODE (Fig. 1 (b)) to extract hidden features from the future data, and compute the posterior distribution of latent state as:

$$P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t) = \mathcal{N}(\mu_{1,t}, \sigma_{1,t}^2), \quad [\mu_{1,t}, \sigma_{1,t}] = N_2(\overrightarrow{h_t}, \overleftarrow{h_t}), \tag{4}$$

where $\overleftarrow{h}_t$ denotes the hidden feature of the backward GRU-ODE. N_2 is the encoder network that integrates the hidden features of the forward and backward GRU-ODEs to compute the parameters of the posterior distribution. In the next part, we present the variational Bayesian method to jointly train $\mathcal{G}$ and $\mathcal{Q}$.

3.3 Differentiable Evidence Lower Bound

The exact log-likelihood of the generative model is given as

$$\log P_{\mathcal{G}}(y_{1:n_i}) = \log \int \prod_{i=1}^{n} P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i}) dX_{t_i}, \tag{5}$$

which does not have the closed-form solution in general. Therefore, $\mathcal{G}$ can not be directly trained by maximizing log-likelihood. Based on Auto-Encoding Variational Bayes [11], we can train our models by the Evidence Lower Bound (ELBO) of the log-likelihood:

$$\mathcal{L}(y_{1:n_i}) = -\beta \sum_{i=1}^{n} KL\Big(P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i}) || P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})\Big) + \mathbb{E}_{P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i})} \log P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}), \tag{6}$$

where $P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})$, $P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i})$ and $P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i})$ are given in Eqs. (2) - (4). $KL(\cdot||\cdot)$ denotes the KL divergence between two distributions and β is a hyper-parameter to weight the effect of the KL terms. As the training loss is similar to that of a discrete-time sequential VAE, we name our first framework as VSDN-VAE. In both the generative and inference models in VSDN-VAE, we define the distribution of the latent state as Gaussian, which may limit the capacity of VSDN-VAE. In section 4, we apply Stochastic Differential Equations (SDEs) to further improve the performance of our method.

4 Variational Stochastic Differential Network with SDE

VSDN-VAE posits that the latent state at any time follows Gaussian distribution, which in principle assumes that the temporal transition of the latent state is defined by a linear Stochastic Differential Equation (SDE). In order for our model to more accurately and flexibly track practical system dynamics, we further propose an extended framework called VSDN-SDE, which represents the latent state dynamics by nonlinear SDEs.

4.1 Neural Stochastic Differential Equations

While VSDN-VAE explicitly models the distributions of the latent state in the generative and inference models, VSDN-SDE applies neural networks to learn the nonlinear SDEs that implicitly induce the distributions. The latent state transition of the ***generative model*** $\mathcal{G}$ is given by

$$dX_t = H_{\mathcal{G}} dt + R_{\mathcal{G}} dW_t, \quad [H_{\mathcal{G}}, R_{\mathcal{G}}] = N_1(\overrightarrow{h_t}, X_t), \tag{7}$$

where $H_{\mathcal{G}}$ and $R_{\mathcal{G}}$ are called the drift and diffusion of SDE. Similarly, the latent state transition of the ***inference model*** $\mathcal{Q}$ is defined as

$$dX_t = H_{\mathcal{Q}}dt + R_{\mathcal{G}}dW_t, \quad H_{\mathcal{Q}} = N_2(\overrightarrow{h_t}, \overleftarrow{h_t}, X_t). \tag{8}$$

Eq. (8) is restricted to having the same diffusion function as Eq. (7). A feasible ELBO can not be defined to train VSDN-SDE without this restriction, as the KL divergence of two SDEs with different diffusions will be infinite [14].

4.2 Continuous-Time Evidence Lower Bound

As VSDN-SDE represents the latent state transitions by SDEs, both $P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i})$ and $P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})$ become intractable. Therefore, ELBO used in VSDN-VAE can not be applied to train VSDN-SDE. Intead, we define an alternative ELBO by replacing the KL term in Eq. (6) with the KL divergence between the two SDEs, which is given by an integral in the time domain [14]:

$$KL(P_{\mathcal{Q}}||P_{\mathcal{G}}) = \frac{1}{2}\int_0^{t_n} \mathbb{E}_{X_t\sim\mathcal{Q}_t}\Big((H_{\mathcal{Q}} - H_{\mathcal{G}})^T[R_{\mathcal{G}}R_{\mathcal{G}}^T]^{-1}(H_{\mathcal{Q}} - H_{\mathcal{G}})\Big)dt. \tag{9}$$

where $\mathcal{Q}_t(X_t)$ is the marginal distribution of X_t in the inference SDE.

4.3 Discussion

VSDN-VAE and VSDN-SDE have different advantages and drawbacks. Besides modeling the latent states with different methods, VSDN-VAE and VSDN-SDE take different sampling strategies to compute the expectation in the ELBO Eq. (6). When VSDN-VAE explicitly parameterizes the distribution $P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i})$, it is only required to generate random samples of the latent states at observation time. VSDN-SDE, instead, has to synthesize the whole latent trajectory according to the inference SDE, which injects more randomness into the training loss. Therefore, VSDN-SDE can model the distribution of system dynamics in a more general format, while the training process of VSDN-VAE is more stable.

