Learning to Embed Probabilistic Structures Between Deterministic Chaos and Random Process in a Variational Bayes Predictive-Coding RNN

Ahmadreza Ahmadi[1, 2] **Jun Tani**[*1]

[1]Okinawa Institute of Science and Technology, Okinawa, Japan 904-0495.

[2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, 305-701, Republic of Korea.

Keywords: Recurrent neural network, variational Bayes, predictive coding, generative model, inference model

Abstract

This study introduces a stochastic predictive-coding RNN model that can learn to extract probabilistic structures hidden in fluctuating temporal patterns by dynamically

[*]Corresponding author

changing the uncertainty of latent variables. The learning process of the model involves maximizing the lower bound on the marginal likelihood of the sequential data, which consists of two terms. The first one is the expectation of prediction errors and the second one is the divergence of the prior and the approximated posterior. The main focus in the current study is to examine how weighting of the second term during learning affects the way of internally representing the uncertainty hidden in the sequence data. The simulation experiment on learning a simple probabilistic finite state machine demonstrates that the estimation of uncertainty in the latent variable approaches zero at each time step and that the network imitates the probabilistic structure of the target sequences by developing deterministic chaos in the case of the high weighting. On the contrary, in the case of the low weighting, the estimate of uncertainty increases significantly because of developing a random process in the network. The analysis shows that generalization in learning is most successful between these two extremes. Qualitatively, the same property has been observed in a trail of learning more complex sequence data consisting of probabilistic transitions between a set of hand-drawn primitive patterns using the model extended with hierarchy.

1 Introduction

Recently, predictive coding has attracted considerable attention in cognitive neuroscience, as a neuroscientific model accounting for possible neuronal mechanisms in learning, recognition, and generation of behaviors in a unified manner (Rao and Ballard, 1999; Lee and Mumford, 2003; K. Friston, 2018; Clark, 2015). The principle of predic-

tive coding suggests that first, agents are able to predict perceived outcomes through a top-down internal process. Then, prediction errors are generated by comparing the real perception and the predicted one. In addition, prediction errors are propagated through the bottom-up connections to update internal states in the direction of minimizing errors by which recognition of perceptual inputs can be achieved. Moreover, learning can be achieved by updating connectivity weights as well, in the direction of minimizing error.

Tani and colleagues (Tani and Nolfi, 1999; Tani and Ito, 2003; Tani, Ito, and Sugita, 2004) investigated the predictive-coding framework extended especially in the time domain for the purpose of providing learning capabilities to robots. They used recurrent neural network (RNN) models (Elman, 1990; Jordan, 1997; Hochreiter and Schmidhuber, 1997) because RNN models are capable of capturing temporal dependencies hidden in sequential patterns through learning. However, their ability can be limited when the learning involves with uncertainty or fluctuation in temporal patterns, which often becomes crucial when they are used for real robots. This limitation mainly originates from the fact that conventional RNN models can predict next perceptual inputs only in a deterministic fashion.

Active (hierarchical Bayesian) inference proposed by Friston (K. Friston, 2010; K. J. Friston, Daunizeau, Kilner, and Kiebel, 2010) is considered to remedy this situation. One of the essential ideas of this framework is to predict the precision or variance, as well as the mean of the input, instead of predicting its exact value deterministically. Friston formulated this idea into the so-called free energy minimization principle (FEMP) (K. Friston, 2005). Inspired by the FEMP, Murata et al. (Murata, Namikawa, Arie, Sugano, and Tani, 2013; Murata et al., 2017) proposed a predictive-

coding stochastic RNN model (S-RNN). In this model, the mean and standard deviation (SD) of the target value are estimated in the output of the network, meaning that uncertainty is captured in the output layer. The learning process of S-RNN involves not only optimization of connectivity weights, but also latent states at the initial step for each training sequence by means of error back-propagation through time (Werbos, 1974; Rumelhart, Hinton, and Williams, 1985). The authors alter the degree of the initial state dependency in learning to see how it affects internal information processing of the S-RNN. They found that the degree of deterministicity against stochasticity developed in the network can be arbitrated by the initial state dependency.

However, one major drawback to this model is that the standard deviation is not estimated for latent variables, which means that latent variables are deterministically determined. Consequently, the capability of the model to learn hidden probabilistic structures in target temporal patterns is significantly impaired. One solution is to define latent variables as random variables. However, one major difficulty with introducing stochasticity in the latent variables of neural networks is that error cannot be propagated through random variables. This difficulty was overcome by means of reparameterization, in which an auxiliary random variable was introduced, and the mean and SD of the latent variables were computed deterministically (Kingma and Welling, 2013). By this method, the error could be propagated through the mean and SD. As a matter of fact, Kingam and Welling proposed a Variational Autoencoder (VAE) model that used reparameterization in the approximated posterior of their model. Optimization was done using the variational Bayes (VB) approach in order to maximize a variational lower bound on the marginal likelihood of the data, and the prior was sampled from a

standard normal Gaussian.

Not surprisingly, various RNN models have been proposed based on the VAE. First variational Bayes RNNs also sampled the prior from a standard normal Gaussian at each time step (Fabius and van Amersfoort, 2014; Bayer and Osendorfer, 2014). Later, Chung et al. proposed a model (VRNN) with conditional prior distribution in order to consider dependencies between latent variables across time steps. Since then, there have been various attempts to modify the approximated posterior. For example, some models proposed approximated posteriors that had more similar structures to the true posterior by considering future dependencies on sequential data (Fraccaro, Sønderby, Paquet, and Winther, 2016; Goyal, Sordoni, Côté, Ke, and Bengio, 2017; Shabanian, Arpit, Trischler, and Bengio, 2017). Some models applied different approaches to force latent variables in the approximated posteriors to extract meaningful information (Bowman et al., 2015; Karl, Soelch, Bayer, and van der Smagt, 2016; Goyal et al., 2017). The current study brings the idea of the predictive coding to variational Bayes RNN models for the purpose of approximating the posterior through an iterative search by means of the error regression or the active inference in Friston's term, because such models should be more biologically plausible.

In a similar attempt to the current study, (Ahmadi and Tani, 2017a) also proposed a variational Bayes predictive-coding RNN (VBP-RNN). In their model, the RNN and latent variables are mixed, and the prior is sampled from a standard normal Gaussian at each time step. As with many variational Bayes RNNs, the model received an external input to generate the current output during the training phase, so the generation is not completely done in a top-down process. This means that the training process of the

model is not perfectly consistent with the framework of predictive coding.

In considering this problem, the current paper proposes another version of Variational Bayes predictive-coding RNN (VP-RNN) in which the deterministic state of the RNN and the latent variables are separated. The prior distribution is computed using conditional prior parameterization similar to (Chung et al., 2015). The model does not receive external inputs even during the training phase, and information about bottom-up perceptual inputs is given in terms of its prediction errors for latent variables. We apply predictive coding along with a variational Bayes approach in the generative model and approximated posterior by maximizing a variational lower bound on the marginal likelihood, which is analogous to the FEMP(K. Friston, 2005), using backpropagation through time. The approximated posterior considers future dependencies on sequential data in terms of prediction errors and is obtained through an iterative search by means of error regression.

The main motivation of the current study is to clarify how uncertainty or probabilistic structure hidden in fluctuating temporal patterns can be extracted and then internally represented in latent variables of the proposed RNN model. For this purpose, we introduce a parameter that weights minimization of the divergence between the posterior and the prior against that of the prediction error in computation of the variational lower bound. We investigate how this parameter, later referred to as the meta-prior, influences development of different types of information processing in the model. To this end, we first conduct a simulation experiment using a simple probabilistic finite state machine (PFSM) and observe how different settings of the meta-prior affect representation of uncertainty in the latent state of the model. Next, we examine how different represen-

tations of latent states in the model can lead to development of purely deterministic dynamics, random processes, or something between these two extremes. Analysis of simulation results indicates that most generalization in learning is achieved between these extremes. For the purpose of examining the same problem, but also involving more complexity, such as related to hierarchical information processing, the current paper describes a further experiment evaluating the model extended with a multiple timescale property in order to deal with fluctuating temporal patterns that consist of a set of hand-drawn primitives with probabilistic transitions among them. The simulation result shows that the extended model exhibits qualitatively the same property in extracting latent probabilistic structure in such compositionally organized sequence data.

