APSIPA Transactions on Signal and Information Processing, 2022, 11, e34

Original Paper

Bayesian Multi-Temporal-Difference Learning

Jen-Tzung Chien* and Yi-Chung Chiu

Institute of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, Hsinchu, Taiwan

ABSTRACT

This paper presents a new sequential learning via a planning strategy where the future samples are predicted by reflecting the past experiences. Such a strategy is appealing to implement an intelligent machine which foresees multiple time steps instead of predicting step by step. In particular, a flexible sequential learning is developed to directly predict future states without visiting all intermediate states. A Bayesian approach to multi-temporal-difference neural network is accordingly proposed to calculate the stochastic belief state for an abstract state machine so as to capture large-span context as well as make high-level prediction. Importantly, the sequence data are represented by *multiple jumpy states* with varying temporal differences. A Bayesian state machine is trained by maximizing the variational lower bound of log likelihood of sequence data. A generalized sequence model with various number of Markov states is derived with the simplified realization to the previous temporal-difference variational autoencoder. The predictive states are learned to roll forward with jumps. Experiments show that this approach is substantially trained to predict jumpy states in various types of sequence data.

Keywords: Bayesian learning, variational autoencoder, sequential learning, temporal-difference learning, state machine.

*Corresponding author: Jen-Tzung Chien, jtchien@nycu.edu.tw.

Received 11 June 2022; Revised 11 September 2022
ISSN 2048-7703; DOI 10.1561/116.00000037

1 Introduction

Machine learning is generally categorized into supervised learning, unsupervised learning and reinforcement learning. Traditional machine learning works extensively on supervised learning while modern learning strategies involve a lot of unsupervised learning and reinforcement learning (RL) [13] due to the numerous applications in the real world. Merging the concept of RL into unsupervised learning is a new research trend. A meaningful learning approach can be implemented by introducing a simulator for the world which is learned in an unsupervised manner. Typically, unsupervised learning based on the generative model is to approximate the data distribution by the simulator for the environment that generates or mimics a class of samples which are close to the observation data. Generative models are feasible to build various applications including image generation [20], text generation [14], acoustic modeling [17], data augmentation and environment simulation [23], to name a few. This study adopts the aspect of RL and develops a new type of generative model for multi-step planning in a framework of Bayesian sequential learning.

1.1 Related Work

Generative model can be trained to generate observation samples or latent variables which are feasible to implement individual or combined learning strategies for real-world applications. The emerging approaches to deep generative models, for example, variational autoencoder (VAE), generative adversarial network [16] and autoregressive neural network [39], have been successfully developed according to different learning aspects. One of the most important solutions is based on the *latent variable model* which is constructed as a probabilistic generative model. This model is driven by a top-down generation from latent variables to observations through Bayesian learning based on the prior and posterior distributions [5]. VAE is recognized as a latent variable model consisting of an inference model or encoder and a generative model or decoder. The encoder compresses high-dimensional input sample $\mathbf{x}_n$ into a low-dimensional random variable $\mathbf{z}_n$ while the decoder generates the sample $\widehat{\mathbf{x}}_n$ from random samples of $\mathbf{z}_n$. The encoder and decoder are jointly learned by maximizing the variational lower bound of log likelihood of observations $\mathbf{x} = \{\mathbf{x}_n\}_{n=1}^N$. However, VAE could not directly represent the temporal information in sequence data $\mathbf{x} = \{\mathbf{x}_t\}_{t=1}^T$. In [21, 23], the autoregressive neural network with sequential latent variables $\{\mathbf{z}_t\}_{t=1}^T$ was constructed and trained to predict *long-term future*.

In [11, 12], the stochastic recurrent neural network was proposed to explore the randomness in structured data for sequential learning. In [1], the stochastic temporal convolutional network was presented to implement the Bayesian learning for sequence data. In [17], the variational bidirectional recurrent network was developed as a latent variable model. This model employed

the Bayesian treatment based on the Z-forcing which combined the latent codes $\mathbf{z}_t$ from forward and backward paths and fused the information of history and future at each time step t. Future information was adopted as a regularizer [35]. In [44], VAE was incorporated in a neural ordinary differential equations for sequential learning where the continuous-time state dynamics [8] were represented for future prediction. In [28], the variational RL based on the Stein variational gradient descent algorithm was proposed to implement Bayesian inference for maximum entropy policy optimization which balanced the tradeoff between exploration and exploitation in RL.

In [33], a self-consistent trajectory autoencoder was implemented for hierarchical RL based on the trajectory-level generative model. A state decoder and a policy decoder were trained to generate consistent trajectories as a constraint in variational inference. Nevertheless, in real circumstances, humans think and plan across multiple jumpy time steps instead of acting in a step-by-step manner. In [18], the pairs of temporally separated time samples were represented. The variational lower bound was optimized to fulfill *temporal-difference learning* [2, 27] which was popularly employed in a model-based RL. Such a learning algorithm was used to build a generative model to predict adjacent latent states which conveyed the pairwise information.

1.2 Main Idea of this Work

This study presents a new sequential learning machine where the aspects of supervised learning, unsupervised learning and reinforcement learning are integrated in a probabilistic generative model. A *stochastic temporal-difference learning* [9] is proposed to simulate the experience of a RL agent which holds three properties. First, the *latent states* $\{\mathbf{z}_t\}$ are inferred from input observations $\{\mathbf{x}_t\}$ and then employed to make predictions in latent variable space rather than observation space. These latent states contain temporal information from the association patterns [4] at *multiple jumpy steps* $\{t_1, \ldots, t_K\}$ in random distance which are generalized from the association information in multiple latent states $\{\mathbf{z}_{t_k}\}_{k=1}^K$ at the corresponding time steps $\{t_k\}_{k=1}^K$. Second, these random states are inferred from a sequence of *deterministic belief states* $\{\mathbf{b}_t\}_{t=1}^T$, which are updated by using the observations to represent the features in environment or low-level information in the world. The belief states act as the prior information to encourage state transition and penalize state smoothing. A hierarchical state machine is constructed to fulfill a new temporal-difference learning. Both the deterministic and stochastic states in this sophisticated latent variable model are learned according to a variational lower bound of log likelihood which is optimized to allow multi-step prediction [32].

Finally, this model learns a *temporal abstraction* [24] which is feasible to predict with jumpy distant states. We develop a Bayesian multi-temporal-difference neural network consisting of four network modules including belief network, inference network, transition network and generation network. The whole

framework is learned from input observations $\mathbf{x} = \{\mathbf{x}_t\}$ in an unsupervised manner. Such an unsupervised model can be extended as a supervised network by connecting hidden states $\mathbf{z} = \{\mathbf{z}_t\}$ to output targets $\mathbf{y} = \{\mathbf{y}_t\}$ to build classification model. Practically, Markov constraint and padding scheme are employed to reduce the model size as well as stabilize the training procedure with the increased number of jumpy states. Experiments are conducted to illustrate how this framework works for the prediction of moving objects and directions as well as preservation of continuing scenario in game environment [3]. Language modeling for word prediction and sentiment classification are investigated.

2 Bayesian Sequential Learning

Variational autoencoder (VAE) acts as the theoretical foundation to build various Bayesian recurrent networks [6, 7].

2.1 Variational Autoencoder

VAE [26, 38, 41] was proposed to estimate the distribution of latent variable $\mathbf{z}$. This distribution is adopted to generate random samples for stochastic reconstruction of input data $\mathbf{x}$. VAE is seen as a generative model which can synthesize new samples to simulate the statistical behavior of latent variable in neural network. VAE consists of an inference model as encoder and a generative model as decoder. The encoder in a form of variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ with parameter ϕ and the decoder with a generative distribution $p_\theta(\mathbf{x}|\mathbf{z})$ using parameter θ are jointly learned by maximizing the evidence lower bound (ELBO) $\mathcal{L}(\mathbf{x}; \theta, \phi)$ which is obtained by

$$\begin{aligned}
\log p(\mathbf{x}) &= \log \int q_\phi(\mathbf{z}|\mathbf{x}) \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} d\mathbf{z} = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right] \\
&\geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right) \right] \\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] - \mathcal{D}_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})) \\
&\triangleq \mathcal{L}(\mathbf{x}; \theta, \phi).
\end{aligned} \tag{1}$$

The first term in ELBO reflects the log likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ or the negative reconstruction error due to decoder by using those samples $\mathbf{z} \in \mathbb{R}^M$ from encoder $q_\phi(\mathbf{z}|\mathbf{x})$. The second term is a Kullback-Leibler (KL) divergence which regularizes the variational posterior of encoder $q_\phi(\mathbf{z}|\mathbf{x})$ to get close to a standard Gaussian prior $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ where $\mathbf{I}$ is an identity matrix. A re-parameterization trick [26] was applied to compute the stochastic gradients

in backpropagation algorithm with three steps. First, a standard Gaussian distribution is sampled to find $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ where l is the sample index. Second, the posterior sample is drawn by

$$\mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_z, \text{diag}(\boldsymbol{\sigma}_z^2)), \tag{2}$$

using $\mathbf{z}^{(l)} = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z \odot \boldsymbol{\epsilon}^{(l)}$ where Gaussian parameters are obtained by an encoder or inference network $[\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2] = [\{\mu_{zi}\}, \{\sigma_{zi}^2\}] = f_\phi^{(q)}(\mathbf{x})$. Third, the stochastic gradients are calculated over ELBO $\mathcal{L}(\mathbf{x}; \theta, \phi)$ by using a set of latent variables $\{\mathbf{z}^{(l)}\}_{l=1}^L$

$$\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}^{(l)}) + \frac{1}{2}\sum_{i=1}^{M}\left(1 + \log(\sigma_{zi}^2) - \mu_{zi}^2 - \sigma_{zi}^2\right). \tag{3}$$

Parameters $\{\theta, \phi\}$ are updated by a stochastic backpropagation according to autoencoding variational Bayesian [26]. VAE represents a bag of N samples $\mathbf{x} = \{\mathbf{x}_n\}$ where temporal dependency in individual samples $\mathbf{x}_n$ is disregarded.