5 Experiments

In this section, we conduct comprehensive experiments to validate the performance of our models and demonstrate its advantages in real-world applications. We compare the performance of VSDN with two state-of-the-art models: ODE-RNN [4] and GRU-ODE [10]. For VSDN, we also introduce the residual learning, where the encoder network N_2 is applied to learn the differences $(\mu_{1,t} - \mu_{0,t})$ and $(H_{\mathcal{Q}} - H_{\mathcal{G}})$. In [15], it appears that the residual learning improves the performance of discrete-time VAEs, with its simplification in the optimization of the KL divergence in ELBO.

5.1 Music Synthesis

We first evaluate the performance of VSDN on modeling symbolic MIDI data, which are widely used to store polyphonic music. We consider three different tasks, including density estimation, prediction and interpolation. For the density estimation, we use the average Negative Log-Likelihood (NLL) per frame as the evaluation metric. As NLLs of VSDN are intractable, we report the negative ELBO instead. For the other tasks, we use the average accuracy (ACC$\%$) of frame-wise pitch prediction [16] as the metric. Same as [16], we do not consider True Negative (TN) in computing the accuracy, as the symbolic MIDI sequence generally has sparse values. The detailed configurations of the models are given in Appendix A.

Data pre-processing: We evaluate the models on the Lakh MIDI dataset [17], which contains over 19000 clean MIDI files. We follow the preprocessing steps in [18]. Each MIDI file is segmented by one minute. A segment is transferred into piano-rolls and forms a sequence of data frames. Data in each frame are held by a 128-dimensional binary vector to represent the activated pitches at different time instants. During the training, we synthesize the discrete-time observations by taking samples at irregular time intervals, and test model performances using the samples at regular intervals. In experiments, we consider two different ratios between the sampled data and the whole sequence: 25% and 50%.

Performance: The model performance is shown in Table 1. We can see that no matter the sampling rate is 25% or 50%, both VSDN-VAE and VSDN-SDE have significant performance improvements compared to the reference schemes. For the density estimation, our VSDN models reduce the negative ELBO to a very low magnitude. In the case of prediction, even though the accuracy of peer schemes has already reached 93.32%, our models increase this value by about 5.48%. In the interpolation task, VSDN models has up to 58.34% (49.98 $\rightarrow$ 79.14) relative performance gain in 50% case. Comparing VSDN-VAE and VSDN-SDE, we find that using residual training is not always beneficial and the performance depends on the architecture of the model and the type of data. Specifically, the improvement of residual learning is related to the optimization of the KL divergence in ELBO. When the models have difficulty in minimizing the KL divergence, exploiting residual learning can improve the training process and performance. Otherwise, residual learning will make the KL divergence vanishes into zero quickly. As a result, the inference model will be identical with generative model, and can not extract information from future observations. Besides, although both models have satisfactory performances, we find that our extended VSDN-SDE achieves a little bit better NLL than our original VSDN-VAE model.

Table 1: Negative Log-Likelihood and Accuracy Comparison on Lakh MIDI dataset.

	NLL		Prediction		Interpolation	
	25 %	50 %	25 %	50 %	25 %	50 %
ODE-RNN	10.67	8.76	93.32	92.72	40.63	49.53
GRU-ODE	10.18	8.35	91.70	91.35	41.33	49.98
VSDN-VAE	0.002	0.002	**98.80**	**98.80**	**54.96**	74.06
VSDN-VAE (res)	0.007	0.004	98.71	98.79	51.27	73.91
VSDN-SDE	0.003	**0.001**	98.79	**98.80**	54.69	74.08
VSDN-SDE (res)	**0.001**	**0.001**	**98.80**	**98.80**	54.82	**74.14**

After training with a fixed number of points, we evaluate how the number of observations influences the model performance. In this experiment, we fix the total length of the sequence, and obtain different numbers of observations by changing the sampling rate. A comparison between ODE-RNN, GRU-ODE and our models is shown in Figure 2. For the average NLL, our VSDN-VAE and VSDN-SDE have obtained a consistently better value. Although the change trend of GRU-ODE is much more obvious, there is still a huge gap between the performances of the two models.

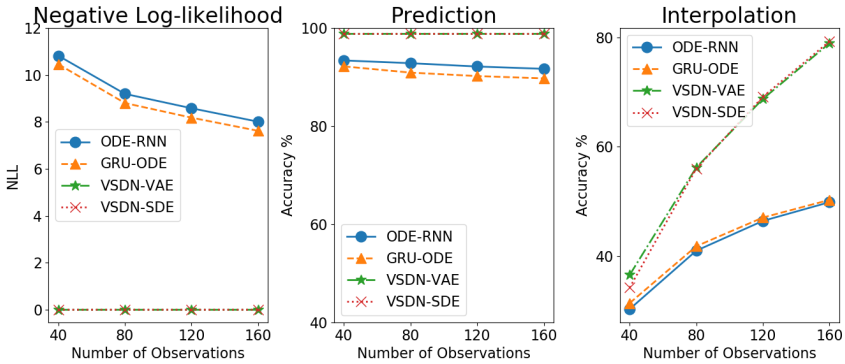


Figure 2: The model performance with respect to the number of observations (trained in 25% case).