The first contribution of the current study is to propose a novel stochastic RNN model that is consistent with the predictive-coding framework and can account for three processes of learning, generating, and recognizing temporal patterns under the same principle. To this end, the current study benefits from recent developments in variational Bayes RNN models as well. The second contribution is to show that in some cases uncertainty in temporal patterns can be internally represented either more deterministically by chaos or more probabilistically by a random process in which the balance between these two can be arbitrated by changing a system parameter of the model. This could contribute to bridging the gap between the two approaches, one based on deterministic dynamics and the other based on probabilistic modeling.

2 Model

This section first introduces the proposed generative model graph and its related equations. The generative model generates prediction sequences using top-down connections without external information. Then, the proposed inference model is described in detail. Because the true posterior is intractable to computation, the inference model is proposed to approximate the true posterior. The design of the inference model is based on the predictive-coding framework and recent developments of variational Bayes models. Later, the lower bound equation and the learning procedure are explained. Network parameters are optimized by maximizing the lower bound on the marginal likelihood of sequential data. Next, the error regression scheme, used during the test phase and designed on the predictive-coding framework, is introduced. During error regression, the approximated posterior is obtained via an iterative process to minimize errors between the predicted and test sequences inside a temporal window of an immediate past. Finally, connections and relations of the proposed model to previous variational Bayes RNNs are described.

2.1 Generative Model

We define a VP-RNN as a generative model P_θ by inserting a sequence of stochastic latent variables into an RNN model. The generative model is illustrated in Figure 1(A) by black lines, and for a sequence of observations $X_{1:T} = (X_1, X_2, ..., X_T)$, it factorizes

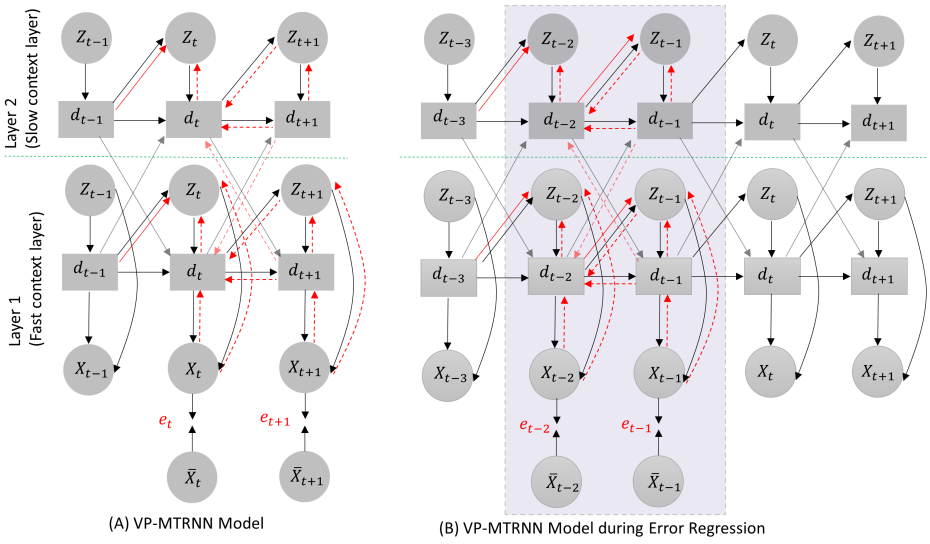

Figure 1: This figure shows: (A) generative and inference models of the VP-MTRNN; and (B) the error regression scheme used during the test phases. Black lines represent the generative model and red lines show the inference model. Dashed and solid red lines show the BBTT and feed-forward computations, respectively. The gray area in (B) is the temporal window of the immediate past in which latent states are modified to maximize the lower bound. In the inference model in (B), only connections for computation of the latent state Z at the time step t are shown.

as:

$$\begin{aligned} P_\theta(X_{1:T}, Z_{1:T}, d_{1:T}|Z_0, d_0) &= P_{\theta_X}(X_{1:T} \mid d_{1:T}, Z_{1:T}) P_{\theta_Z}(Z_{1:T} \mid d_{1:T}, Z_0) P_{\theta_d}(d_{1:T} \mid Z_{1:T}, d_0) \\ &= \prod_{t=1}^{T} P_{\theta_X}(X_t \mid d_t, Z_t) P_{\theta_Z}(Z_t \mid d_{t-1}) P_{\theta_d}(d_t \mid d_{t-1}, Z_t) \end{aligned} \tag{1}$$

where Z and d are the stochastic and deterministic latent states, respectively. The initial values Z_0 and d_0 of the latent states are set to 0 in our experiments. Generative model parameters are denoted as $\theta = \{\theta_X, \theta_Z, \theta_d\}$. Figure 1(A) shows connections for computing Z_t in the inference model. The latent state d_t is recursively computed using an auto-regressive model $d_t = f_{\theta_d}(d_{t-1}, Z_t)$. An MTRNN (Yamashita and Tani, 2008) is used as the auto-regressive model, but the proposed model can use any type of RNN, such as the LSTM or the GRU, instead, which is why the proposed model is usually

called a VP-RNN in this paper. MTRNN is a type of RNN model that uses different time constant values for different units. The internal dynamic of an MTRNN model is computed as:

$$h_{t,i} = (1 - \frac{1}{\tau_i})h_{t-1,i} + \frac{1}{\tau_i}(\sum_{j=1}^{N_d} w_{ij}^{dd} d_{t-1,j} + \sum_{k=1}^{N_Z} w_{ik}^{dZ} Z_{t,k} + \sum_{j=1}^{N_{d^{(H)}}} w_{ij}^{dd^{(H)}} d_{t-1,j}^{(H)} + \sum_{j=1}^{N_{d^{(L)}}} w_{ij}^{dd^{(L)}} d_{t-1,j}^{(L)} + b_i^h) \tag{2}$$

where $h_{t,i}$ is the internal state value of the i_{th} d unit at time t, w_{ij}^{dd} is the connectivity weight from the i_{th} d unit to the j_{th} d unit, w_{ik}^{dZ} is the connectivity weight from the i_{th} d unit to the k_{th} Z unit, b_i^h is the bias of the i_{th} neural unit, N_d is the number of the d units, N_Z is the number of Z units, and τ is the time constant. The letters "H" and "L" represent higher and lower layers, respectively. The lowest layer sub-network with the smallest time constant is referred to as a fast context layer, while the highest layer sub-network with the largest time constant is referred to as a slow context layer. If there are sub-networks between the highest and lowest sub-networks, the layers are referred as middle context layers. When the slow layer internal states are computed, the higher layer does not exist, and the lower layer does not exist for computation of the fast layer. The d states are computed as $d_{t,i} = \tanh(h_{t,i})$. The hierarchical architecture of a VP-MTRNN with two context layers are shown in Figure 1(A). Unlike many other variational Bayes RNN models, we do not provide any external inputs to the generative model. If we were to do so, prediction outputs might be affected mostly by mapping from current actual observations to future ones. This is not consistent with the predictive-coding framework, which posits that the brain is continually making predictions of incoming sensory stimuli in a top-down process (Rao and Ballard, 1999;

K. Friston, 2005), and prediction errors are generated by comparing these predictions and actual observations. During generations of generative models, there are no target sequences, so there are no prediction errors. However, prediction errors are generated during the inference as well as training process of the model while back-propagated through levels and time leading to an update of the prediction.