2.2 Variational Recurrent Neural Network

VAE aims to characterize a set of random inputs $\mathbf{x} = \{\mathbf{x}_n\}$ based on latent variable model where a probabilistic model is trained via variational inference. Model uncertainty is represented to improve the prediction performance [42, 43]. Considering a set of T sequence samples $\mathbf{x} = \{\mathbf{x}_t\}_{t=1}^T$ and latent variables $\mathbf{z} = \{\mathbf{z}_t\}_{t=1}^T$, it is crucial to develop the variational recurrent neural network (VRNN) [12] for Bayesian sequential learning by maximizing the likelihood

$$\begin{aligned} p(\mathbf{x}) &= \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{<t}) = \prod_{t=1}^{T}\int_{\mathbf{z}_t} p_\theta(\mathbf{x}_t, \mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) d\mathbf{z}_t \\ &= \prod_{t=1}^{T}\int_{\mathbf{z}_t} p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\le t}) p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) d\mathbf{z}_t. \end{aligned} \tag{4}$$

$\mathbf{x}_{<t} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}\}$ denotes the history of current time sample $\mathbf{x}_t$ in a causal system. The random latent variable $\mathbf{z}_t$ in VRNN is comparable with the deterministic hidden state $\mathbf{h}_t$ in standard recurrent neural network (RNN). State $\mathbf{h}_t$ is continuously updated by using current sample $\mathbf{x}_t$ and previous state $\mathbf{h}_{t-1}$ which depends on the histories of data $\mathbf{x}_{<t}$ and states $\mathbf{h}_{<t}$. VRNN is superior to RNN because of uncertainty modeling which is crucial in adverse condition with heterogeneous data. From Equations (1) and (4), VRNN is derived by maximizing the ELBO due to sequence data

$$\begin{aligned} \log p(\mathbf{x}) \ge \sum_{t=1}^{T} &\mathbb{E}_{q_\phi(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})}[\log p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\le t})] \\ &- \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \| p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})), \end{aligned} \tag{5}$$

where KL divergence between variational posterior $q_\phi(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ and history posterior $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ is measured with parameters ϕ and θ, respectively. VRNN is seen as an extended VAE for sequential learning. VRNN runs VAE at each time t by using causal data $\mathbf{x}_{\leq t} = \{\mathbf{x}_{<t}, \mathbf{x}_t\}$ based on the stochastic latent codes $\mathbf{z}_{\leq t}$ with condition on the hidden state $\mathbf{h}_{t-1}$ of RNN at previous time $t-1$. Figure 1 depicts the graphical representation of VRNN at neighboring time steps $t-1$ and t. The generated sample is denoted by $\widehat{\mathbf{x}}_t$. VRNN consists of an inference network and a generation or output network at each time t, which are used to calculate the Gaussian distributions $q_\phi(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = \mathcal{N}(\boldsymbol{\mu}_{z,t}, \mathrm{diag}(\boldsymbol{\sigma}^2_{z,t}))$ and $p_\theta(\mathbf{x}_t|\mathbf{x}_{<t}, \mathbf{z}_{\leq t}) = \mathcal{N}(\boldsymbol{\mu}_{x,t}, \mathrm{diag}(\boldsymbol{\sigma}^2_{x,t}))$ with parameters $[\boldsymbol{\mu}_{z,t}, \boldsymbol{\sigma}^2_{z,t}] = f^{(q)}_\phi(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$ and $[\boldsymbol{\mu}_{x,t}, \boldsymbol{\sigma}^2_{x,t}] = f^{(o)}_\theta(\mathbf{z}_t, \mathbf{h}_{t-1})$, respectively. A recurrent network $\mathbf{h}_t = f^{(r)}_\theta(\mathbf{x}_t, \mathbf{z}_t, \mathbf{h}_{t-1})$ and a prior network $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) = \mathcal{N}(\boldsymbol{\mu}_{p,t}, \mathrm{diag}(\boldsymbol{\sigma}^2_{p,t}))$ are also merged with $[\boldsymbol{\mu}_{p,t}, \boldsymbol{\sigma}^2_{p,t}] = f^{(p)}_\theta(\mathbf{h}_{t-1})$. The inference network $f^{(q)}_\phi$, prior network $f^{(p)}_\theta$, recurrent network $f^{(r)}_\theta$ and generation network $f^{(o)}_\theta$ are jointly trained by maximizing the ELBO [17].

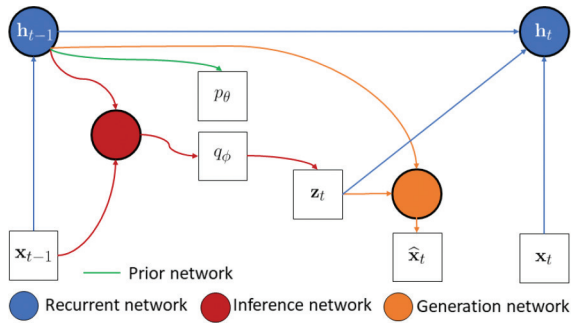


Figure 1: Graphical model for variational recurrent neural network consisting of recurrent network, prior network, inference network and generation (or decoder) network. Here, p_θ denotes the posterior $p_\theta(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ which is approximated by variational posterior $q_\phi(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$.

3 Bayesian Temporal-Difference Learning

Bayesian sequential learning can be extended by considering the aspect of temporal-difference learning from reinforcement learning based on the belief state representation.

3.1 Belief State-Space Model

A simple way to sequential learning is based on an autoregressive model using Equation (4). RNN is able to aggregate the temporal information from the

past $\mathbf{x}_{<t}$ based on a recurrent machine where the hidden state $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$ is continuously updated at each time t. The previous hidden state $\mathbf{h}_{t-1}$ and the current observation $\mathbf{x}_t$ are used as inputs to RNN. One potential weakness using the autoregressive model is that the learning process is only run over the observation space without compressed representation. This process becomes challenging in presence of high-dimensional data. An alternative solution is to learn a high-level abstraction for state transitions in sequence data based on a latent variable model. The resulting state-space model is more computationally efficient than the autoregressive model. VRNN builds a latent variable model, but the high-level abstraction is missing. To tackle this weakness, the online belief state $\mathbf{b}_t$ [18] was merged to characterize the conditional distribution for the future given the past

$$\begin{aligned} p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | \mathbf{x}_1, \ldots, \mathbf{x}_t) &\approx p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | \mathbf{b}_t) \\ &= \int p(\mathbf{x}_{t+1}, \ldots, \mathbf{x}_T | \mathbf{z}_t) p(\mathbf{z}_t | \mathbf{b}_t) d\mathbf{z}_t. \end{aligned} \tag{6}$$

$\mathbf{b}_t$ contains all the information about the state in an environment where the agent interacts with. A neural belief state is updated by $\mathbf{b}_t = f(\mathbf{x}_t, \mathbf{b}_{t-1})$ using RNN. From the perspective of RL, the agent usually handles a partially observed environment which is modeled by a partially observed Markov decision process. Representing such a process is comparable of finding an optimal behavior policy that maps the agent's available knowledge of environment, i.e. its belief state, to the corresponding actions [10, 34]. Therefore, the hidden state $\mathbf{h}_t$ is treated as sufficient statistics $\mathbf{b}_t = \mathbf{b}_t(\mathbf{x}_1, \ldots, \mathbf{x}_t)$ to formulate the belief state. Considering this abstraction information, the belief-state-based evidence lower bound of conditional log likelihood $\log p(\mathbf{x}_t | \mathbf{x}_{<t})$ is derived by referring Equations (1) and (5) so as to obtain

$$\begin{aligned} &\mathbb{E}_{(\mathbf{z}_{t-1}, \mathbf{z}_t) \sim q(\mathbf{z}_{t-1}, \mathbf{z}_t | \mathbf{x}_{\leq t})} \\ &\times \left[\log \underbrace{p(\mathbf{x}_t | \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{x}_{<t})}_{=p(\mathbf{x}_t | \mathbf{z}_t)} \underbrace{p(\mathbf{z}_{t-1}, \mathbf{z}_t | \mathbf{x}_{<t})}_{=p(\mathbf{z}_t | \mathbf{z}_{t-1}) p(\mathbf{z}_{t-1} | \mathbf{x}_{<t})} - \log \underbrace{q(\mathbf{z}_{t-1}, \mathbf{z}_t | \mathbf{x}_{\leq t})}_{=q(\mathbf{z}_{t-1} | \mathbf{z}_t, \mathbf{x}_{\leq t}) q(\mathbf{z}_t | \mathbf{x}_{\leq t})} \right] \end{aligned} \tag{7}$$

which is calculated at each time t and summed up to accumulate ELBO for log likelihood of all observations $\log p(\mathbf{x}) = \sum_t \log p(\mathbf{x}_t | \mathbf{x}_{<t})$. Notably, this ELBO is obtained by inferring over two neighboring latent states $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ in a RL manner by applying the probability chain rule. In particular, $q(\mathbf{z}_{t-1}, \mathbf{z}_t | \mathbf{x}_{\leq t})$ is decomposed as a belief over $\mathbf{z}_t$ and a one-step smoothing distribution over

$\mathbf{z}_{t-1}$ based on future sample $\mathbf{z}_t$. A direct representation of future states is performed. Equation (7) is finally rewritten as

$$\begin{aligned}\mathcal{L}_{t-1,t} = \mathbb{E}_{(\mathbf{z}_{t-1},\mathbf{z}_t)\sim q(\mathbf{z}_{t-1},\mathbf{z}_t|\mathbf{b}_{t-1},\mathbf{b}_t)}\,[&\log p(\mathbf{x}_t|\mathbf{z}_t)\\ &+\log p(\mathbf{z}_t|\mathbf{z}_{t-1})+\log p_B(\mathbf{z}_{t-1}|\mathbf{b}_{t-1})\\ &-\log q(\mathbf{z}_{t-1}|\mathbf{z}_t,\mathbf{b}_{t-1},\mathbf{b}_t)-\log p_B(\mathbf{z}_t|\mathbf{b}_t)]\,.\end{aligned} \tag{8}$$

Notably, the *prior belief probabilities* $p_B(\mathbf{z}_{t-1}|\mathbf{b}_{t-1})$ and $p_B(\mathbf{z}_t|\mathbf{b}_t)$ are defined for $p(\mathbf{z}_{t-1}|\mathbf{x}_{<t})$ and $q(\mathbf{z}_t|\mathbf{x}_{\leq t})$ where the histories $\mathbf{x}_{<t}$ and $\mathbf{x}_{\leq t}$ in the conditions are expressed by belief states $\mathbf{b}_{t-1}$ and $\mathbf{b}_t$, respectively.

3.2 Temporal-difference Variational Autoencoder

In [18], the temporal-difference variational autoencoder (TD-VAE) was proposed to extend the sequential learning with two jumpy states. Using TD-VAE, the latent states were calculated from sequence data by aggregating two time samples t_1 and t_2 separated by a *random interval* rather than integrating two neighboring samples in consecutive times $t-1$ and t. The latent states in TD-VAE characterize the future explicitly. In the optimization, the objective in Equation (8) is modified to construct the ELBO $\mathcal{L}_{t_1,t_2}$ by using the time pairs $\{t_1,t_2\}$ with temporally separated samples $\{\mathbf{x}_{t_1},\mathbf{x}_{t_2}\}$. The lower bound $\mathcal{L}_{t_1,t_2}$ from t_1 to t_2 is then expressed by

$$\begin{aligned}\mathcal{L}_{t_1,t_2} = \mathbb{E}_{\mathbf{z}_{t_2}\sim q(\mathbf{z}_{t_2}|\mathbf{b}_{t_2})}\left[\log p_D^{t_2}(\mathbf{x}_{t_2})\right]\\ +\mathcal{D}_{\mathrm{KL}}(p_T^{t_2}(\mathbf{z}_{t_2})\|p_B^{t_2}(\mathbf{z}_{t_2}))-\mathcal{D}_{\mathrm{KL}}(q_S^{t_1|t_2}\|p_B^{t_1})\end{aligned} \tag{9}$$

where p_D denotes the decoder distribution, p_T denotes the transition distribution and q_S denotes the smoothing distribution. Figure 2 shows the graphical model for TD-VAE with two jumpy states in randomly-separated times t_1 and t_2. TD-VAE is composed of belief network, inference network, transition network and generation network. Belief network is applied to find the prior belief probability $p_B(\mathbf{z}_t|\mathbf{b}_t)$ or p_B^t, similar to the recurrent and prior networks in VRNN, for belief states $\mathbf{b}_t$ at each time $t=t_1$ or t_2 which are updated by an RNN. Inference network is used to infer the variational distribution $q(\mathbf{z}_{t_1}|\mathbf{z}_{t_2},\mathbf{b}_{t_1},\mathbf{b}_{t_2})$ or $q_S^{t_1|t_2}$ as a smoothing factor for t_1 using future t_2. Transition network is introduced to calculate the transition likelihood $p(\mathbf{z}_{t_2}|\mathbf{z}_{t_1})$ or $p_T^{t_2}$ from t_1 to t_2. Generation network is estimated as a decoder network to determine the likelihood $p(\mathbf{x}_{t_2}|\mathbf{z}_{t_2})$ or $p_D^{t_2}$ to generate the sample $\widehat{\mathbf{x}}_{t_2}$ at time t_2. Belief distribution at t_2 acts as a penalty in the ELBO of TD-VAE while belief distribution at t_1 acts as a prior to regularize the smoothing function $q_S^{t_1|t_2}$ for inferring sample $\mathbf{z}_{t_1}$ from $\mathbf{z}_{t_2}$.