Theoretically, more observations could potentially provide more information about the data sequence, which in turn helps the model to infer the future observations. However, the frames of MIDI data are binary vectors, which may not have a smooth trend among consecutive frames as in continuous data. Therefore, historical information brought by more observations makes the observed patterns more complicated, and results in greater challenges for the prediction task. According to Figure 2, ODE-RNN and GRU-ODE are less capable of learning the complicated pattern of the activated pitches, and has a performance drop in the prediction task for a larger number of observations.

Instead, VSDN-VAE and VSD-SDE are more stable to handle the complicated patterns. For the interpolation task, increasing the number of observations makes the task easier. However, ODE-RNN and GRU-ODE have lower accuracy in modeling the complicated pattern of the data and experience a large error on reconstructing the observations. Therefore, the improvement of ODE-RNN and GRU-ODE from more observations is much smaller than that of our models.

Visualization: We visualize some interpolation examples for qualitative evaluation, which are shown in Figure 3. Compared with existing models, VSDNs can recover more details from the sparse observations. As shown in the first column, existing models can only recover the content of the ground-true sequence when the data has more stable patterns.

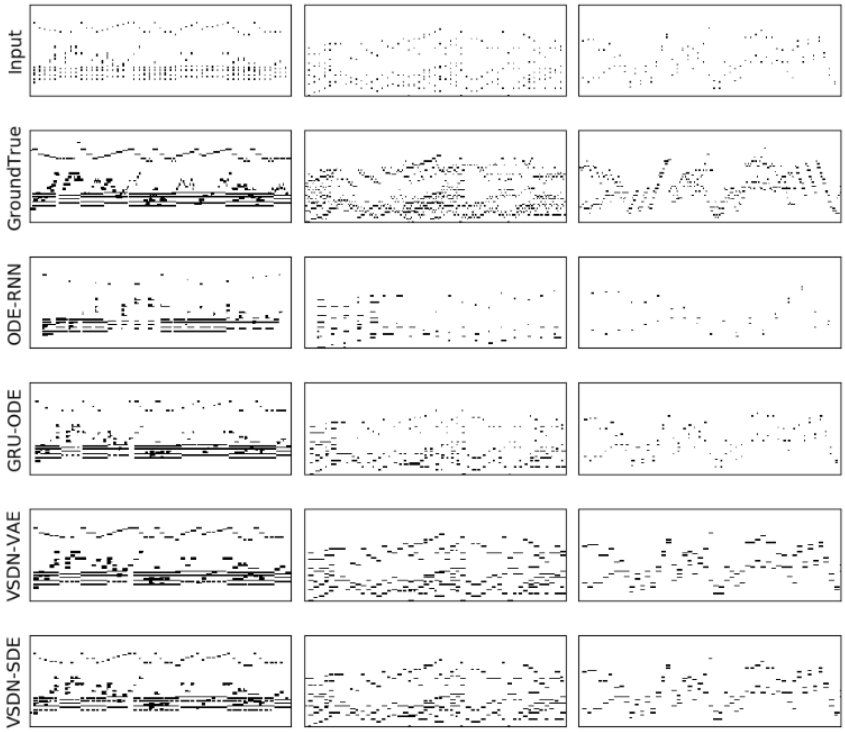


Figure 3: Examples of the interpolation tasks. For each image, the column represents a frame at specific time and the horizontal direction is the time axis.

5.2 Speech Synthesis

We further evaluate our proposed methods on speech synthesis. We intend to directly model the waveforms of the speech in TIMIT* dataset. For the density estimation, we use the average Negative Log-Likelihood (NLL) per dimension as the evaluation metric. For the other tasks, we use the average Rooted Mean Square Error (RMSE) per sequence as the metric. The detailed model configurations are given in Appendix B.

Data pre-processing: TIMIT dataset contains speech waveform files of about 6300 phonetically rich sentences. Before the experiments, these files are segmented by one second with 16kHz sampling frequency (i.e. 16000 points per segment). We group each 100 consecutive points as one data frame, so every sequence has 160 such frames totally. Similarly, from each sequence, we take irregular samples for training and regular samples for testing. We also consider 25% and 50% cases in the experiments.

*https://catalog.ldc.upenn.edu/LDC93S1

Table 2: Negative Log-Likelihood and RMSE Comparison on TIMIT dataset.