Marginal likelihood or evidence can be obtained as

$$P_\theta(X_{1:T}|Z_{1:T}, d_{1:T}) = \iint \prod_{t=1}^{T} [P_{\theta_X}(X_t \mid d_t, Z_t) P_{\theta_Z}(Z_t \mid d_{t-1}) P_{\theta_d}(d_t \mid d_{t-1}, Z_t)] dZ_{1:T}\, dd_{1:T} \tag{3}$$

For convenience, $d_{1:T}$ computed by f_{θ_d} (the MTRNN model) are denoted as $\tilde{d}_{1:T}$ from now on in this paper. $d_{1:T}$ follows a Dirac-delta distribution centered at $\tilde{d}_{1:T}$, meaning that the distribution has values only for $\tilde{d}_{1:T}$ and it is zero for others

$$P_{\theta_d}(d_t \mid d_{t-1}, Z_t) = \delta(d_t - \tilde{d}_t) \tag{4}$$

So, Eq. 3 can be rewritten as

$$\begin{aligned} P_\theta(X_{1:T}|Z_{1:T}, d_{1:T}) &= \int \prod_{t=1}^{T} [P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})] dZ_{1:T} \\ &= \prod_{t=1}^{T} \left[\int P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1}) dZ_t \right] \end{aligned} \tag{5}$$

The prior distribution is obtained using a non-linear transformation of the previous hidden state of the forward network. This type of prior was used in (Chung et al., 2015), and it outperformed a standard Gaussian prior used in STORN (Bayer and Osendorfer, 2014). We chose the prior to be a Gaussian with diagonal covariance matrix, in which

the mean and logarithm of the standard deviation are parameterized as:

$$P_{\theta_Z}(Z_t \mid d_{t-1}) = \mathcal{N}(Z_t; \mu_t^{(p)}, \sigma_t^{(p)}) \quad \text{where} \quad [\mu_t^{(p)}, \log \sigma_t^{(p)}] = f^{(p)}(d_{t-1}) \tag{6}$$

where $f^{(p)}$ denotes a one layer feed-forward neural network. We use the reparameterization trick (Kingma and Welling, 2013) such that the latent Z in both posterior and prior are reparameterized as $Z = \mu + \sigma * \varepsilon$, where ε is sampled from $\mathcal{N}(0, I)$. In this study, $p_{\theta_X}(X_t \mid d_t, Z_t)$ is obtained by a one layer feed-forward model $f^{(x)}$.

2.2 Inference Model

Based on the generative model, the true posterior distribution of Z_t depends on $X_{t:T}$, which can be verified using d-separation (Geiger, Verma, and Pearl, 1990). Computing the true posterior is intractable, so an inference model is designed to compute an approximated posterior using the framework of the predictive coding (Rao and Ballard, 1999; K. Friston, 2005; Rao and Sejnowski, 2000; Lee and Mumford, 2003; Tani and Ito, 2003). The inference model is shown in Figure 1(A) by red lines. The inference model receives the RNN states d_{t-1}, shown by a solid line, by means of a feed-forward neural network, whereas future observations from time steps T to t are brought to the current latent state Z_t in terms of prediction errors through back-propagation through time (BBTT), shown by dashed lines. More specifically, in the predictive-coding-inspired generative model, predictions are conveyed using top-down connections. Then, the predictions are compared with actual observations and prediction errors are generated, and the KL divergence between the prior and approximated posterior distributions is also

computed. Next, the lower bound is obtained by summation of the prediction errors and the KL divergence. Last, the networks parameters and the posterior are updated by maximizing the lower bound. The lower bound and KL divergence equations are described in the next section. The approximated posterior is obtained as

$$q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) = \mathcal{N}(Z_t; \mu_t^{(q)}, \sigma_t^{(q)}) \quad \text{where} \quad [\mu_t^{(q)}, \log \sigma_t^{(q)}] = f^{(q)}(\tilde{d}_{t-1}, A_t^{(q)}) \quad (7)$$

where $f^{(q)}$, e, and ϕ denote a one layer feed-forward neural network, the prediction error, and the posterior parameters, respectively. $A_t^{(q)}$ is computed using BBTT, which exploits prediction errors $e_{t:T}$. $A_t^{(q)}$ can be obtained using gradient ascent as

$$A_t^{(q)} = A_t^{(q)} + \alpha \frac{\partial L}{\partial A_t^{(q)}} \quad (8)$$

where α and L are the learning rate and the lower bound, respectively. For the sake of simplicity, the simple computation of $A_t^{(q)}$ is shown here using the gradient ascent algorithm. However, we use the ADAM optimizer (Kingma and Ba, 2014) in the experiments. After μ values are computed in both the prior and posterior, they are given to a $\tanh$ in order to set their values between -1 to 1. This does not apply to $\log \sigma$. $A_t^{(q)}$ values are multiplied by the identity matrix in $f^{(q)}$. Detailed computations of $A_t^{(q)}$ for both $\mu_t^{(q)}$ and $\log \sigma_t^{(q)}$ are given in Appendix A

2.3 Learning Process

We add $\frac{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})}{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})}$ to Eq. 5 and take $\log$ from both sides as

$$\begin{aligned}\log P_\theta(X_{1:T} \mid Z_{1:T}, d_{1:T}) &= \log \prod_{t=1}^{T} [\int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \frac{P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})}{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})} P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) dZ_t] \\ &= \sum_{t=1}^{T} \log [\int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \frac{P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})}{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})} P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) dZ_t]\end{aligned} \tag{9}$$

Since logarithm is a concave function, we can apply the Jensen's inequality ($E[f(X)] \leq f(E[X])$)

$$\begin{aligned}\log P_\theta(X_{1:T} \mid Z_{1:T}, d_{1:T}) &= \sum_{t=1}^{T} \log [\int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \frac{P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})}{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})} P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) dZ_t] \\ &\geq \underbrace{\sum_{t=1}^{T} \int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \log [\frac{P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})}{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})} P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t)] dZ_t}_{Variational\ Evidence\ Lower\ Bound\ (L(\theta,\phi))}\end{aligned} \tag{10}$$

Now, the Variational Evidence Lower Bound (ELBO) can be maximized instead of the logarithm of the marginal likelihood $P_\theta(X_{1:T} \mid Z_{1:T}, d_{1:T})$ in order to optimize the parameters of the generative model and approximated posterior. This formula for maximizing the lower bound is equivalent to the principle of free energy minimization provided by Friston (K. Friston, 2005). $L(\theta, \phi)$ can be rewritten as

$$\begin{aligned}L(\theta, \phi) &= \sum_{t=1}^{T} (\int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \log P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t) dZ_t - \int q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \log \frac{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})}{P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})} dZ_t) \\ &= \sum_{t=1}^{T} (E_{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})}[\log P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t)] - KL[q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \mid\mid P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})])\end{aligned} \tag{11}$$

where, the first term on the right hand side is the expected log-likelihood under $q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})$, and the second term is the Kullback-Leibler (KL) divergence between the posterior and

prior distributions of the latent variables. Only summation over time is shown in this equation, but the lower bound will be also summed over the number of training samples, the dimension of X on the right side, and the dimension of Z on the left side. We divided the first term by the dimension of X and the second term by the dimension of Z during experiments. The KL divergence is computed analytically as:

$$KL[q_\phi(Z_t) \,||\, P_{\theta_Z}(Z_t)] = -\tfrac{1}{2}(1 + \log(\sigma_t^{(q)})^2 - \log(\sigma_t^{(p)})^2 + \tfrac{-(\mu_t^{(q)})^2 - (\sigma_t^{(q)})^2 + 2\mu_t^{(q)}\mu_t^{(p)} - (\mu_t^{(p)})^2}{(\sigma_t^{(p)})^2}) \quad (12)$$

The detailed derivation of the KL divergence is in Appendix B.

A parameter w before the KL divergence term is introduced in the ELBO. We assume that the dynamics of the model will change based on different values of w. If w is set to a small value, the dynamics of the prior and posterior tend to diverge. The parameters of prior are optimized through the KL divergence term, whereas parameters of the posterior are optimized through both prediction errors and the KL divergence terms. With a small w, the posterior is optimized mainly by the prediction errors and optimization is affected weakly by the KL term. The ELBO with w is written as

$$L(\theta, \phi) = \sum_{t=1}^{T}(E_{q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T})}[\log P_{\theta_X}(X_t \mid \tilde{d}_t, Z_t)] - w \cdot KL[q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) \,||\, P_{\theta_Z}(Z_t \mid \tilde{d}_{t-1})]) \quad (13)$$

Finding an adequate range of w at a beginning of an experiment depends on network parameters settings, data-set, and tasks. w was found empirically as 0.1 and 0.001 for the first and second experiments in this paper. In order to have a clear demonstration of the sensitivity of w throughout the current paper, we consider w equal to $\gamma\ W$, so we have "$\gamma\ W\ KL(q_\phi(Z)||P_{\theta_Z}(Z))$". The parameter W is called the meta-prior. Now,the

meta-prior W can be valued from 1.0 by setting the parameter γ to 0.1 and 0.001 in the first and second experiments, respectively. If the KL divergence term becomes dominant by choosing large γ values at the beginning of the training process, prediction errors cannot be propagated properly through the network parameters.