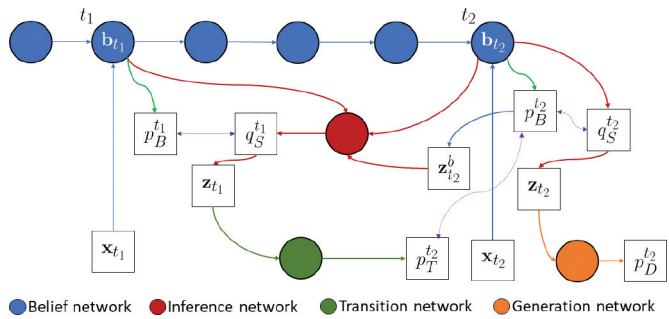


Figure 2: Graphical model for temporal-difference variational autoencoder consisting of belief network, inference network, transition network and generation network.

4 Bayesian Multi-temporal-difference Learning

TD-VAE carries out the Bayesian sequential learning where the aspect of reinforcement learning based on temporal-difference (TD) learning [37] is implemented. The temporal difference from the pairs of states $\mathbf{z}_{t_1}$ and $\mathbf{z}_{t_2}$ in two randomly-separated times t_1 and t_2 is characterized. Such a RL method is constrained since only two distant states and actions are modeled. To relax this constraint, TD learning is extended to characterize a *trajectory* with more than two states and actions. There are twofold novelties in comparison with RL algorithm based on the pairwise TD learning. First, this study focuses on sequential learning without the action inputs. A new type of recurrent state machine is proposed. Second, the previous TD learning is deterministic without involving latent variable model. This paper presents a new Bayesian learning, which is called the *stochastic temporal-difference learning*. In particular, this study develops the stochastic temporal-difference neural network (STDNN) which is recognized as a generalization of TD-VAE for multi-temporal-difference learning. This *multi-temporal-difference learning* characterizes the regularity in a set of K jumpy time samples or association patterns $\{\mathbf{x}_{t_1}, \ldots, \mathbf{x}_{t_K}\}$ [4] from a collection of sequence data. Typically, t_1 is chosen randomly. $t_k - t_{k-1}$ is selected from a uniform distribution in a period between 1 and D where D is a bound of planning window.

4.1 Stochastic Temporal-difference Neural Network

Bayesian sequential learning is accordingly developed to represent the sequential patterns in multiple time samples with random temporal differences. Again, we extend from the sequence modeling with K latent states $\{\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t\}$ in consecutive times to that with K distant latent states $\{\mathbf{z}_{t_1}, \ldots, \mathbf{z}_{t_K}\}$ where t_1 and t_K denote the beginning and ending times in K separated time stamps

along the sequence data $\mathbf{x} = \{\mathbf{x}_t\}_{t=1}^{T}$, respectively. Number of samples K is general. The belief state-space representation based on TD-VAE in Sections 3.1 and 3.2 is seen as a special realization of the proposed STDNN with $K = 2$. By adopting three latent states $\mathbf{z}_{t-2}$, $\mathbf{z}_{t-1}$ and $\mathbf{z}_t$ in STDNN, the ELBO of the conditional log likelihood at a given t based on the belief state representation, denoted by $\mathcal{L}_{t-2,t-1,t}$, is manipulated by

$$\begin{aligned}
&\log p(\mathbf{x}_t|\mathbf{x}_{<t}) \\
&= \log \int \left(\frac{p(\mathbf{x}_t|\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{x}_{<t}) p(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{<t})}{q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t})} \right) \\
&\quad \times q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t}) d\mathbf{z}_{t-2} d\mathbf{z}_{t-1} d\mathbf{z}_t \\
&\ge \int \log \left(\frac{p(\mathbf{x}_t|\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{x}_{<t}) p(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{<t})}{q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t})} \right) \\
&\quad \times q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t}) d\mathbf{z}_{t-2} d\mathbf{z}_{t-1} d\mathbf{z}_t \\
&= \mathbb{E}_{(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t) \sim q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t})} \left[\log p(\mathbf{x}_t|\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{x}_{<t}) \right. \\
&\quad \left. + \log p(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{<t}) - \log q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{x}_{\le t}) \right] \\
&= \mathbb{E}_{(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t) \sim q(\mathbf{z}_{t-2}, \mathbf{z}_{t-1}, \mathbf{z}_t|\mathbf{b}_{t-2}, \mathbf{b}_{t-1}, \mathbf{b}_t)} \left[\log p(\mathbf{x}_t|\mathbf{z}_t) \right. \\
&\quad + \log p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{z}_{t-2}) + \log p(\mathbf{z}_{t-1}|\mathbf{z}_{t-2}) \\
&\quad + \log p_B(\mathbf{z}_{t-2}|\mathbf{b}_{t-2}) - \log q(\mathbf{z}_{t-2}|\mathbf{z}_{t-1}, \mathbf{z}_t, \mathbf{b}_{t-1}, \mathbf{b}_t) \\
&\quad \left. - \log q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{b}_{t-1}, \mathbf{b}_t) - \log p_B(\mathbf{z}_t|\mathbf{b}_t) \right] \triangleq \mathcal{L}_{t-2,t-1,t}.
\end{aligned} \tag{10}$$

which is an extension of Equation (8) from $K = 2$ to $K = 3$. Considering K latent variables $\{\mathbf{z}_{t-k}\}_{k=0}^{K-1}$, the ELBO $\mathcal{L}_{t-(K-1),\ldots,t}$ of $\log p(\mathbf{x}_t|\mathbf{x}_{<t})$ can be extended in a form of

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t} \left[\log p(\mathbf{x}_t|\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t, \mathbf{x}_{<t}) + \right. \\
&\quad \left. \log p(\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t|\mathbf{x}_{<t}) - \log q(\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t|\mathbf{x}_{\le t}) \right].
\end{aligned} \tag{11}$$

Similar to Equation (7), we consider the properties

$$\begin{aligned}
&p(\mathbf{x}_t|\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t, \mathbf{x}_{<t}) = p(\mathbf{x}_t|\mathbf{z}_t) \\
&p(\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t|\mathbf{x}_{<t}) = p(\mathbf{z}_t|\mathbf{z}_{t-1}, \ldots, \mathbf{z}_{t-(K-1)}) \\
&\quad \times p(\mathbf{z}_{t-1}|\mathbf{z}_{t-2}, \ldots, \mathbf{z}_{t-(K-1)}) \cdots p(\mathbf{z}_{t-(K-1)}|\mathbf{x}_{<t}) \\
&q(\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_t|\mathbf{x}_{\le t}) = q(\mathbf{z}_{t-(K-1)}|\mathbf{z}_{t-(K-2)}, \ldots, \mathbf{z}_t, \mathbf{x}_{\le t}) \\
&\quad \times q(\mathbf{z}_{t-(K-2)}|\mathbf{z}_{t-(K-3)}, \ldots, \mathbf{z}_t, \mathbf{x}_{\le t}) \cdots q(\mathbf{z}_t|\mathbf{x}_{\le t}).
\end{aligned} \tag{12}$$

By following the *belief state-space model*, the same expression $p_B(\mathbf{z}|\mathbf{b})$ with belief state $\mathbf{b}$ is used to approximate the belief distributions for $p(\mathbf{z}_{t-(K-1)}|\mathbf{x}_{<t})$ and $q(\mathbf{z}_t|\mathbf{x}_{\le t})$, which are characterized by the belief states at times $t - (K-1)$ and t, respectively. Meanwhile, the variational posteriors over

$\{\mathbf{z}_{t-(K-1)}, \ldots, \mathbf{z}_{t-1}\}$ can be calculated via belief states in smoothing distributions $q(\mathbf{z}_{t-(K-1)}|\mathbf{z}_{t-(K-2)}, \ldots, \mathbf{z}_t, \mathbf{b}_{t-(K-1)}, \ldots, \mathbf{b}_t)$ until $q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{b}_{t-1}, \mathbf{b}_t)$. The ELBO of STDNN in Equation (11) with K time steps under these properties is therefore rewritten as

$$\begin{aligned}\log p(\mathbf{x}_t|\mathbf{x}_{<t}) \geq \mathbb{E}_{\mathbf{z}_{t-(K-1)},\ldots,\mathbf{z}_t}\big[&\log p(\mathbf{x}_t|\mathbf{z}_t) \\ &+ \log p(\mathbf{z}_t|\mathbf{z}_{t-1}, \ldots, \mathbf{z}_{t-(K-1)}) \\ &+ \cdots + \log p_B(\mathbf{z}_{t-(K-1)}|\mathbf{b}_{t-(K-1)}) \\ &- \log q(\mathbf{z}_{t-(K-1)}|\mathbf{z}_{t-(K-2)}, \ldots, \mathbf{b}_t) - \cdots - \log p_B(\mathbf{z}_t|\mathbf{b}_t)\big] .\end{aligned} \tag{13}$$

Instead of predicting the latent states $\{\mathbf{z}_t\}$ at each time t in VRNN and the paired states $\{\mathbf{z}_{t_1}, \mathbf{z}_{t_2}\}$ at the connected times $\{t_1, t_2\}$ in TD-VAE, the proposed STDNN aims to learn the multiple latent states $\{\mathbf{z}_{t-k}\}_{k=0}^{K-1}$ from sequence data $\{\mathbf{x}_t\}_{t=1}^{T}$ based on the belief state representation. In addition to capture the neighboring information from $t-(k-1)$ to t, $1 \leq k \leq K$, an alternative realization of STDNN is designed to learn *time abstraction* by associating K *remote* times through their corresponding states $\{\mathbf{z}_{t_1}, \ldots, \mathbf{z}_{t_K}\}$. Following the multi-temporal-difference learning, the ELBO using STDNN with K jumpy states or association patterns is derived by generalizing $\mathcal{L}_{t_1,t_2}$ in Equation (9) for TD-VAE to $\mathcal{L}_{t_1,\ldots,t_K}$ in Equation (17) which is detailed in the Appendix.

4.2 Generalization and Interpretation

We develop a general solution to STDNN for multi-temporal-difference learning. TD-VAE implements the STDNN with $K = 2$ where the conditional log likelihood $\log p(\mathbf{x}_{t_2}|\mathbf{x}_{<t_2})$ is calculated over two latent states $\{\mathbf{z}_{t_1}, \mathbf{z}_{t_2}\}$. The current variable $\mathbf{z}_{t_1}$ is regularized by future variable $\mathbf{z}_{t_2}$ based on a *smoothing* distribution $q(\mathbf{z}_{t_1}|\mathbf{z}_{t_2}, \mathbf{b}_{t_1}, \mathbf{b}_{t_2})$ or $q_S^{t_1|t_2}$. For the general case of STDNN with K steps $\{\mathbf{z}_{t_k}\}_{k=1}^{K}$, each current variable $\mathbf{z}_{t_k}$ is smoothed by using the variational distribution $q_S^{t_k|t_{k+1},\ldots,t_K}$ from future variables $\{\mathbf{z}_{t_k+1}, \ldots, \mathbf{z}_{t_K}\}$. This smoothing distribution is used to sequentially predict next state $\mathbf{z}_{t_{k+1}}$ from $\mathbf{z}_{t_k}$, and eventually generate the sample $\mathbf{x}_{t_K}$ at time t_K. The learning objective of STDNN is generalized from Equations (8) to (9) with $K = 2$, to Equation (10) with $K = 3$ and Equations (13)–(17) with a general K. The intuition behind the derived ELBO can be interpreted. In particular, in Equation (17), the first term is the decoder or generative likelihood p_D to be maximized. The resolution in observation space is preserved. The second term is a sum of KL terms which is maximized to promote or encourage a jumpy state-to-state transition from $\mathbf{z}_{t_{k+1}}$ to $\mathbf{z}_{t_{k+2}}$ via p_T for *predictive rollout*. The third term is a sum of KL terms to be minimized to prevent or penalize too much information from future states $\{\mathbf{z}_{t_j}\}_{j=k+2}^{K}$ via smoothing posterior q_S as a *variational*

bottleneck penalty [18]. Prior belief p_B is used to regularize p_T and q_S at $\mathbf{z}_{t_{k+2}}$ and $\mathbf{z}_{t_{k+1}}$, respectively. The summations in Equation (17) contain $K-1$ differences of KL values which implement individual jumpy states at different distant times $\{t_k\}_{k=1}^{K}$ for *state transition* as well as *future planning*.