	NLL		Prediction		Interpolation	
	25 %	50 %	25 %	50 %	25 %	50 %
ODE-RNN	-0.499	-0.500	0.1166	0.1165	0.1166	0.1165
GRU-ODE	-0.500	-0.500	0.1154	0.1157	0.1166	0.1165
VSDN-VAE	-0.655	-0.655	0.0518	0.0508	0.1044	0.0902
VSDN-VAE (res)	**-0.673**	-0.667	**0.0341**	0.0404	**0.1030**	0.0878
VSDN-SDE	-0.669	**-0.672**	0.0379	**0.0351**	0.1033	**0.0868**
VSDN-SDE (res)	-0.664	-0.667	0.0432	0.0405	0.1039	0.0876

Performance: The evaluation results are given in Table 2. Compared to previous ODE-RNN and GRU-ODE, our models gain consistently better performance, no matter in 25% case or 50% case. In the prediction case, our model reduce the error rate into half. As for the comparison among our own models, we can see that without residual learning, VSDN-SDE performs better than VSDN-VAE, which confirms the effectiveness of nonlinear modeling of the latent states in our VSDN-SDE. After adding the residual learning strategy, VSDN-VAE obtains a certain degree of improvement. However, since TIMIT dataset has continuous and real values, there is a relatively obvious change trend among consecutive data points. As original VSDN-SDE is already good enough to model the sequence, the addition of residual learning may cause KL divergence vanishment, as we explained in Section 5.1, and in turn reduce the model performance. But, we still find that even with the benefit from residual learning, the improved VSDN-VAE underperforms the original VSDN-SDE in the 50% case. Therefore, more observations will lead to a better nonlinear modeling and consequently higher accuracy in reconstructing the sequence. Increasing observations is often more effective than using the residual learning strategy.

Ablation Studies: We further conduct ablation studies to understand how the dimension of the latent state and the weighted coefficient β in ELBO affect the model performance. As we observe that NLL and the interpolation performance are less sensitive to these two hyperparameters, we focus on the prediction performance, which is illustrated in Figure 4. For original VSDN-VAE and VSDN-SDE, a larger β helps models to better learn KL divergence and improve the performance of the generative model. When β is small, it is hard to learn an optimal KL divergence, so the RMSE of the prediction is relatively high. Under this circumstance, a large dimension of latent states can increase the difficulty of training, which in turn aggravates the negative effects of small β. However, when β is selected appropriately, the dimension of latent states has no obvious effect on model training and prediction performance. Another thing we can see from the figure is that, the addition of residual learning helps VSDN-VAE to learn KL divergence and achieve a better prediction performance. However, as discussed earlier, VSDN-SDE has no great difficulty in optimizing the KL divergence, the residual learning strategy will make the KL divergence decrease too quickly (i.e. vanishing) to learn an optimal model in time. As a result, it reduces the model performance of VSDN-SDE. Under this condition, increasing β will aggravate the negative effects.

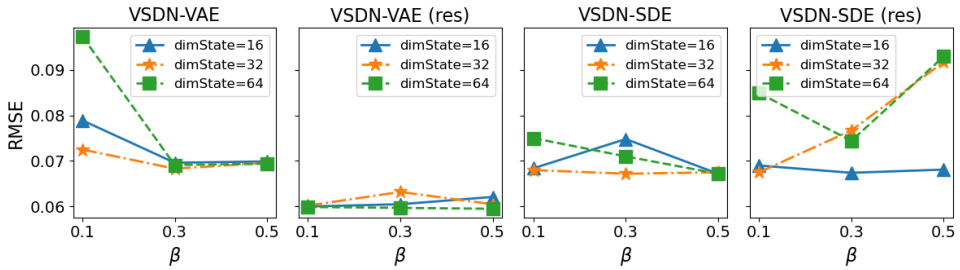


Figure 4: RMSE of prediction with respect to different values of β and dimensions of latent state.

6 Conclusion and Discussion

In this paper, we propose a deep learning framework called VSDN to learn the continuous-time stochastic dynamics from a discrete set of sequential data. We provide two variants, one is VSDN-

VAE with latent states implicitly modeled by linear stochastic differential equations, and the other is VSDN-SDE with the nonlinear modeling of latent states. We also explore the use of residual learning in our models. We demonstrate the effectiveness of VSDN through evaluations studies on both binary and continuous datasets, and our results show that VSDN can achieve better performance than ODE-RNN and GRU-ODE, two continuous-time models proposed recently. Specifically, our generative model can capture the hidden dynamics of sequences to more accurately predict the future unseen data points, while the inference model can handle the interpolation task with both forward and backward information to fill in the missing data or generate data in fine-grained time scale. In the future work, we will investigate along several potential directions: First, we will apply our models to higher dimensional and more complicated data, such as videos, which are more challenging to model yet, especially under the premise of increasing demand for producing videos in high resolution; Second, as SDEs are the base of many significant control methodologies, we will try to further extend the capacity of our models such that they can be used in precise control scenarios.