In the experiments, all parameters are optimized in order to to maximize the lower bound using ADAM (Kingma and Ba, 2014). We use the same parameter setting for the ADAM optimizer as the original paper: α = 0.001, β_1 = 0.9, and β_2 = 0.999 in the training phase. In both experiments, the number of latent units Z were 10 times smaller than number of deterministic units d. In order to examine the sensitivity of model dynamics to W, different sets of W are considered in the experiments.

2.4 Error Regression Scheme

The trained network can generate unseen data during the test phase by means of the error regression scheme that was previously used in predictive-coding models (Tani and Ito, 2003; Ahmadi and Tani, 2017b). Although error regression is performed by updating estimates in an iterative procedure like the training process, all network parameters are fixed during error regression except $A^{(q)}$. Values of $A^{(q)}$ are updated using BBTT inside a temporal window of the immediate past. During error regression, network outputs $X_{t-WS:t-1}$ are generated through top-down connections by providing latent states $Z_{t-WS:t-1}$, where WS is the temporal window size. Then, the prediction errors $e_{t-WS:t-1}$ are computed by comparing the outputs $X_{t-WS:t-1}$ with targets $\bar{X}_{t-WS:t-1}$. Furthermore, the KL divergence between prior and posterior distributions inside the temporal window is computed, and prediction errors and the KL divergence

are summed to obtain the lower bound. Finally, the variables $A^{(q)}_{t-WS:t-1}$ are updated using the ADAM optimizer. These steps are done in an iterative process to maximize the lower bound inside the window. After the last step, the latent state Z_t is sampled from the prior, which is a function of d_{t-1}, and X_t is generated. Sampling of Z can be continued for future time steps in order to generate multiple look-ahead output predictions X_{t+1}, X_{t+2}, and so on. The process of error regression proceeds to the next step, t by sliding the window one step forward, and the cycle repeats itself.

The schematic of the error regression scheme is shown in Figure 1(B) for a VP-RNN. The gray area shows the temporal window in which all formal states are overwritten from the time step $t-2$ to $t-1$ by updating $A^{(q)}_{t-2:t-1}$ through BBTT. Current and future predictions $X_{t:t+1}$ are computed after the window by sampling $Z_{t:t+1}$ from the prior.

2.5 The relationship between related work on variational Bayes RNNs and the present work

RNNs are widely used to model temporal sequences due to their ability to capture long dependencies in data. However, a deterministic RNN $d_t = f(d_{t-1}, \bar{X}_{t-1})$ can present problems when modeling sequences with a high signal-to-noise ratio (Chung et al., 2015). In an attempt to solve this problem, Bayer and Osendorfer (Bayer and Osendorfer, 2014) introduced a model called STORN by inserting a set of independent latent variables (sampled from a fixed distribution) into the RNN model. Later, the VRNN model was proposed using conditional prior parameterization (Chung et al., 2015). In their model, the prior distribution is obtained using a non-linear transformation of the

previous hidden state of the forward network as $[\mu_t^{(p)}, \log(\sigma_t^{(p)})] = f^{(p)}(d_{t-1})$. VRNN outperformed STORN by using this type of conditional prior. However, VRNN's posterior does not receive future observations in the sequence, which is different from the structure of the true posterior. Later, this issue was considered by using two RNNs, one forward and one backward. The backward RNN was used in the posterior to transfer future observations for the current prediction (Fraccaro et al., 2016; Goyal et al., 2017; Shabanian et al., 2017). Instead of using another RNN, taking inspiration from predictive coding, we propose a model with an approximated posterior that brings future observations to the current latent state Z_t in terms of prediction errors $e_{t:T}$ by means of BBTT. The prior distribution is obtained in the same method as (Chung et al., 2015).

Recent studies of generative models show that extracting a meaningful latent representation is difficult because of using a powerful auto-regressive decoder. The RNN ignores the latent variables and captures most of the entropy in the data distribution (Goyal et al., 2017). Many researchers have addressed this issue and tried to solve it by either weakening the auto-regressive decoder or by annealing the KL divergence term during training (Bowman et al., 2015; Karl et al., 2016; Kingma et al., 2016; Chen et al., 2016; Zhao, Zhao, and Eskenazi, 2017). Authors of Z-forcing also proposed an auxiliary cost on the lower bound, which forces the latent variables to reconstruct the state of the backward RNN (Goyal et al., 2017). This method forces the latent variables to have meaningful representation. In the proposed model, external input is not provided for either the prior or the posterior. This means that the posterior needs to receive all information through the latent variables first, and transfer them to the RNN. This way, the RNN cannot ignore the latent variables, and the latent variables are forced to extract

meaningful representation.

3 Experiments

We conducted simulation experiments to determine how learning in the proposed model depends on a meta-prior W. The first experiment examined how the proposed model could learn to extract latent probabilistic structure from discrete (0 or 1) sequence data generated from a simple probabilistic finite state machine (PFSM) under different settings of the parameter W. The purpose of this relatively simple experiment was to examine precisely the underlying mechanism of the VP-RNN in embedding probabilistic data structures. In the second experiment, a more complex situation was considered where the model was required to extract latent probabilistic structures from observed continuous sequence data (movement trajectories). For this purpose, trajectory data were generated by considering probabilistic switching of primitive movement patterns based on another PFSM in which each primitive was generated with certain fluctuations in amplitude, velocity, and shape. We examined how the proposed model could learn such temporal patterns by reconstructing either deterministic or probabilistic structure, depending on the meta-prior W used during training, as well as how changing this parameter affected the generalization capability of the model in generating fluctuating temporal patterns.

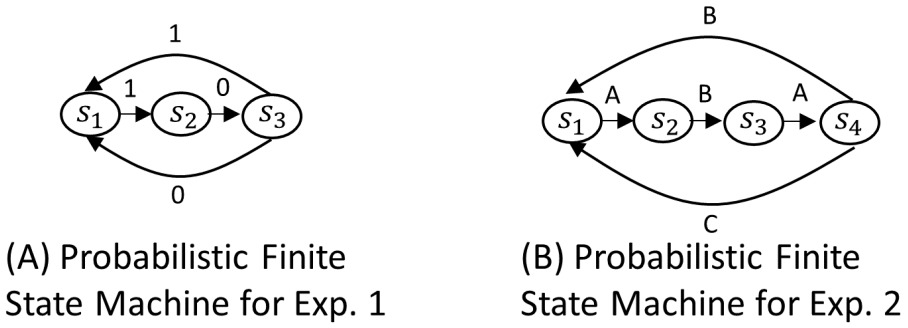

Figure 2: This figure shows probabilistic finite state machines used to generate training patterns for VP-RNNs in the first (A) and second experiments (B).

3.1 Experiment 1

A PSFM was used as the target generator, as shown in Figure 2(A). Transitions from s_1 to s_2 and s_2 to s_3 were deterministically determined with the output of 1 and 0, respectively. However, the transitions from s_3 to s_1 were randomly sampled with 70% and 30% probabilities of the output of 1 and 0, respectively. 10 target patterns , 24 time steps for each, were generated by PFSM and provided to the VP-RNN as training data. Training was conducted with the meta-prior W set to 1.0, 0.5, 0.25, 0.15, 0.1, 0.01, and 0.001. Each model had only one context layer consisting of 10 d units and a single Z unit. The time constant τ for all d units was set to 2.0.