STDNN can be further illustrated by interpreting its connection with VAE and VRNN which maximizes the ELBO consisting of a log likelihood (or a negative reconstruction error) given by latent samples and a KL divergence between variational posterior and belief prior. Reconstruction term aims to decode the observations from latent variables while KL term forces the latent samples constrained by a prior distribution. An alternative view of STDNN with parameter Θ is to maximize an ELBO similar to VAE or VRNN, but K jumpy states are considered in a belief state representation where KL divergence $\mathcal{D}_{\mathrm{KL}}(p_T^{t_{k+2}} \| p_B^{t_{k+2}})$ is viewed as a constraint which is larger than γ in a form of

$$\begin{aligned} &\max_{\Theta} \ \mathbb{E}_{\mathbf{z}_{t_K}}[\log p_D^{t_K}(\mathbf{x}_{t_K})] - \sum_{k=0}^{K-2} \mathcal{D}_{\mathrm{KL}}(q_S^{t_{k+1}|t_{k+2},\ldots,t_K} \| p_B^{t_{k+1}}) \\ &\text{subject to} \sum_{k=0}^{K-2} \mathcal{D}_{\mathrm{KL}}(p_T^{t_{k+2}} \| p_B^{t_{k+2}}) \geq \gamma. \end{aligned} \tag{14}$$

By introducing a Lagrange multiplier α in this constrained optimization problem, the learning objective $\mathcal{L}_{t_1,\ldots,t_K}$ turns out as $\mathbb{E}_{\mathbf{z}_{t_K}}[\log p_D^{t_K}(\mathbf{x}_{t_K})] - \sum_{k=0}^{K-2} \mathcal{D}_{\mathrm{KL}}(q_S^{t_{k+1}|t_{k+2},\ldots,t_K} \| p_B^{t_{k+1}}) + \alpha \sum_{k=0}^{K-2} \mathcal{D}_{\mathrm{KL}}(p_T^{t_{k+2}} \| p_B^{t_{k+2}})$ where a hyper-parameter α is applied. Figure 3 displays the graphical model for STDNN, which is generalized from TD-VAE in Figure 2 by fulfilling the proposed multi-temporal-difference learning. There are four networks in STDNN. Belief network is used to update the belief states $\mathbf{b}_t$ via RNN to calculate the distributions $p_B^{t_k}(\mathbf{z}_{t_k}) \triangleq p_B(\mathbf{z}_{t_k}|\mathbf{b}_{t_k})$ at each time stamp t_k. Inference network is used to infer the smoothing distribution $q_S^{t_k}$ at current time t_k which is calculated from the belief state $\mathbf{b}_{t_k}$ at current time t_k as well as the latent variables $\mathbf{z}_t$ at future times $t = \{t_j\}_{j=k+1}^{K}$. This distribution is used to sample current state $\mathbf{z}_{t_k}$. Transition network is used to calculate the transition distribution p_T from t_{k-1} to t_k. Finally, the generation network is used to generate the observation sample $\widehat{\mathbf{x}}_{t_K}$ at time t_K.

In addition, STDNN can be seen as a *horizontally* extension of TD-VAE where the modeling of belief states at two temporally-separated time points is extended to that at multiple temporally-separated time points. By referring to the variant of TD-VAE in [18], STDNN can be also *vertically* extended as a hierarchical state machine where higher-level states predict the lower-level states, and ideally represent more abstract information. A deep version of STDNN is accordingly constructed. The belief states in higher layers affect those in lower ones through a recurrent neural network.

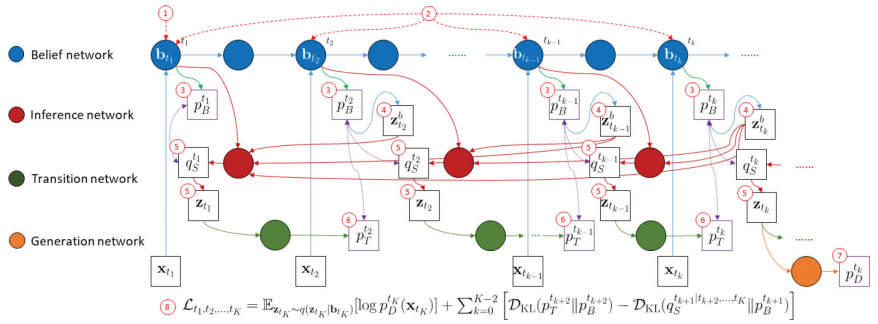


Figure 3: Graphical model for stochastic temporal-difference neural network consisting of belief network, inference network, transition network and generation network for $p_B^{t_k}$, $q_S^{t_k}$, $p_T^{t_k}$ and $p_D^{t_k}$, respectively, marked with different colors. Eight implementation steps are numbered. Purple lines connecting two boxes denote the KL divergence between two distributions.

4.3 Implementation Issues

In this study, *Markov property* and *padding mechanism* are applied to reduce the model size and stabilize the training procedure when increasing the number of jumpy states. In derivation of Equation (17), the *Markov chain* for hidden states

$$p(\mathbf{z}_{t_k}|\mathbf{z}_{t_{k-1}},\ldots,\mathbf{z}_{t_1}) \approx p(\mathbf{z}_{t_k}|\mathbf{z}_{t_{k-1}}), \tag{15}$$

for $3 \leq k \leq K$ has been assumed so as to reduce the input dimensions for transition networks since the state conditional distribution of t_k only depends on t_{k-1}. In addition, the smoothing distributions in inference network were calculated with different input dimensions, but now *zero padding* is enforced to fix the input dimensions by

$$\begin{aligned} &q(\mathbf{z}_{t_{K-1}}|\mathbf{z}_{t_K},\mathbf{b}_{t_{K-1}},\mathbf{b}_{t_K}) \\ &\quad\approx q(\mathbf{z}_{t_{K-1}}|\mathbf{0},\ldots,\mathbf{0},\mathbf{z}_{t_K},\mathbf{0},\ldots,\mathbf{0},\mathbf{b}_{t_{K-1}},\mathbf{b}_{t_K}), \end{aligned} \tag{16}$$

which accordingly turns out to share the inference networks $q_S^{t_k}$. Overall, the stochastic temporal-difference neural network is implemented by fulfilling the following eight steps:

1. Calculate the belief states $\mathbf{b}_t$ from observations $\mathbf{x}_t$ using RNN to construct a deterministic path.

2. Choose a trajectory $\{t_k\}_{k=1}^K$, separated by random intervals, and form the corresponding states $\{\mathbf{z}_{t_k}\}$.

3. Compute the belief prior $p_B^{t_k}$ for states $\mathbf{z}_{t_k}$.

4. Sample the latent states $\mathbf{z}_{t_k}^b$ from belief distribution.

5. Given those states from *future*, the smoothing distribution is inferred as $q_S^{t_k|t_{k+1},\ldots,t_K}$ and used to find samples $\mathbf{z}_{t_k}$.

6. Use these state samples $\mathbf{z}_{t_k}$ to sample next state $\mathbf{z}_{t_{k+1}}$ via transition distribution $p_T^{t_{k+1}}$.

7. Maximize the decoder distribution $p_D^{t_K}$ for reconstructed observation $\widehat{\mathbf{x}}_{t_K}$ using the last state $\mathbf{z}_{t_K}$.

8. Accumulate $\mathcal{L}$ and maximize it by gradient ascent.

Notably, $p_B(\mathbf{z}_{t_k}|\mathbf{b}_{t_k}) \triangleq q(\mathbf{z}_{t_k}|\mathbf{b}_{t_k})$ is the marginal distribution acting as the prior to calculate the smoothing posterior $q(\mathbf{z}_{t_k}|\mathbf{z}_{t_{k+1}},\ldots,\mathbf{z}_{t_K})$. When realizing $(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}) \sim q(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})$ in Equation (17), we first individually draw samples $\mathbf{z}_{t_k}^b$ via the belief prior $p_B(\mathbf{z}_{t_k}|\mathbf{b}_{t_k})$ given by belief state $\mathbf{b}_{t_k}$. The samples $\mathbf{z}_{t_k}$ of the smoothing posterior $q(\mathbf{z}_{t_k}|\mathbf{z}_{t_{k+1}},\ldots,\mathbf{z}_{t_K})$ are then sequentially drawn in the factorized sampling procedure in a backward manner. Samples of $\mathbf{z}_{t_k}^b$ and $\mathbf{z}_{t_k}$ are different. STDNN is seen as the belief-state-based model.

STDNN is seen as an unsupervised learning, but is definitely feasible to supervised learning by incorporating the one-hot output label $\mathbf{y}_{t_k}$ with M classes for input data $\mathbf{x}_{<t_k}$ at a target time t_k. Algorithm 1 illustrates the supervised learning procedure of STDNN with six parameters where an additional classification network is merged. First, belief network is implemented to continuously update the belief state $\mathbf{b}_{t_k}$ by using RNN based on long short-term memory (LSTM) [22] with parameter θ_B. Second, the belief network with sampling parameter θ_S is used to find the Gaussian parameters in belief distribution $p_B^{t_k} = f_{\theta_S}(\mathbf{b}_{t_k})$. Latent variables $\mathbf{z}_{t_k}$ are then sampled by the prior $p_B^{t_k}$. Similar to $\{f_\theta^{(r)}, f_\theta^{(p)}\}$ in VRNN, belief network in STDNN contains the recurrent and prior parameters $\{\theta_B, \theta_S\}$. Third, the inference network with variational parameter ϕ_I is implemented to find the smoothing distribution via $q_S^{t_k|t_{k+1},\ldots,t_K} = f_{\phi_I}(\{\mathbf{z}_{t_j}\}_{j=k+1}^K, \{\mathbf{b}_{t_j}\}_{j=k}^K)$. Latent variables $\mathbf{z}_{t_k}$ are again sampled by variational posterior q_S. Four, the state prediction network with parameter θ_T is used to find transition distribution $p_T^{t_{k+1}} = f_{\theta_T}(\mathbf{z}_{t_k})$.