7 Broader Impact

Modeling the continuous-time stochastic process plays an important role in a large range of real-world applications. On the one hand, the stochastic modeling provides an essential knowledge summation of the collected sequence of data, and is a fundamental tool to analyze the states of the system monitored. On the other hand, many efficient controlling and decision methodologies can not be achieved without an accurate modeling of the stochastic dynamics. In this paper, we propose a deep learning framework to learn the continuous-time dynamics for many complicated scenarios. The learned models can help us understand the properties of high-dimensional sequential data, and gain the knowledge of the underlying systems. What's more, our proposed models provide new feasibility for the combination of machine learning methods and traditional control algorithms. Since many stochastic control algorithms rely on stochastic modeling of data, but there is still a lot of complicated data that is hard to model manually. Instead, our models can automatically learn an accurate stochastic process for these domains. Therefore, our methods can potentially help migrating traditional stochastic control methods into many challenging applications.

References

[1] T. Ryder, A. Golightly, A. S. McGough, and D. Prangle, "Black-box variational inference for stochastic differential equations," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 4423–4432.

[2] S. Särkkä, J. Hartikainen, I. S. Mbalawata, and H. Haario, "Posterior inference on parameters of stochastic differential equations via non-linear gaussian filtering and adaptive MCMC," *Statistics and Computing*, vol. 25, no. 2, pp. 427–437, Mar 2015.

[3] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 613–621.

[4] Y. Rubanova, T. Q. Chen, and D. K. Duvenaud, "Latent ordinary differential equations for irregularly-sampled time series," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 5321–5331.

[5] J. H. Macke, L. Buesing, J. P. Cunningham, B. M. Yu, K. V. Shenoy, and M. Sahani, "Empirical models of spiking in neural populations," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 1350–1358.

[6] B. M. Yu, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," in *Advances in Neural Information Processing Systems 21*, 2009, pp. 1881–1888.

[7] ——, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," *Journal of Neurophysiology*, vol. 102, no. 1, pp. 614–635, 2009.

[8] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 6571–6583.

[9] J. Jia and A. R. Benson, "Neural jump stochastic differential equations," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 9843–9854.

[10] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "GRU-ODE-Bayes: Continuous modeling of sporadically-observed time series," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 7379–7390.

[11] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[14] C. Archambeau, M. Opper, Y. Shen, D. Cornford, and J. S. Shawe-taylor, "Variational inference for diffusion processes," in *Advances in Neural Information Processing Systems 20*, 2008.

[15] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 3601–3610.

[16] G. E. Poliner and D. P. W. Ellis, "A discriminative model for polyphonic piano transcription," in *EURASIP Journal on Advances in Signal Processing*, 2007.

[17] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," in *PhD Thesis*, 2016.

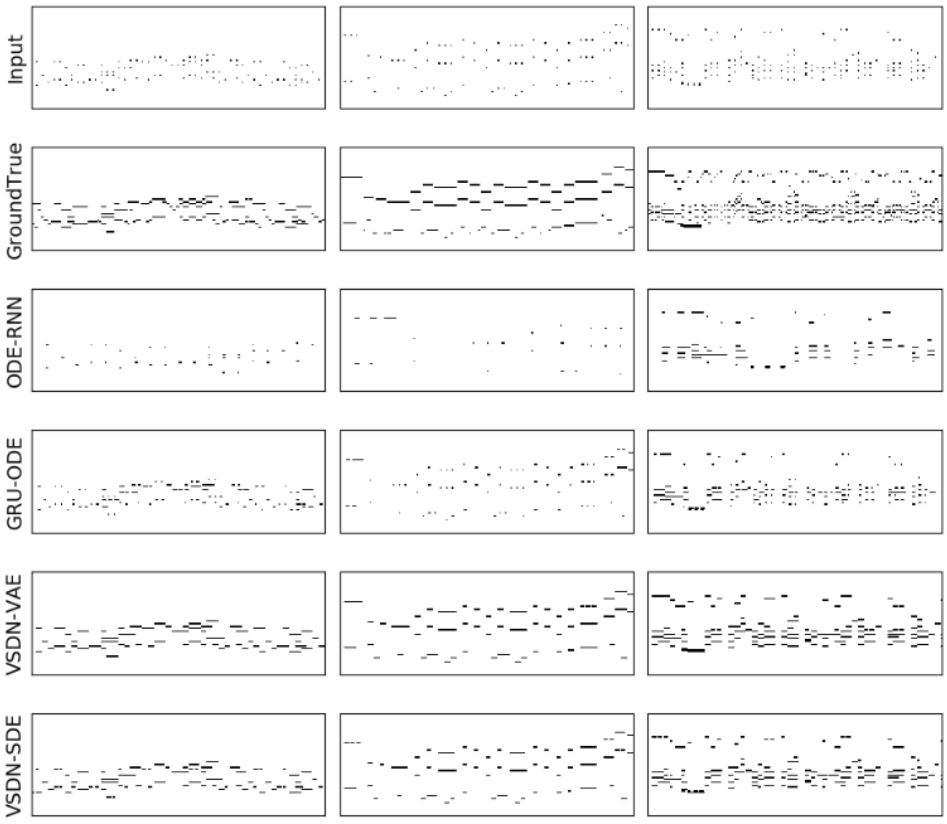
[18] Y. Liu, D. Xie, and X. Wang, "Energy-based recurrent model for stochastic modeling of music," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 236–241.