After training for $500,000$ epochs, the mean and the $\log$ standard deviation of the latent state for each target pattern at the first time step ($\mu_1^{(q)}$ and $\log(\sigma_1^{(q)})$) were given to the generative models in order to regenerate the target patterns. This means that Z_1, inferred by the posterior during training, was given to the generative model at the first step and then X_1 was generated. Furthermore, the latent states $Z_{2:T}$ were computed by the prior, and $X_{2:T}$ were generated. In order to fit the discrete representation of PFSM,

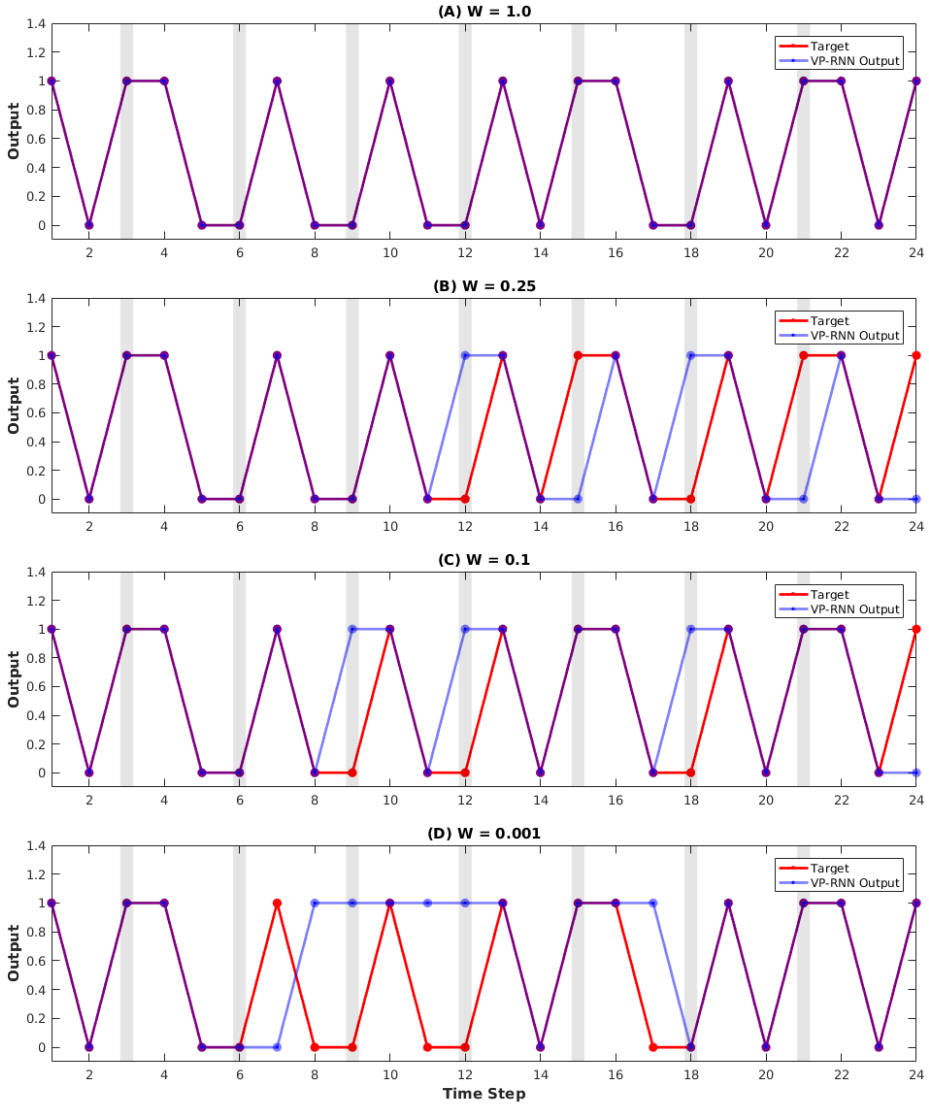

Figure 3: A typical comparison between a teaching target sequence pattern and reconstruction outputs generated by the VP-RNN models trained with different values for W. Gray bars show the time steps corresponding to uncertain states.

a regenerated output less than 0.5 was assigned to 0 and that equal or larger than 0.5 is assigned to 1. Figure 3 compares one target sequence pattern and its corresponding regeneration by the VP-RNN models trained with different values for W. From first to last rows, the target pattern and regenerated patterns with W set to 1.0, 0.25, 0.1, and 0.01 are shown, respectively. When W set to 1.0, the target sequence pattern was

completely regenerated for all steps. When W is equal to 0.25, 0.1, and 0.001, target and regenerated patterns begin to diverge at the $11th$, $8th$, and $6th$ steps, respectively. When W was set to 0.001, the deterministic transition rules were also broken. For example, the generated output must be 0 at $11th$ step but it was 1. An average diverging step (ADS) was computed for all target sequences by taking the average step at which the target sequence pattern and the regenerated pattern diverged. For example, at time step t, if the regenerated output of a target sequence was 1 while the value of the target was 0, the diverging step was equal to t. The larger ADS is, the more similar target and regenerated patterns are. ADS results for all VP-RNNs are demonstrated in Table 1 in which ADS decreases as W decreases. In order to compute ADS, simulations were run 10 times for each target pattern and the means (of all patterns and simulation runs) were computed. It was observed that ADS results did not change for VP-RNN with W set to 1.0 in any simulation run. On the contrary, ADS results changed significantly in each run for VP-RNN with W set to 0.001. These observations, along with ADS values on the table, suggest that the VP-RNN trained with W set to 1.0 develops a pure deterministic dynamic, while that trained with W set to 0.001 develops an extremely stochastic dynamic. Dynamics developed with W set to 1.0 are indeed deterministic, since the target pattern could be regenerated deterministically by providing Z_1. The results suggests that the strength of top-down intention for regenerating a learned sequence pattern tends to decay as W decreases.

Target and regenerated output patterns and the mean $\mu^{(p)}$, and standard deviation $\sigma^{(p)}$ of the latent unit for VP-RNNs trained with W equal to 1.0 and 0.25 are illustrated in Figure 4. With $W = 1.0$, sigma approaches zero at each time step accounting for

Table 1: Average diverging step (ADS) and KL divergence (KL div.) between $P(\bar{X}_{t:t+11})$ and $P(X_{t:t+11})$ computed during training and test phases.

	Meta-Prior W						
	W=1.0	W=0.5	W=0.25	W=0.15	W=0.1	W=0.01	W=0.001
ADS	**22**	19	14	12	11	9	8
KL div. of Training Phase	**0.0158**	1.5837	1.5748	2.8404	4.9824	8.1597	18.5159
KL div. of Test Phase	5.0399	2.2762	**0.0684**	0.1195	0.1482	1.0679	5.6069

development of deterministic dynamics. With $W = 0.25$, sigma increases on uncertain states, on the gray bars in the figure, and decreases on deterministic states. These results suggest that the VP-RNN with W set to 0.25, is capable of detecting uncertain states in the target sequence by raising expectation of sigma. A figure showing meta-parameters set to 0.01 and 0.001 is also given in Appendix (Figure 10). Sigma activities for W set to 0.001 rise even in deterministic states at some steps, so the VP-RNN becomes a random process in this case.

We examined the capability of extracting latent probabilistic structure from the data for each VP-RNN by measuring the divergence between probability distributions of output sequences and the target using both the target regeneration and free generation approaches. The target regeneration approach is referred to the case when the Z_1, inferred by the posterior during training, is given to the generative model at the first step, and $Z_{2:T}$ are sampled by the prior model. The purpose of this phase is to regenerate target patterns, so T is set to the number of time steps in the training patterns. This approach was previously used in this paper for computing ADS. In the free generation approach, Z_1 is sampled randomly at the first time step and $Z_{2:T}$ are sampled by the prior model. The purpose of this approach is to evaluate the generalization capability of the generative model, and T is set freely by the experimenter. After using the target regeneration approach, the joint probability distributions of 12 discrete variables in

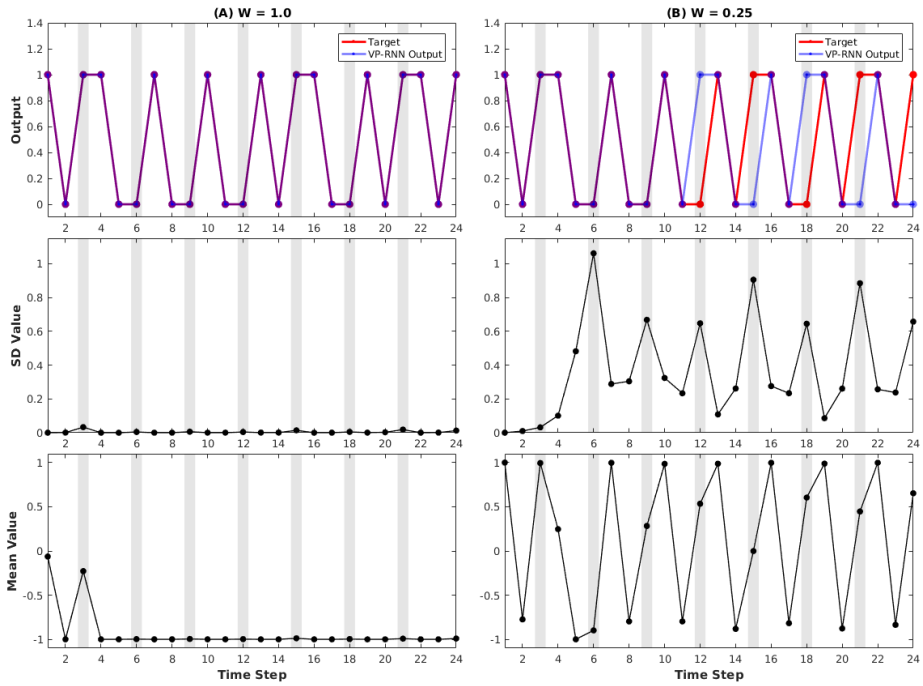

Figure 4: The target and the regenerated outputs, the mean $\mu^{(p)}$, and the standard deviation $\sigma^{(p)}$ of two VP-RNNs trained with meta-prior W set to 1.0 (A) and 0.25 (B). Gray bars show the time steps corresponding to uncertain states.