Five, the generation network with parameter θ_D is applied to decode the sample $\widehat{\mathbf{x}}_{t_K}$ via $p_D^{t_K} = f_{\theta_D}(\mathbf{z}_{t_K})$. Six, the classification network with softmax parameter θ_C is used to estimate target output $\widehat{\mathbf{y}}_{t_K} = f_{\theta_C}(\widehat{\mathbf{x}}_{t_K})$. The ELBOs of generative likelihood $p(\mathbf{x})$ and conditional likelihood $p(\mathbf{y}|\mathbf{x})$ of input sequence $\mathbf{x}$ and target sequence $\mathbf{y}$ can be expressed by $\mathcal{L}(\mathbf{x};\theta_B,\theta_S,\phi_I,\theta_T,\theta_D) = \log p(\mathbf{x})$ via $\widehat{\mathbf{x}}_t$ and $\mathcal{L}(\mathbf{x},\mathbf{y};\theta_B,\theta_S,\phi_I,\theta_T,\theta_D,\theta_C)$ via $\widehat{\mathbf{y}}_t$ as the negative cross entropy error between estimated $\widehat{\mathbf{y}}$ and true targets $\mathbf{y}$, $\log p(\mathbf{y}|\mathbf{x}) = \sum_{t=1}^T \sum_{m=1}^M y_{tm}$

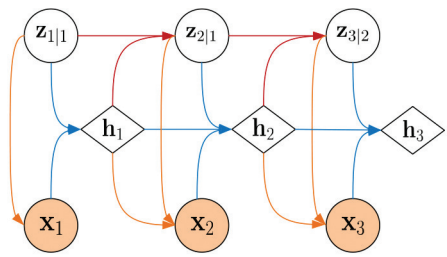
Algorithm 1: Supervised learning procedure for stochastic temporal-difference neural network.

Input training mini-batches $\mathbf{x} = \{\mathbf{x}^i\}$ & $\mathbf{y} = \{\mathbf{y}^i\}$
Initialize $\theta_B, \theta_S, \phi_I, \theta_T, \theta_D$, θ_C for belief, inference, transition, decoder and classification networks
while $\theta_B, \theta_S, \phi_I, \theta_T, \theta_D$ *not converged* **do**
  for *each mini-batch* $\boldsymbol{x}^i$ *and* $\boldsymbol{y}^i$ **do**
    Belief network:
    $\mathbf{b}_t \leftarrow \text{LSTM}(\mathbf{b}_{t-1}, \mathbf{x}_t^i, \theta_B)$
    choose K random time steps $t_1, \ldots, t_K$
    for $k = 1, \ldots, K$ **do**
      calculate the belief distribution $p_B^{t_k} = f_{\theta_S}(\mathbf{b}_{t_k})$
      sample $\mathbf{z}_{t_k}^b$ from belief prior $p_B^{t_k}$
    end
    for $k = 1, \ldots, K-1$ **do**
      Inference network:
      calculate the inferred posterior $q_S^{t_k|t_{k+1},\ldots,t_K} = f_{\phi_I}(\{\mathbf{z}_{t_j}^b\}_{j=k+1}^K, \{\mathbf{b}_{t_j}\}_{j=k}^K)$
      sample $\mathbf{z}_{t_k}$ from $q_S^{t_k|t_{k+1},\ldots,t_K}$
      Transition network:
      calculate the transition distribution $p_T^{t_{k+1}} = f_{\theta_T}(\mathbf{z}_{t_k})$ to sample $\mathbf{z}_{t_{k+1}}$
    end
    Decoder network:
    calculate the generation distribution $p_D^{t_K} = f_{\theta_D}(\mathbf{z}_{t_K})$ to decode $\widehat{\mathbf{x}}_{t_K}$
    Classification network:
    calculate the class output $\widehat{\mathbf{y}}_{t_K} = f_{\theta_C}(\widehat{\mathbf{x}}_{t_K})$
    accumulate the ELBO of $p(\mathbf{y}|\mathbf{x})$ as $\mathcal{L}(\mathbf{x}^i, \mathbf{y}^i; \theta_B, \theta_S, \phi_I, \theta_T, \theta_D, \theta_C)$
    update the parameters by gradients $\frac{\partial \mathcal{L}}{\partial \theta_B}, \frac{\partial \mathcal{L}}{\partial \theta_S}, \frac{\partial \mathcal{L}}{\partial \phi_I}, \frac{\partial \mathcal{L}}{\partial \theta_T}, \frac{\partial \mathcal{L}}{\partial \theta_D}, \frac{\partial \mathcal{L}}{\partial \theta_C}$
  end
end

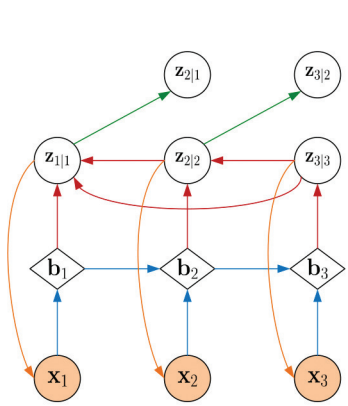
$\log \widehat{y}_{tm}$, which are calculated by Equation (17) and maximized to fulfill unsupervised learning and supervised learning, respectively. Parameters $\Theta = \{\theta_B, \theta_S, \phi_I, \theta_T, \theta_D, \theta_C\}$ are then updated by stochastic gradient ascent algorithm. When comparing TD-VAE in Figure 2 and STDNN in Figure 3 with VRNN in Figure 1, we find that the agents in TD-VAE and STDNN are able to adopt the future features at $\{t_{k+1}, \ldots, t_K\}$ to smooth the current feature at

t_k, as shown in red lines and nodes. The strategy of planning is implemented in this Bayesian multi-temporal-difference learning with the perspective from RL. Nevertheless, in inference or test time, the prediction of an observation $\mathbf{x}_t$ at time t is only based on the history $\mathbf{x}_{<t}$ by using latent variables $\{\mathbf{z}_t\}$ or $\{\mathbf{z}_{t_k}\}$ obtained from belief network p_B, transition network p_T and decoder network p_D. The inference network q_S is not required in inference time.

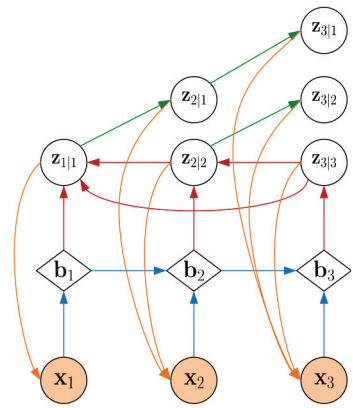
In the implementation, the latent samples $\mathbf{z}_{t_k}$ at each target time t_k are drawn from the inference network q_S as $\mathbf{z}_{t_k|t_k}$ as well as the transition network p_T as $\mathbf{z}_{t_k|t_{k-1}}$ which is transited from time t_{k-1}. The overshooting regularization [21] is feasible to enhance the robustness in multi-step prediction. Figure 4 compares the state transitions based on VRNN and STDNN. VRNN is seen as a *causal learner* via $\{\mathbf{h}_t\}$ which performs step-by-step learning where the overshooting is not valid as shown in Figures 1 and 4(a). Figures 3



(a) VRNN



(b) STDNN w/o overshooting



(c) STDNN with overshooting

Figure 4: State transitions in VRNN and STDNN without/with overshooting. Red lines denote the inference network, green lines denote the transition network and orange lines denote the generation network. Deterministic and random variables are shown by diamond and circle nodes, respectively.

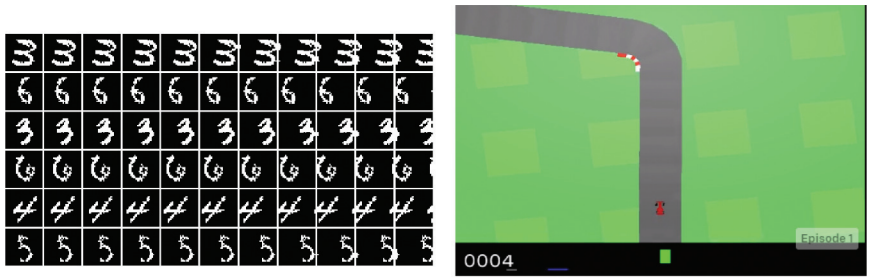
and 4(b,c) illustrate how STDNN is implemented for multi-step prediction with $\{\mathbf{b}_{t_k}\}$ where $\mathbf{z}_{t_k|t_k}$ is inferred or smoothed in a *backward* manner. Figure 4(c) shows how the observation overshooting is employed to perform the augmented reconstruction. STDNN optimizes the generative model of sample $\widehat{\mathbf{x}}_{t_k}$ based on the inferred state $\mathbf{z}_{t_k}$. By applying the observation overshooting, the sample $\widehat{\mathbf{x}}_{t_k}$ depends on the multiple state variables $\mathbf{z}_{t_k}$ at time t_k which are estimated from different paths including the state $\mathbf{z}_{t_k|t_k}$ inferred from t_k via q_S, the state $\mathbf{z}_{t_k|t_{k-1}}$ inferred from t_{k-1} and then transited to t_k via p_T, and the state $\mathbf{z}_{t_k|t_{k-2}}$ inferred from trajectory $\{t_{k-2}, t_{k-1}, t_k\}$ via p_T. Owing to the multiple variables $\mathbf{z}_{t_k}$, it is meaningful to augment the learning objective by merging the additional reconstruction errors (or log likelihoods $p(\mathbf{x}_{t_k}|\mathbf{z}_{t_k})$ or $p_D^{t_k}(\mathbf{x}_k)$) as the regularization terms. However, the computational overhead due to overshooting from multiple paths is required. This paper is considerably extended from the previous work [9] by enriching the survey of literature, illustrating the evolution of related works, detailing the derivation of learning objectives, and enhancing the comparison of results as illustrated in what follows.

5 Experiments

This study conducted unsupervised and supervised learning for sequence prediction, language modeling and sentiment classification.

5.1 Experimental Setup

Different tasks were conducted in the evaluation. The task on sequence prediction aims to predict or rollout a number of future frames in video simulation based on the partially observed sequence data. As shown in Figure 5, there were two datasets evaluated for sequence prediction in multiple time steps via unsupervised learning. The first dataset was the moving MNIST. This dataset consisted of 60K video simulations. Each simulation was a sequence of 20 frames. For each sequence, a random digit and a random direction of right or left were chosen. At each frame, a digit of 28×28 pixels was moved by one pixel in a chosen direction. The second dataset was the CarRacing-v0 video frames collected from RL environment in OpenAI Gym [3]. This was a continuous control task to learn from pixels. We collected 27 videos where each video had the length of 732 frames. In each video, a red car was driving along gray road surrounded by green ground. Sequential learning was performed to predict car direction and road scenario. In this task, the first two-third of the sequences were arranged for training and validation, and the rest of the sequences were used for testing. Validation data were used for hyperparameter tuning. Bayesian multi-temporal-difference learning was applied to capture the temporal as well spatial information in a video clip.



(a) moving MNIST (b) car-racing environment

Figure 5: Examples in the experiments on video prediction.

In addition, the Bayesian modeling of sequence data was also evaluated for two supervised classification tasks where the target outputs $\mathbf{y}$ corresponding to sequence inputs $\mathbf{x}$ were provided. The first task was to evaluate the performance of language model of text sequence $\mathbf{x} = \{\mathbf{x}_t\}_{t=1}^T$ where the conditional likelihood was maximized to predict word $\mathbf{x}_t$ at time t as the target word $\mathbf{y}_t$ given its history words $\mathbf{x}_{<t}$ [5]. Sequential learning based on STDNN was to produce the softmax outputs $\widehat{\mathbf{y}}_t$ with classification parameters θ_C to match the one-hot classification outputs of predicted words $\mathbf{y}_t$ by maximizing the ELBO of conditional log likelihood or negative cross entropy error. Language model was examined in terms of negative log likelihood (NLL) and perplexity (PPL) of test sentences where the lower the better. The second task was to train a STDNN to find classification output $\mathbf{y}_T$ corresponding to a text document $\{\mathbf{x}_t\}_{t=1}^T$ for sentiment classification. There were two datasets in this evaluation. Penn Treebank (PTB) [31] was collected as a benchmark dataset for language modeling. PTB consisted of news documents containing 929K training words, 73K validation words and 82K test words with a vocabulary of 10K words. Stop words were excluded.