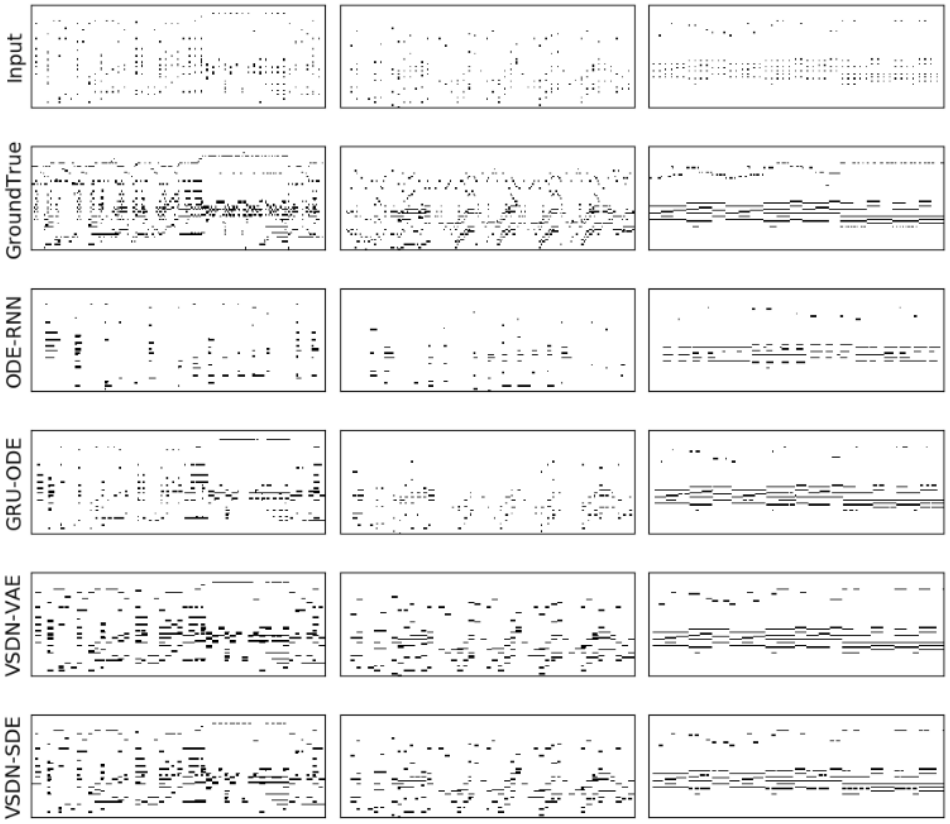
A Configuration of the Symbolic MIDI Experiments