the target patterns $P(\bar{X}_{t:t+11})$ and regenerated outputs $P(X_{t:t+11})$ were first computed where t was set from 1 to 229, and then the KL divergence between $P(\bar{X}_{t:t+11})$ and $P(X_{t:t+11})$ was obtained. Because $X_{t:t+11}$ and $\bar{X}_{t:t+11}$ at each time step t could take values 1 or 0, there were 2^{12} possible outcomes for each of them. The joint probability distributions were actually computed by counting possible outcomes in the target and regenerated output sequences for time steps 1–229. By looking at the ADS obtained in the previous task, it was expected that the pure deterministic model would outperform other models because the regenerated output sequences were almost same as target sequences. The inverse was also expected to be true for the networks with W set to 0.001, which had smallest ADS. The KL divergence between target and regenerated patterns is given in Table 1, shown as "KL div. of Training Phase". The results conform to

the expectations since VP-RNNs with W equal to 1.0 and 0.001 show the smallest and largest KL divergences, respectively.

By means of the free generation approach, a sequence output was first generated for $50,000$ steps ($X_{1:50000}$) by each VP-RNN. Then, the joint probability distributions of 12 discrete variables in the generated output $P(X_{t:t+11})$ were computed where t was set from 1 to $49,989$. Furthermore, the KL divergence between the joint probability distributions of the target patterns $P(\bar{X}_{t:t+11})$ and the generated outputs $P(X_{t:t+11})$ were computed for each VP-RNN. Table 1 shows the results of computing the KL divergence of the test phase for each W. The KL-divergence is minimized between the two extreme W settings of 1.0 and 0.001. It can be said that the VP-RNN with W set to 1.0 regenerated target patterns well using strong top-down intentionality, although it did not extract probabilistic structures of data. However, the VP-RNN with W set to 0.25 was the most generalized network, showing the minimum KL divergence value between target and generated outputs during the test phase. Figure 5 illustrates the generated output, the mean $\mu^{(p)}$ of the Z unit, and standard deviation $\sigma^{(p)}$ of the Z unit for VP-RNNs trained with W equal to 1.0 and 0.25 from time steps $20,002$ to $20,040$. Time developments of sigma values in both cases are similar to those shown in Figure 4. It may be argued that in the case of W set to 0.25, the network was capable of extracting the latent probabilistic structure from the data by detecting uncertain states in the sequence adequately. The generated output, the mean $\mu^{(p)}$ of the Z unit, and standard deviation $\sigma^{(p)}$ of the Z unit for VP-RNNs trained with W equal to 0.01 and 0.001 from time steps $20,002$ to $20,040$ are shown in Appendix (Figure 11). Transition rules are mostly broken for the smaller meta-prior (0.001) and at some time steps for W equal to 0.01 as well, which

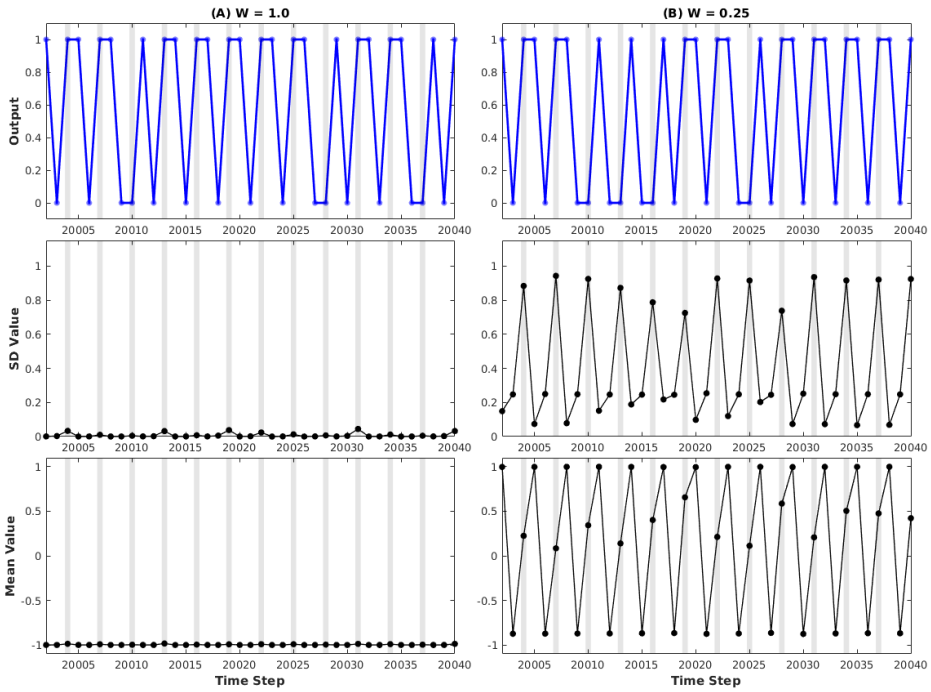

Figure 5: The generated output, the mean $\mu^{(p)}$, and the standard deviation $\sigma^{(p)}$ from time steps $20,002$ to $20,040$ of two VP-RNNs trained with the meta-prior W set to 1.0 (A) and 0.25 (B). Gray bars show the time steps corresponding to uncertain states.

means that the uncertainty is high even, in deterministic states.

It was noted that the deterministic network developed with W set to 1.0 generated non-periodic output patterns. This can be roughly seen in Figure 5(A). We assumed that deterministic chaos or transient chaos developed in this learning condition. For the purpose of confirming this assumption, the Lyapunov exponent was computed using the method in (Alligood, Sauer, and Yorke, 1996). Interestingly, the largest Lyapunov exponent was positive. We evaluated this by generating patterns (using the free generation approach) for 50,000 steps with and without shutting off inserted noise $\varepsilon_{1:50000}$. In both cases, the largest Lyapunov exponents were positive (around 0.1). These results accord with (Tani and Fukumura, 1995), in which the authors showed that a deterministic RNN can learn a PFSM by evolving toward chaos. The algorithm for computing Lyapunov

exponents is given in Appendix C.

3.2 Experiment 2

In this experiment, the VP-RNN was required to extract latent probabilistic structures from observed continuous sequence data (movement trajectories). These sequence data were generated in three stages as follows. First, a human generated $2-D$ patterns compositions from a set of different primitive patterns with certain fluctuation in amplitude, velocity, and shape using a tablet input device and by following the PFSM shown in Figure 2(B). Primitive patterns "A, B, C" were circles, rotated figure 8, and triangles similar to those depicted in the first row of Figure 7. The total number of "A, B, C" primitive patterns was 160 (4458 steps), and each primitive pattern was a different periodic pattern with 2 cycles. The transitions from s_4 to s_1 in the PFSM were randomly chosen by the human. We measured the conditional probabilities in the data after generation, and they were $P(B|ABA) = 0.275\%$ and $P(C|ABA) = 0.725\%$. Next, a target generator was built using the human-generated data for the purpose of producing training and testing patterns of the VP-RNN. An MTRNN was used as the target generator in this paper, which means that human-generated data were provided to an MTRNN as training data. After training, the MTRNN, closed-loop (feeding next step inputs with current step prediction outputs) patterns were generated while adding Gaussian noise with zero mean and with constant σ of 0.05 to the internal state of each context unit at each time step. This was done in order to make network output patterns stochastic while maintaining a certain probabilistic structure, not necessarily the same as the training pattern structures, in transitions between primitive patterns. More details and

implementations of this target generator MTRNN can be seen in (Ahmadi and Tani, 2017a). Because of inserting noise into the internal dynamics of the MTRNN, output patterns were noisier and fluctuated more than the human-expressed patterns, and patterns could have different numbers of cycles than 2. Finally, three groups of patterns were sampled from MTRNN-generated outputs, one consisting of 16 sequence patterns, each with a 400 step length for the training phase of the VP-RNN, one comprising 1 sequence patterns with a 6400 step length for the first test phase of the VP-RNN, and one consisting of 32 sequence patterns, each with a 400 step length for the second test phase of the VP-RNN. The main reason that the target generator was used instead of using human-generated trajectory data was because significantly larger amount of target data were needed for evaluating the proposed model than could be sampled using human generation. The target generator, MTRNN, can effortlessly generate as many diverse patterns as one wants.