The other database was the Internet Movie Database (IMDB) [29]. IMDB consisted of 50K movie reviews. The original setting of training, validation and test data was referred. A dictionary with 20K words was used. IMDB was adopted to investigate the performance of language modeling as well as sentiment classification. The averaged length in a document in PTB and IMDB was 21 and 78 words, respectively. For a comparative study, different sequential learning methods were evaluated. We implemented the classification models based on RNN [31], Bayesian RNN (BRNN) [5], VRNN [12], Z-forcing [17], TD-VAE [18] (STDNN with $K = 2$) and STDNN with different K. Z-forcing was seen as a kind of bidirectional VRNN (simply denoted as Bi-VRNN) where the stochastic state $\mathbf{z}_t$ was simultaneously inferred from forward as well as backward RNNs at each time t. The recent methods based on dropoutRNN [15], skip-RNN [19] and transformer [40] were included in the comparison. LSTM [22] was consistently adopted in RNN-based solutions.

5.2 Detailed Implementation

In the implementation, Markov assumption and zero padding in Equations (15) and (16) were consistently applied to simplify STDNN with a single transition network and a single inference network, respectively. These two tricks were required to obtain desirable results. The mini-batch size was fixed as 32. Batch normalization was applied in different layers. Adam optimization [25] was used with the annealed learning rate which was decreased by a factor when NLL of validation data was higher than previous NLL. This factor was tuned in different tasks. Pytorch was used in the implementation. Training time relative to RNN was measured. Number of parameters using different sequential learning was compared.

For sequence prediction, the dimensions of belief states $\mathbf{b}_{t_k}$ (or hidden states $\mathbf{h}_t$) and random states $\mathbf{z}_{t_k}$ were 50 and 20, respectively. The random distance $t_k - t_{k-1}$ between two associated states $\{\mathbf{z}_{t_{k-1}}, \mathbf{z}_{t_k}\}$ in multi-temporal-difference learning was sampled from a discrete uniform distribution in a range $[1, 4]$. Rollout prediction was based on the final time step T of an input sequence. Belief network consisted of one layer of LSTM and one fully-connected layer of prior network. Inference network and transition network were both implemented by two fully connected layers. In moving MNIST task, the generation network was fully connected with three hidden layers. The activation functions using the hyperbolic tangent in the first two layers and the sigmoid in the last layer were specified. In CarRacing-v0 task, the generation network was composed of four de-convolutional layers. There was an additional pre-processing network to receive video frames by using two convolutional layers combined with max pooling. In both tasks, 9K learning epochs were run.

In the tasks of PTB and IMDB, the interval of jumpy states was fixed to one. The step-by-step prediction was performed. The decoding likelihood due to the reconstructed sample $\widehat{\mathbf{x}}_{t_k}$ was affected by $\{\mathbf{z}_{t_k}, \mathbf{b}_{t_k}\}$. STDNN fused the information of the deterministic states $\mathbf{b}_{t_k}$ with past step-by-step patterns and the stochastic states $\mathbf{z}_{t_k}$ with future jumpy clues at each prediction time t_k. In addition to this STDNN cost, we imposed NLL as an auxiliary cost which directly benefited the evaluation of NLL and PPL. In learning procedure with each mini-batch, NLL cost was minimized one time before minimizing STDNN cost five times. The settings in PTB and IMDB were similar to those in sequence prediction except that the word embedding size, the hidden state size ($\mathbf{h}_t$ or $\mathbf{b}_t$), the latent variable size ($\mathbf{z}_t$) were 400, 1150 and 40, respectively, and the number of training epochs was 30. Similar to [12], the deterministic recurrent states $\mathbf{h}_t$ (or $\mathbf{b}_t$) and the stochastic states $\mathbf{z}_t$ were both treated as inputs to generation network to strengthen language modeling. Number of states K was varied from 2 to 3, 4, 5, 8, 12 and 16 for comparison in different tasks. There were six layers in encoder/decoder of transformer. BRNN [5] had similar structure as RNN [31]. VRNN [12] and Bi-VRNN [17] used similar

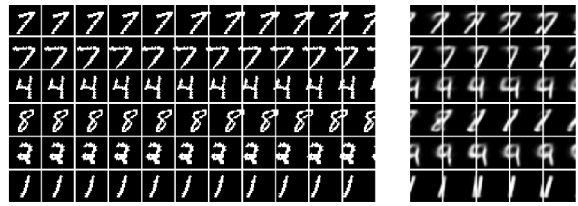
structure of inference and generation networks as TD-VAE and STDNN. This work ignored the implementation of deep STDNN.

5.3 Evaluation on Sequence Prediction

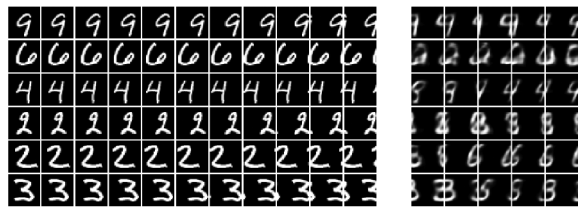
First of all, the unsupervised learning based on TD-VAE and STDNN with different K was evaluated for sequence prediction. Figure 6 shows the input sequences on the left-hand-side and the rollout sequences on the right-hand-side where the effects of K and the observation overshooting in Figure 4(c) are examined. Five frames in a rollout sequence are generated by the latent variable $\mathbf{z}_{t_k}$ inferred at the last frame of the corresponding input sequence. TD-VAE and STDNN are learned to predict not only the spatial images but also the moving directions. The prediction of frames and directions for several time steps is achieved. Prediction capability is preserved through the temporal-difference learning. Notably, these rollout results are obtained by jumpy predictions rather than step-by-step predictions by one pixel motion. Nevertheless, using TD-VAE, the digits in the third and fifth input sequences are varied to the other digits in their rollout sequences. This situation is less likely to happen for those sequences using STDNN. When comparing Figure 6(b) and (c), we can see that the observation overshooting does enhance the resolution in sequence generation. Hereafter, overshooting is consistently applied. In addition, the rollouts are improved by increasing K. For example, the digit '4' in the third input sequence with $K = 3$ is changed to digit '9' in rollout sequence. Such a variation is unseen in the results of STDNN with $K = 4$ and $K = 5$.

Figure 7 compares two-dimensional (2-D) visualizations [30] for the inferred samples of latent variables $\mathbf{z}_{t_k}$ by using TD-VAE and STDNN with $K = 3$. MNIST is evaluated. Different digits are reflected by various colors. Markers $\times$ show the rollout samples of a video of the moving digit '5'. It is obvious that the clustering of latent samples of different digits using STDNN is more visible than that using TD-VAE. When evaluating the jumpy process for prediction, the rollouts using these two methods are well behaved under the same digit '5'. The rollout samples using STDNN are closer to each other than those samples using TD-VAE. Sequential learning via STDNN works not only for clustering of latent variables but also for jumping to future states.

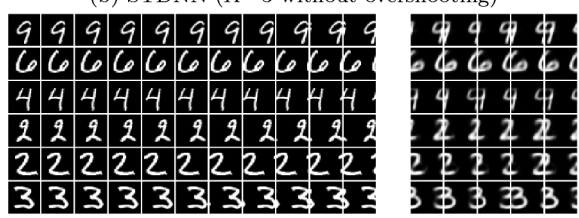
Figure 8 displays three examples of rollouts by using STDNN with $K = 4$ where car-racing environment under different road conditions is investigated. The video frames on the left-hand-side are the input sequences and those on the right-hand-side are the rollout frames based on latent variable in the final input frame inferred from input sequence. Again, STDNN can predict future frames correctly without step-by-step prediction under different road conditions. STDNN is able to predict possible scenarios through planning the future states based on jumpy rollouts. Figure 8(a) shows simple road conditions where the car drives along straight roads. Figure 8(b) and (c) are



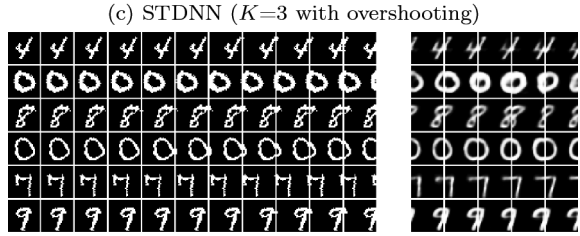
(a) TD-VAE



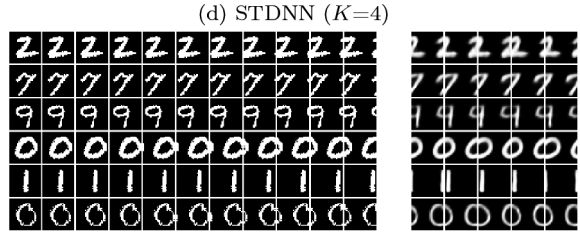
(b) STDNN (K=3 without overshooting)



(c) STDNN (K=3 with overshooting)

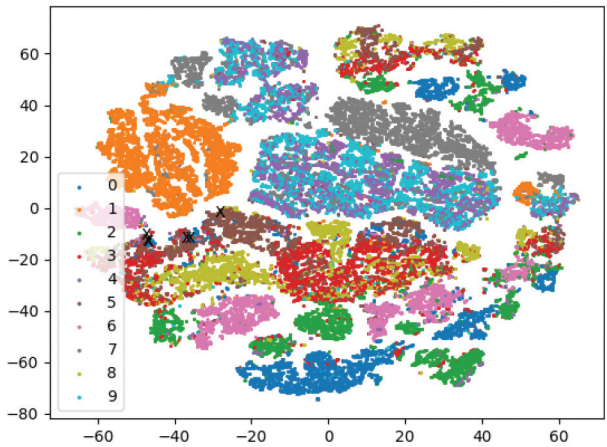


(d) STDNN (K=4)

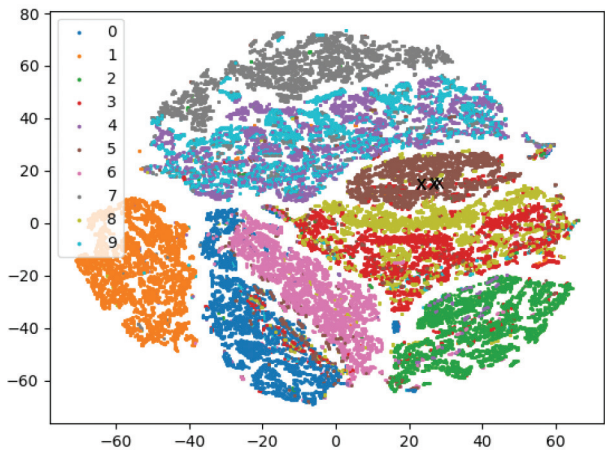


(e) STDNN (K=5)

Figure 6: Rollout results of moving MNIST by using TD-VAE and STDNN where different K without/with overshooting is evaluated. Left: input sequences. Right: jumpy rollouts.