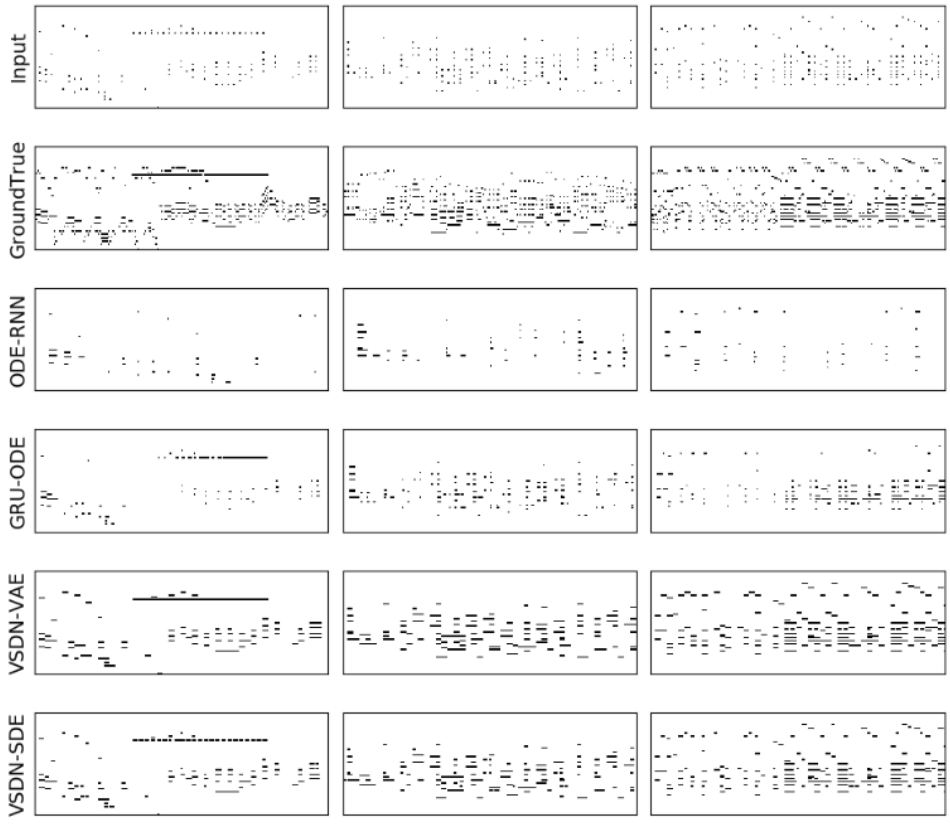
A.1 Model Configuration

For all the models, the feed-forward neural networks except the decoding network have 4 hidden layers, which have 1024 ELU units. The dimension of recurrent layers is 512 and the dimension of latent state is 128. The decoding network merely contains a single output layer. For VSDN, we set $\beta = 1$. The optimizer is Adam and the learning rate is 0.0001. The maximum training epoch is 200 and the early stopping epoch is 5.

A.2 More Interpolation Examples




Input
GroundTrue
ODE-RNN
GRU-ODE
VSDN-VAE
VSDN-SDE


Input
GroundTrue
ODE-RNN
GRU-ODE
VSDN-VAE
VSDN-SDE

B Configuration of the Speech Experiments

B.1 Model Configuration

In the TIMIT experiments, the dimension of outputs from recurrent units is 512 while the one of the latent state is 32. Except the decoders, remaining feed-forward neural networks used for all the models have 4 hidden layers, and each layer has 1024 neurons with ELU as the activation function. As for the decoder, only one output layer is contained. During the training, we choose Adam as the optimizer and the learning rate is 0.0001. The batch size is fixed as 200, and the maximum training epoch is 200 while the early stopping epoch is 10. For the loss calculation, we set $\beta = 0.1$ for our VSDN.

C Derivation of Eq. (6)

We briefly introduce the derivation of Eq. (6), which is similar to the ELBO of the discrete-time squential VAEs. By applying Jensen's inequality, we can obtain that:

$$
\begin{aligned}
&\log P_{\mathcal{G}}(y_{1:n_i}) = \log \int \prod_{i=1}^{n} P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i}) dX_{t_i} \\
&= \sum_{i=1}^{n} \log \int P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i}) dX_{t_i} \\
&= \sum_{i=1}^{n} \log \int P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t) \frac{P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})}{P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t)} dX_{t_i} \\
&\geq \sum_{i=1}^{n} \int P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t) \log \frac{P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})}{P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t)} dX_{t_i} \\
&= \sum_{i=1}^{n} \int P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t) \log \frac{P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})}{P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t)} dX_{t_i} + \int P_{\mathcal{Q}}(X_t|\mathcal{Y}_t, \mathbb{Y}_t) \log P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}) dX_{t_i} \\
&= -\sum_{i=1}^{n} KL\Big(P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i}) || P_{\mathcal{G}}(X_{t_i}|\mathcal{Y}_{t_i})\Big) + \mathbb{E}_{P_{\mathcal{Q}}(X_{t_i}|\mathcal{Y}_{t_i}, \mathbb{Y}_{t_i})} \log P_{\mathcal{G}}(y_i|X_{t_i}, \mathcal{Y}_{t_i}).
\end{aligned}
$$

In practical implementation, a weighted coefficient β will be added to the KL term to control the regularization effect of the KL divergence.

D Derivation of Eq. (9)

The detailed proof of Eq. (9) can be found in [14]. We briefly introduce here for reference.

For a SDE $dX = H(X)dt + R(X)dW$, we can discretize it by Euler-Maruyama method:

$$
X_{k+1} = X_k + H(X_k)\Delta t + R(X_k)\sqrt{\Delta t}\varepsilon, \tag{10}
$$

where $\varepsilon \sim \mathcal{N}(0, 1)$, $t_k = k\Delta t$ and Δt is the sampling interval. Eq. (10) converges to the original SDE when $\Delta t \to 0$.

The state X_{k+1} in Eq. (10) follows the conditional Gaussian distribution $P(X_{k+1}|X_k) = \mathcal{N}(X_k + H(X_k)\Delta t, \Delta t R(X_k)R(X_k)^T)$. As we assume X_0 to be constant in this paper, the joint distribution of the sate sequence $X_{1:K}$ of Eq. (10) is given by

$$
P(X_{1:K}) \propto \exp\Big(-0.5 \sum_{k=0}^{K-1} (X_{k+1} - m_k)^T \Sigma_k^{-1} (X_{k+1} - m_k)\Big), \tag{11}
$$

where $m_k = X_k + H(X_k)\Delta t$ and $\Sigma_k = \Delta t R(X_k)R(X_k)^T$.