Six VP-RNN models were trained with the meta-prior W set to 1.0, 0.5, 0.25, 0.15, 0.1, and 0.01. Each model had three context layers consisting of 80 d units and 8 Z units for the fast context (FC) layer, 40 d units and 4 Z units for the middle context (MC) layer, and 20 d units and 2 Z units for the slow context (SC) layer. Time constants of FC, MC, and SC units were set to 2, 4, and 8, respectively. Training ran for $250,000$ epochs in each case.

After training, the capability of the generative model to regenerate target patterns was evaluated using the target regeneration approach. For this purpose, target patterns were regenerated by providing the mean and $\log$ standard deviation of latent states for each target pattern at the first time step ($\mu_1^{(q)}$ and $\log(\sigma_1^{(q)})$) to the generative model.

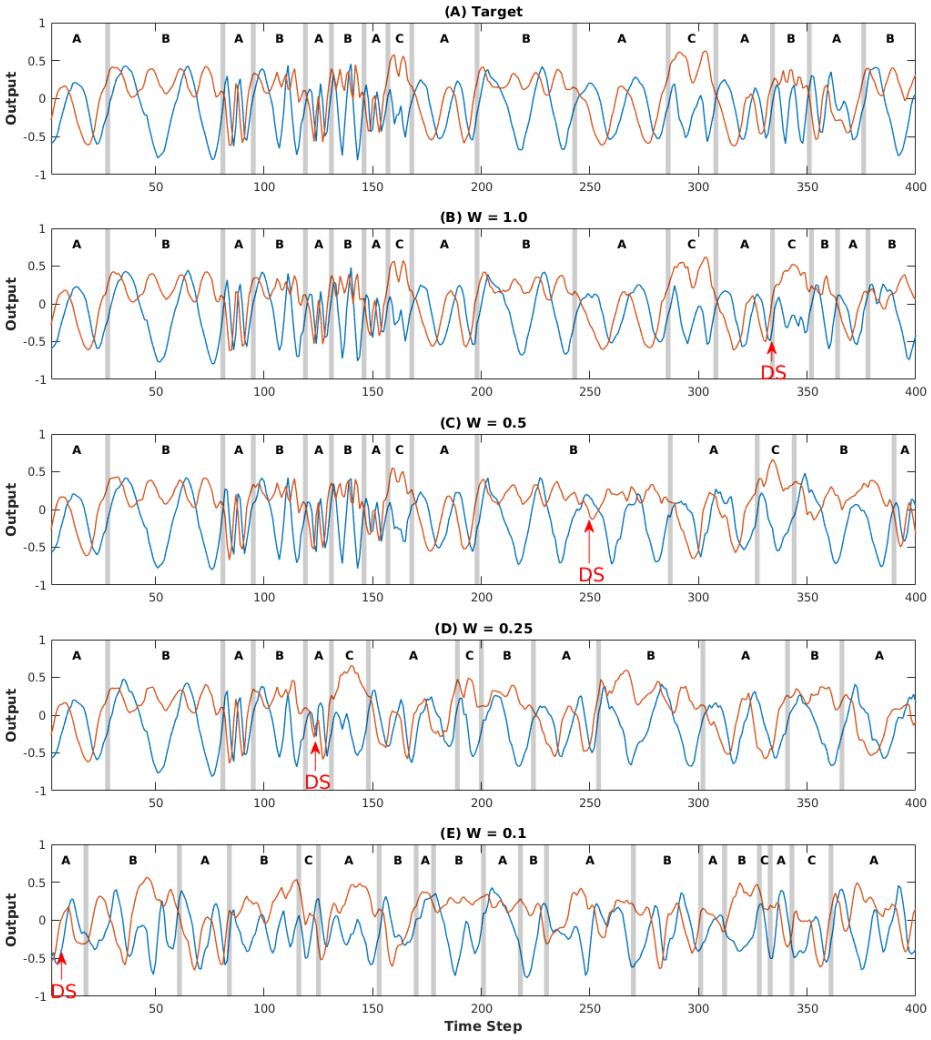

Figure 6: A typical comparison between a teaching target sequence pattern and the reconstruction output generated by VP-RNN models trained with different values for W. Capital letters above each plot denote the corresponding primitive type. Gray bars show the steps that transition between primitive patterns. Red arrows demonstrate the diverging steps (DS) in which regenerated outputs diverge from the target pattern.

This means that Z_1, inferred from the posterior during training, was given to the generative model at the first step and then X_1 was generated. Moreover, the latent states $Z_{2:T}$ were computed by the prior, and $X_{2:T}$ were generated. Figure 6 compares one target sequence pattern and its corresponding regeneration by the VP-RNNs trained with different values for W. The first row shows the target pattern and the second, third,

Table 2: Average diverging steps (ADS) computed during training phase.

	Meta-Prior W					
	W=1.0	W=0.5	W=0.25	W=0.15	W=0.1	W=0.01
ADS	**343**	229	103	17	4	1

and fourth rows show the regenerated patterns with W set to 1.0, 0.5, 0.25, and 0.1, respectively. Each pattern is associated with a sequence of labels, and transitions from one pattern to another are shown by gray bars. When $W = 1.0$, the divergence starts from the $343th$ step, and as W becomes smaller, the target and regenerated patterns begin to diverge earlier. When $W = 0.1$, the divergence started immediately after the onset. By looking at the target and generated patterns, one can see that primitive patterns in the target sequence are quite diverse, and such diverse primitive patterns can be regenerated by each VP-RNN until the divergence. After the divergence, the diversity is still maintained in generated primitive patterns. For instance in the target, the longest patterns A, B, and C have 43, 54, and 22 steps while the shortest ones have 10, 16, and 11steps, respectively. This shows that primitive patterns in the target have significantly diverse periodicity.

We also computed ADS in which the divergence was detected when the mean square error between the target and the generated pattern exceeded a threshold (0.01). The results are listed in Table 2. As expected, the VP-RNN trained with W set to 1.0 had the largest ADS (the most deterministic dynamics) while the one trained with W set to 0.01 had the smallest ADS (random process). For the purpose of computing ADS, simulations were run 10 times for each target pattern and the mean (of all patterns and simulation runs) were computed. It was observed that ADS results changed only slightly for the VP-RNN with W set to 1.0 in each simulation run, which is why this

Table 3: MSE between the test target and the 1-step to 5-steps ahead generated outputs.

	Meta-Prior W					
	W=1.0	W=0.5	W=0.25	W=0.15	W=0.1	W=0.01
1-step pred.	0.01011	0.00726	0.00418	0.00376	**0.00341**	0.00834052
2-steps pred.	0.01712	0.01272	**0.00907**	0.00918	0.01157	0.02285
3-steps pred.	0.02223	0.01831	**0.01399**	0.01527	0.02051	0.03779
4-steps pred.	0.02789	0.02325	**0.01894**	0.02117	0.03012	0.04969
5-steps pred.	0.03248	0.02739	**0.02344**	0.02696	0.03747	0.05776

network is said to have the most deterministic dynamics. More specifically, by providing intentional states (Z_1) to the network, it regenerates the target patterns mostly deterministically for long time steps until around the $340th$ step (on overage), even though with different noise sampling at each run. The VP-RNN with W set to 0.01 is known as a random process because not only did it completely fail to regenerate target patterns, but it generated completely different sequence patterns from early steps at each test generation by using different noise sampling.