(a) TD-VAE



(b) STDNN (K=3)

Figure 7: 2-D latent visualization using TD-VAE and STDNN. Different colors show the samples $\mathbf{z}_{t_k}$ for different MNIST digits. Markers × denote the rollout samples of a video.

seen as complex scenarios with changing roads. STDNN predicts successfully for different conditions including the circumstances that the car drives along the road in a slowly-changing scenario as well as outside the road in a rapidly-changing scenario, e.g. fourth sequence in Figure 8(b) and fifth sequence in 8(c). Table 1 reports the values of negative ELBO and the estimated negative

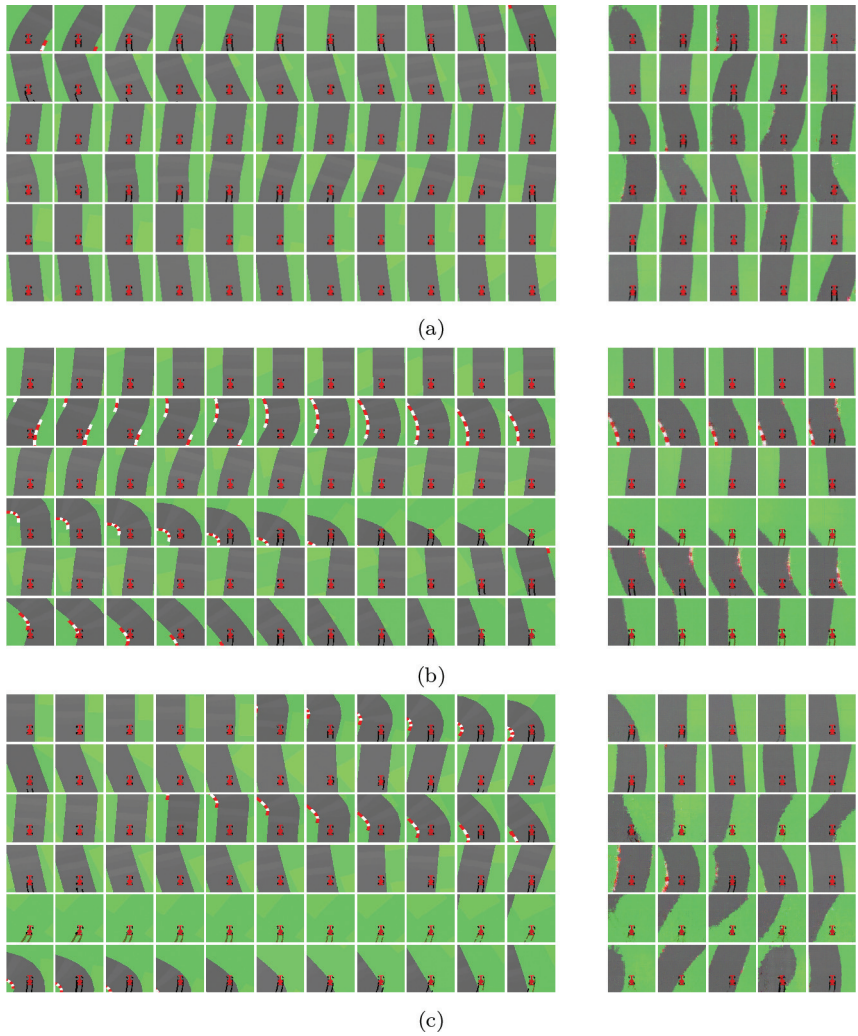


Figure 8: Rollout results of racing car by using STDNN with $K = 4$ under different car directions and road scenarios. Left: input sequences. Right: jumpy rollouts.

log probability [18] using a test set of 10 videos where rollout frames of test sequences are evaluated. The lower the better. The range of values with one standard deviation is shown. STDNN with different K obtains lower values than TD-VAE in prediction of future frames. The best result (shown in bold) is obtained by using STDNN at $K = 4$ with the desirable confidence.

Table 1: Comparison of negative ELBO and estimated negative log probability on a test set of car-racing sequences.

Model	- ELBO	$-\log p(\mathbf{x})$
TD-VAE	0.1534±0.0021	0.0997±0.0037
STDNN ($K = 3$)	0.1358±0.0013	0.0823±0.0021
STDNN ($K = 4$)	**0.1032±0.0008**	**0.0439±0.0015**
STDNN ($K = 5$)	0.1211±0.0015	0.0527±0.0013

5.4 Evaluation on Language Modeling

Sequential learning is further evaluated for language modeling where PTB dataset is used. Using STDNN, language model (LM) is trained to act in a way that future neighboring words are planned and characterized in combination with the latent features from history words for prediction of next word at each time step. Table 2 compares NLL, PPL, model size and training time of LMs using different methods. The previous results on PPL and number of parameters using RNN [31], dropoutRNN [15] and skip-RNN [19] are included. The training time relative to RNN is reported. The unity time corresponds to the computation for RNN. For a comparative study, we also implemented the RNN-large by increasing the width and depth of RNN [31] with a comparable model size as that of STDNN. It is found find that TD-VAE and STDNN perform better than the other variational methods based on VRNN and Bi-VRNN as well as the attention-based method using transformer [40] in terms of NLL and

Table 2: Comparison of NLL, PPL, number of parameters (#Par) and computation time for language modeling over different methods. PTB is evaluated.

Model	NLL	PPL	#Par	Time
RNN [31]	–	124.7	6M	1x
RNN-large	93.0	101.6	22M	3.2x
DropoutRNN [15]	–	78.6	20M	–
Skip-RNN [19]	–	76.5	19M	–
BRNN [5]	95.3	116.3	7M	1.4x
VRNN [12]	83.7	97.2	15M	1.9x
Bi-VRNN [17]	71.1	83.5	23M	2.5x
Transformer [40]	70.5	81.6	23M	3.0x
TD-VAE	65.4	78.9	20M	2.7x
STDNN ($K = 4$)	63.1	76.9	22M	3.0x
STDNN ($K = 8$)	**61.1**	72.1	23M	3.3x
STDNN ($K = 12$)	61.6	**70.8**	23M	3.6x
STDNN ($K = 16$)	61.8	71.9	24M	3.9x

PPL. This is partially because the trained TD-VAE and STDNN are embedded with future information which is helpful for prediction in language modeling.

The lowest NLL and PPL are achieved by using STDNN in this comparison. The results with $K = 8$, 12 and 16 are comparable but are much better than those of $K = 2$ (namely TD-VAE) and $K = 4$. STDNN performs better than the existing methods [15, 19, 31], [40] in NLL and PPL although STDNN adopts a larger number of parameters. The costs of memory and computation using STDNN are higher than those of the other methods. Nevertheless, the scale of the increase of these two costs using STDNN due to large K is limited. This is because the tricks of Markov assumption and zero padding are employed. Under similar model size, STDNN significantly reduces NLL and PPL in comparison of RNN-large.

5.5 Evaluation on Sentiment Classification

In addition to PTB dataset, language models using different recurrent machines are evaluated in the task of IMDB. The results on NLL and PPL are illustrated in Table 3. The accuracy of sentiment classification is also included. Similar to the results in PTB, STDNN with different K obtains lower NLL and PPL than the related methods using RNN, BRNN, VRNN, Bi-VRNN, transformer and TD-VAE. The lowest NLL and PPL were achieved by using STDNN at $K = 12$. STDNN at $K = 16$ is over-parameterized. The resulting performance is degraded in terms of NLL and PPL. Basically, value K is domain dependent and is related to the characteristics of sequence data. Selection of K is known as a model selection problem which can be handled by Bayesian learning [42], but is disregarded in this study. In addition to the evaluation of language

Table 3: Comparison of NLL, PPL and classification accuracy for sentiment classification over different methods. IMDB is evaluated.

Model	NLL	PPL	Acc (%)
RNN [31]	180.1	71.3	85.3
Skip-RNN [19]	–	–	89.1
BRNN [5]	176.3	68.5	87.4
VRNN [12]	169.1	62.1	88.4
Bi-VRNN [17]	162.3	59.7	89.3
Transformer [40]	160.1	58.6	89.1
TD-VAE	157.8	57.4	90.5
STDNN ($K = 4$)	154.3	56.1	**92.1**
STDNN ($K = 8$)	149.8	54.9	91.8
STDNN ($K = 12$)	**146.8**	**53.8**	91.7
STDNN ($K = 16$)	150.1	56.0	91.3

modeling, IMDB dataset is also applied to assess the performance of sentiment classification. The classification accuracy for positive and negative reviews is reported and compared over different methods. In this comparison, the accuracy of skip-RNN in [15] is referred. Using STDNN with $K = 4$, the accuracy is increased as high as 92.1% which is better than 89.1% by using skip-RNN. From a series of comparisons on different methods in different tasks, it is expected to obtain desirable performance in sequential learning via probabilistic modeling, future planning and multi-temporal-difference learning. Source codes are accessible at https://github.com/NCTUMLlab/Yi-Chung-Chiu.

6 Conclusions

This paper presented a new sequential learning where the perspective of reinforcement learning was implemented and merged in unsupervised learning for sequence prediction as well as in supervised learning for language modeling and sentiment classification. The multi-temporal-difference learning was generalized as the multi-step variant of temporal-difference variational autoencoder where twofold novelties were pursued for sequential learning. First, Bayesian modeling was carried out for temporal-difference learning via variational inference. Latent variables were learned from the past as well as the future. Second, multiple-temporal-difference learning was developed to capture the characteristics of a trajectory of association patterns in sequence data at multiple time steps which significantly differed from traditional sequential learning based on step-by-step prediction without planning. A general formulation for stochastic temporal-difference neural network was proposed. Time abstraction was represented. A training algorithm of belief network, inference network, transition network, generation network and classification network was derived. The tricks of Markov assumption, zero padding and observation overshooting were addressed. Experiments showed that rollout sequences using this method could preserve the temporal dynamics in input sequences. Perplexity of the resulting language model was decreased. Accuracy in sentiment classification was increased. Implementation costs were increased. Future work will be extended to build a multi-step search agent which acts as a flexible solution to Bayesian optimization [36] for hyperparameter tuning.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 110-2221-E-A49-074-MY3.

Appendix: Derivation for the ELBO of STDNN

The detailed derivation of ELBO of STDNN is shown in Equation (17) where the paired terms for each individual KL divergence are shown. This derivation is manipulated by adding and substituting the sequence of terms p_B, which are arranged and merged in the summation for the difference between two KL terms similar to that of TD-VAE in Equation (9). The property of *Markov chain* is applied in the derivation.