After discretization, the KL divergence of the two SDEs in VSDN-SDE will be:

$$
\begin{aligned}
KL(P_{\mathcal{Q}}||P_{\mathcal{G}}) &= \int P_{\mathcal{Q}}(X_{1:K}) \log \frac{P_{\mathcal{Q}}(X_{1:K})}{P_{\mathcal{G}}(X_{1:K})} dX_{1:K} \\
&= \int \sum_{k=0}^{K-1} P_{\mathcal{Q}}(X_{1:K}) \log \frac{P_{\mathcal{Q}}(X_{k+1}|X_k)}{P_{\mathcal{G}}(X_{k+1}|X_k)} dX_{1:K} \\
&= \sum_{k=0}^{K-1} \int P_{\mathcal{Q}}(X_{1:K}) \log \frac{P_{\mathcal{Q}}(X_{k+1}|X_k)}{P_{\mathcal{G}}(X_{k+1}|X_k)} dX_{1:K} \\
&= \sum_{k=0}^{K-1} \int P_{\mathcal{Q}}(X_{k+2:K}|X_{k+1}) P_{\mathcal{Q}}(X_{k+1}|X_k) P_{\mathcal{Q}}(X_{1:k}) \log \frac{P_{\mathcal{Q}}(X_{k+1}|X_k)}{P_{\mathcal{G}}(X_{k+1}|X_k)} dX_{1:K} \\
&= \sum_{k=0}^{K-1} \int P_{\mathcal{Q}}(X_{k+1}|X_k) P_{\mathcal{Q}}(X_k) \log \frac{P_{\mathcal{Q}}(X_{k+1}|X_k)}{P_{\mathcal{G}}(X_{k+1}|X_k)} dX_k dX_{k+1} \\
&= \sum_{k=0}^{K-1} \int P_{\mathcal{Q}}(X_k) \cdot KL\Big(P_{\mathcal{Q}}(X_{k+1}|X_k)||P_{\mathcal{G}}(X_{k+1}|X_k)\Big) dX_k \\
&= \sum_{k=0}^{K-1} \mathbb{E}_{X_k \sim P_{\mathcal{Q}}(X_k)} KL\Big(P_{\mathcal{Q}}(X_{k+1}|X_k)||P_{\mathcal{G}}(X_{k+1}|X_k)\Big),
\end{aligned}
$$

where $P_{\mathcal{Q}}(X_k)$ is the marginal distribution of X_k in the inference SDE. According to the KL divergence of two Gaussian distribution, we further have

$$
\begin{aligned}
&KL\Big(P_{\mathcal{Q}}(X_{k+1}|X_k)||P_{\mathcal{G}}(X_{k+1}|X_k)\Big) = \frac{1}{2}\Big(tr(\Sigma_{k,\mathcal{G}}^{-1}\Sigma_{k,\mathcal{Q}}) + (m_{k,\mathcal{G}} - m_{k,\mathcal{Q}})^T \Sigma_{k,\mathcal{G}}^{-1}(m_{k,\mathcal{G}} - m_{k,\mathcal{Q}}) \\
&\quad + \log \frac{\det \Sigma_{k,\mathcal{G}}}{\det \Sigma_{k,\mathcal{Q}}} - d\Big) \\
&= \frac{1}{2}\Big(tr\Big((R_{\mathcal{G}}R_{\mathcal{G}}^T)^{-1} R_{\mathcal{Q}} R_{\mathcal{Q}}^T\Big) + \Delta t (H_{\mathcal{G}} - H_{\mathcal{Q}})^T (R_{\mathcal{G}} R_{\mathcal{G}}^T)^{-1} (H_{\mathcal{G}} - H_{\mathcal{Q}}) + \log \frac{R_{\mathcal{G}} R_{\mathcal{G}}^T}{R_{\mathcal{Q}} R_{\mathcal{Q}}^T} - d\Big)
\end{aligned}
$$

where d is the dimension of X_{k+1}. When we restrict $R_{\mathcal{G}} = R_{\mathcal{Q}}$, we have

$$
KL(P_{\mathcal{Q}}||P_{\mathcal{G}}) = \frac{1}{2} \sum_{k=0}^{K-1} \mathbb{E}_{X_k \sim P_{\mathcal{Q}}(X_k)} (H_{\mathcal{G}} - H_{\mathcal{Q}})^T (R_{\mathcal{G}} R_{\mathcal{G}}^T)^{-1} (H_{\mathcal{G}} - H_{\mathcal{Q}}) \Delta t.
$$

When we set $\Delta t \to 0$, the discretized SDEs converge to the original SDEs and $KL(P_{\mathcal{Q}}||P_{\mathcal{G}})$ converges to Eq. (9).

If $R_{\mathcal{G}}$ does not equal to $R_{\mathcal{Q}}$, we have

$$
\begin{aligned}
KL(P_{\mathcal{Q}}||P_{\mathcal{G}}) &= \frac{1}{2} \lim_{\Delta t \to 0} \sum_{k=0}^{K-1} \mathbb{E}_{X_k \sim P_{\mathcal{Q}}(X_k)} \Big((H_{\mathcal{G}} - H_{\mathcal{Q}})^T (R_{\mathcal{G}} R_{\mathcal{G}}^T)^{-1} (H_{\mathcal{G}} - H_{\mathcal{Q}}) + \frac{const}{\Delta_t}\Big) \Delta t, \\
&= +\infty
\end{aligned}
$$