$$
\begin{aligned}
\mathcal{L}_{t_1,\ldots,t_K} &= \mathbb{E}_{(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K})\sim q(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})} \times \big[\log p(\mathbf{x}_{t_K}|\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K},\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_{K-1}}) \\
&\quad + \log p(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_{K-1}}) - \log q(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})\big] \\
&= \mathbb{E}_{(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K})\sim q(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})} [\log p(\mathbf{x}_{t_K}|\mathbf{z}_{t_K}) \\
&\quad + \log p(\mathbf{z}_{t_K}|\mathbf{z}_{t_{K-1}},\ldots,\mathbf{z}_{t_1}) + \log p(\mathbf{z}_{t_{K-1}}|\mathbf{z}_{t_{K-2}},\ldots,\mathbf{z}_{t_1}) + \cdots \\
&\quad + \log p(\mathbf{z}_{t_2}|\mathbf{z}_{t_1}) + \log p_B(\mathbf{z}_{t_1}|\mathbf{b}_{t_1}) - \log q(\mathbf{z}_{t_1}|\mathbf{z}_{t_2},\ldots,\mathbf{z}_{t_K},\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K}) \\
&\quad - \log q(\mathbf{z}_{t_2}|\mathbf{z}_{t_3},\ldots,\mathbf{z}_{t_K},\mathbf{b}_{t_2},\ldots,\mathbf{b}_{t_K}) - \cdots - \log q(\mathbf{z}_{t_{K-1}}|\mathbf{z}_{t_K},\mathbf{b}_{t_{K-1}},\mathbf{b}_{t_K}) \\
&\quad - \log p_B(\mathbf{z}_{t_K}|\mathbf{b}_{t_K})] \\
&= \mathbb{E}_{(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K})\sim q(\mathbf{z}_{t_1},\ldots,\mathbf{z}_{t_K}|\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})} \Bigg[\underbrace{\log p(\mathbf{x}_{t_K}|\mathbf{z}_{t_K})}_{\log p_D^{t_K}} \\
&\quad \underbrace{+\log p(\mathbf{z}_{t_K}|\mathbf{z}_{t_{K-1}})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_K}\|p_B^{t_K})} \underbrace{+\log p(\mathbf{z}_{t_K-1}|\mathbf{z}_{t_{K-2}})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_K-1}\|p_B^{t_K-1})} + \ldots \underbrace{+\log p(\mathbf{z}_{t_2}|\mathbf{z}_{t_1})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_2}\|p_B^{t_2})} \\
&\quad \cancel{+\log p_B(\mathbf{z}_{t_1}|\mathbf{b}_{t_1})} \cancel{-\log p_B(\mathbf{z}_{t_1}|\mathbf{b}_{t_1})} \underbrace{-\log p_B(\mathbf{z}_{t_2}|\mathbf{b}_{t_2})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_2}\|p_B^{t_2})} - \cdots \\
&\quad \underbrace{-\log p_B(\mathbf{z}_{t_{K-1}}|\mathbf{b}_{t_{K-1}})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_K-1}\|p_B^{t_K-1})} \underbrace{-\log p_B(\mathbf{z}_{t_K}|\mathbf{b}_{t_K})}_{\mathcal{D}_{\mathrm{KL}}(p_T^{t_K}\|p_B^{t_K})} \underbrace{+\log p_B(\mathbf{z}_{t_1}|\mathbf{b}_{t_1})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_1}\|p_B^{t_1})} \\
&\quad \underbrace{+\log p_B(\mathbf{z}_{t_2}|\mathbf{b}_{t_2})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_2}\|p_B^{t_2})} + \cdots \underbrace{+\log p_B(\mathbf{z}_{t_{K-1}}|\mathbf{b}_{t_{K-1}})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_K-1}\|p_B^{t_K-1})} \cancel{+\log p_B(\mathbf{z}_{t_K}|\mathbf{b}_{t_K})} \\
&\quad \underbrace{-\log q(\mathbf{z}_{t_1}|\mathbf{z}_{t_2},\ldots,\mathbf{z}_{t_K},\mathbf{b}_{t_1},\ldots,\mathbf{b}_{t_K})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_1}\|p_B^{t_1})} \\
&\quad \underbrace{-\log q(\mathbf{z}_{t_2}|\mathbf{z}_{t_3},\ldots,\mathbf{z}_{t_K},\mathbf{b}_{t_2},\ldots,\mathbf{b}_{t_K})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_2}\|p_B^{t_2})} - \cdots \underbrace{-\log q(\mathbf{z}_{t_{K-1}}|\mathbf{z}_{t_K},\mathbf{b}_{t_{K-1}},\mathbf{b}_{t_K})}_{-\mathcal{D}_{\mathrm{KL}}(q_S^{t_K-1}\|p_B^{t_K-1})} \\
&\quad \cancel{-\log p_B(\mathbf{z}_{t_K}|\mathbf{b}_{t_K})} \Bigg] \\
&= \mathbb{E}_{\mathbf{z}_{t_K}\sim q(\mathbf{z}_{t_K}|\mathbf{b}_{t_K})} \left[\log p_D^{t_K}(\mathbf{x}_{t_K})\right] \\
&\quad + \sum_{k=0}^{K-2} \left[\mathcal{D}_{\mathrm{KL}}(p_T^{t_{k+2}}\|p_B^{t_{k+2}}) - \mathcal{D}_{\mathrm{KL}}(q_S^{t_{k+1}|t_{k+2},\ldots,t_K}\|p_B^{t_{k+1}})\right] \qquad (17)
\end{aligned}
$$

Biographies

Jen-Tzung Chien is the Lifetime Chair Professor in electrical and computer engineering, and computer science at the National Yang Ming Chiao Tung University, Taiwan. He served as the tutorial speaker for top conferences including AAAI, IJCAI, ACL, KDD, MM, ICASSP, ICME, CIKM, IJCNN, COLING and Interspeech. Dr. Chien has published extensively, including three books and 250 peer-reviewed articles, many on machine learning, deep learning and Bayesian learning with applications on natural language processing and computer vision.

Yi-Chung Chiu received the B.S. and M.S. degrees from National Kaohsiung Normal University and National Chiao Tung University, in 2017 and 2019, respectively, both in electrical and computer engineering. His research interests include deep learning, sequential learning and reinforcement learning.

References

[1] E. Aksan and O. Hilliges, "STCN: Stochastic Temporal Convolutional Networks," in *Proceedings of International Conference on Learning Representations*, 2019.

[2] E. Bengio, J. Pineau, and D. Precup, "Interference and Generalization in Temporal Difference Learning," in *Proceedings of International Conference on Machine Learning*, 2020, 767–77.

[3] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[4] J.-T. Chien, "Association Pattern Language Modeling," *IEEE Transactions on Audio, Speech, and Language Processing*, 14(5), 2006, 1719–28.

[5] J.-T. Chien and Y.-C. Ku, "Bayesian Recurrent Neural Network for Language Modeling," *IEEE Transactions on Neural Networks and Learning Systems*, 27(2), 2016, 361–74.

[6] J.-T. Chien and C. Shen, "Stochastic Recurrent Neural Network for Speech Recognition," in *Proceedings of Annual Conference of International Speech Communication Association*, 2017, 1313–7.

[7] J.-T. Chien, "Neural Bayesian information processing," in *Proceedings of ACM International Conference on Information and Knowledge Management*, 2020, 3501–2.

[8] J.-T. Chien and Y.-H. Chen, "Learning Continuous-Time Dynamics with Attention," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[9] J.-T. Chien and Y.-C. Chiu, "Stochastic Temporal Difference Learning for Sequence Data," in *Proceedings of International Joint Conference on Neural Networks*, 2021, 1–6.

[10] J.-T. Chien, W.-L. Liao, and I. El Naqa, "Exploring State Transition Uncertainty in Variational Reinforcement Learning," in *Proceedings of European Signal Processing Conference*, 2021, 1527–31.

[11] J.-T. Chien and C.-W. Wang, "Hierarchical and Self-Attended Sequence Autoencoder," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9), 2022, 4975–86.

[12] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A Recurrent Latent Variable Model for Sequential Data," in *Advances in Neural Information Processing Systems*, 2015, 2980–8.

[13] S. Evmorfos, K. I. Diamantaras, and A. P. Petropulu, "Reinforcement Learning for Motion Policies in Mobile Relaying Networks," *IEEE Transactions on Signal Processing*, 70, 2022, 850–61.

[14] W. Fedus, I. Goodfellow, and A. M. Dai, "MaskGAN: Better Text Generation via Filling in the _____," in *Proceedings of International Conference on Learning Representations*, 2018.

[15] Y. Gal and Z. Ghahramani, "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks," in *Advances in Neural Information Processing Systems*, 2016, 1019–27.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014, 2672–80.

[17] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio, "Z-forcing: Training Stochastic Recurrent Networks," in *Advances in Neural Information Processing Systems*, 2017, 6713–23.

[18] K. Gregor, G. Papamakarios, F. Besse, L. Buesing, and T. Weber, "Temporal difference variational auto-encoder," in *Proceedings of International Conference on Learning Representation*, 2019.

[19] T. Gui, Q. Zhang, L. Zhao, Y. Lin, M. Peng, J. Gong, and X. Huang, "Long Short-term Memory with Dynamic Skip Connections," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2019, 6481–8.

[20] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville, "PixelVAE: A Latent Variable Model for Natural Images," in *Proceedings of International Conference on Learning Representations*, 2017.

[21] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning Latent Dynamics for Planning from Pixels," in *Proceedings of International Conference on Machine Learning*, Vol. 97, 2019, 2555–65.

[22] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, 9(8), 1997, 1735–80.

[23] N. R. Ke, A. Singh, A. Touati, A. Goyal, Y. Bengio, D. Parikh, and D. Batra, "Learning Dynamics Model in Reinforcement Learning by Incorporating the Long Term Future," *arXiv preprint:1903.01599*, 2019.

[24] T. Kim, S. Ahn, and Y. Bengio, "Variational Temporal Abstraction," *Advances in Neural Information Processing Systems*, 32, 2019, 11570–9.

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] D. P. Kingma and M. Welling, "Auto-encoding Variational Bayes," in *Proceedings of International Conference on Learning Representations*, 2014.

[27] R. Liu and A. Olshevsky, "Temporal Difference Learning as Gradient Splitting," in *Proceedings of International Conference on Machine Learning*, 2021, 6905–13.

[28] Y. Liu, P. Ramachandran, Q. Liu, and J. Peng, "Stein Variational Policy Gradient," *arXiv preprint arXiv:1704.02399*, 2017.

[29] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning Word Vectors for Sentiment Analysis," in *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2011, 142–50.

[30] L. van der Maaten and G. E. Hinton, "Visualizing Data Using t-SNE," *Journal of Machine Learning Research*, 9, 2008, 2579–605.

[31] T. Mikolov and G. Zweig, "Context Dependent Recurrent Neural Network Language Model," in *Proceedings of IEEE Spoken Language Technology Workshop*, 2012, 234–9.

[32] S. Mohamed and D. J. Rezende, "Variational Information Maximization for Intrinsically Motivated Reinforcement Learning," in *Neural Information Processing Systems*, 2015, 2125–33.

[33] J. D. Co-Reyes, Y. Liu, A. Gupta, B. Eysenbach, P. Abbeel, and S. Levine, "Self-consistent Trajectory Autoencoder: Hierarchical Reinforcement Learning with Trajectory Embeddings," in *Proceedings of International Conference on Machine Learning*, 2018, 1009–18.

[34] A. C. Rodriguez, R. Parr, and D. Koller, "Reinforcement Learning using Approximate Belief States," in *Advances in Neural Information Processing Systems*, 2000, 1036–42.

[35] D. Serdyuk, R. N. Ke, A. Sordoni, C. Pal, and Y. Bengio, "Twin Networks: Using the Future as a Regularizer," in *Proceedings of International Conference on Learning Representations*, 2018.

[36] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, 104(1), 2016, 148–75.

[37] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine learning*, 3(1), 1988, 9–44.

[38] Y. Tu, M.-W. Mak, and J.-T. Chien, "Variational Domain Adversarial Learning for Speaker Verification.," in *Proceedings of Annual Conference of International Speech Communication Association*, 2019, 4315–9.

[39] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," in *Proceedings of International Conference on Machine Learning*, 2016, 1747–56.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Advances in Neural Information Processing Systems*, 2017.

[41] H.-P. Wang, W.-H. Peng, and W.-J. Ko, "Learning Priors for Adversarial Autoencoders," *APSIPA Transactions on Signal and Information Processing*, 9(e4), 2020, 1–11.

[42] S. Watanabe and J.-T. Chien, *Bayesian Speech and Language Processing*, Cambridge University Press, 2015.

[43] S. Watanabe and A. Nakamura, "Bayesian Approaches to Acoustic Modeling: A Review," *APSIPA Transactions on Signal and Information Processing*, 1(e5), 2012, 1–11.

[44] C. Yildiz, M. Heinonen, and H. Lahdesmaki, "ODE2VAE: Deep Generative Second Order ODEs with Bayesian Neural Networks," in *Advances in Neural Information Processing Systems*, 2019, 13412–21.