The error regression scheme was used in order to evaluate the generalization capabilities of VP-RNNs trained with different values for W. A test pattern of length 6400 steps was given to each VP-RNN and their task was to predict from 1 to 5 steps ahead. During error regression, the lower bound is maximized inside the temporal window by updating $A^{(q)}_{t-WS:t-1}$ in the inference model. The temporal window size WS was set to 50 and error regression was performed for 30 regression steps. For example, the error regression was iterated 30 times from step $t - WS$ to $t - 1$ in order to generate X_t in the case of predicting 1-step ahead. The generation can be continued to multiple look-ahead predictions steps ($X_{t+1}, X_{t+2}, ...$). The window was continuously sliding 1 step forward at a time to generate all $X_{1:6400}$. The MSE between the test pattern and the generated output for all prediction steps are given in Table 3. The VP-RNN trained

with W set to 0.25 outperforms other models in all cases, except for 1-step ahead prediction (1-step pred.) in which the VP-RNN trained with W set to 0.1 performs best in a small extent compared with networks with W set to 0.25 and 0.15. 2-D visualizations of the test, 1-step ahead predicted, and 5-steps ahead predicted patterns by VP-RNNs with W set to 1.0, 0.25, and 0.1 are displayed in Figure 7. In order to visualize clearly, the results of 200 steps are shown. As expected, predicting 5-steps ahead is challenging for any model. However, in this case, the network with W set to 0.25 performs best at preserving the structure of the target. When W was set to 0.1, it seems that the network generates a significantly noisier pattern, in the case of prediction of 5-steps. The structure looks broken in some areas, for both cases of predictions when was W set to 1.0.

For the purpose of evaluating the model's capability in look-ahead prediction of sequences in the primitive level, another experiment using the error regression scheme, but with a much longer regression window was conducted. In this experiment, 32 test sequence patterns each with a step length of 400 were used. The temporal window size WS was set to 200 and error regression was performed for 1000 regression steps. The location of the window was fixed, meaning that the error regression was done for each test pattern from time step 1 to 200 for 1000 iterations, and the generative model generated predictions of $X_{201:400}$ after the last iteration of the error regression. All 32 predicted sequences and corresponding target sequences were labeled with 3 primitive pattern types of A, B, and C for the purpose of evaluating the prediction capability of the model in the primitive sequence level. The prediction capability was evaluated for each accuracy of 1, 2, and 3 primitive prediction. This test evaluation was conducted

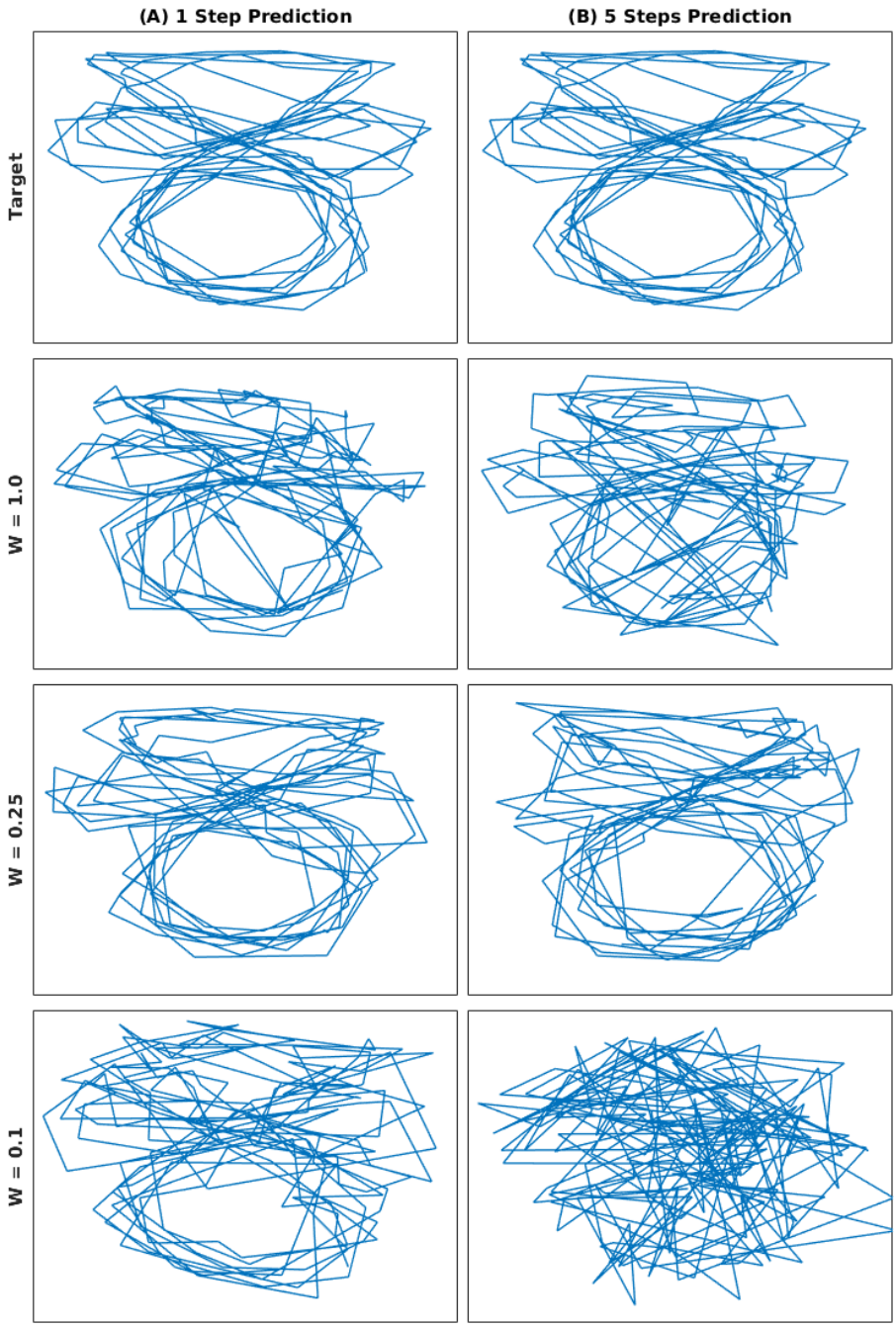

Figure 7: Target and prediction outputs of VP-RNNs with W set to 1.0, 0.25, and 0.1 during error regression when: (A) 1 step is predicted and (B) 5 steps are predicted.

for the 6 VP-RNNs trained with different W previously. Results are summarized in Table 4. It can be seen that the prediction capability for all different primitive-steps

Table 4: Percent accuracy for 1, 2, and 3 primitive prediction for models trained with different values for W.

	Meta-Prior W					
	W=1.0	W=0.5	W=0.25	W=0.15	W=0.1	W=0.01
1-prim. pred. (%)	81.25	90.625	**100**	93.75	90.625	81.25
2-prim. pred. (%)	62.5	78.125	**84.375**	68.75	59.375	37.5
3-prim. pred. (%)	40.625	53.125	**59.375**	37.5	21.875	9.375

look-ahead (from one primitive ahead to three primitives ahead) was best when W was set to 0.25. This indicates that generalization in predicting in the primitive sequence level can be achieved to the highest degree by extracting the latent probabilistic structure in primitive sequences adequately when W set between the two extremes of the most deterministic dynamic and the random process. As it has been shown that higher (middle and slow) layers learned transitions between primitive patterns while the lowest (fast) layer learned detailed information about primitive patterns in the MTRNN model (Yamashita and Tani, 2008; Ahmadi and Tani, 2017b), it can be assumed that the VP-RNN was able to predict long-term primitive sequences by using the slow timescale dynamics developed in the higher levels.

Figure 8 illustrates one instance of look-ahead prediction for a particular test sequence conducted by the model networks trained with W set to different values. In this figure, it can be seen that the network trained with $W = 0.25$ can predict most successfully the primitive sequence of (B, A, B, A, C, A, B) in the target, although there are some discrepancies in the detailed profile of each primitive pattern. By looking at the error regression window for different W values, it can be seen that the reconstruction becomes better as W diminishes. One question here is why the predictability worsened even though the reconstruction became better, by further decreasing W from 0.25 to

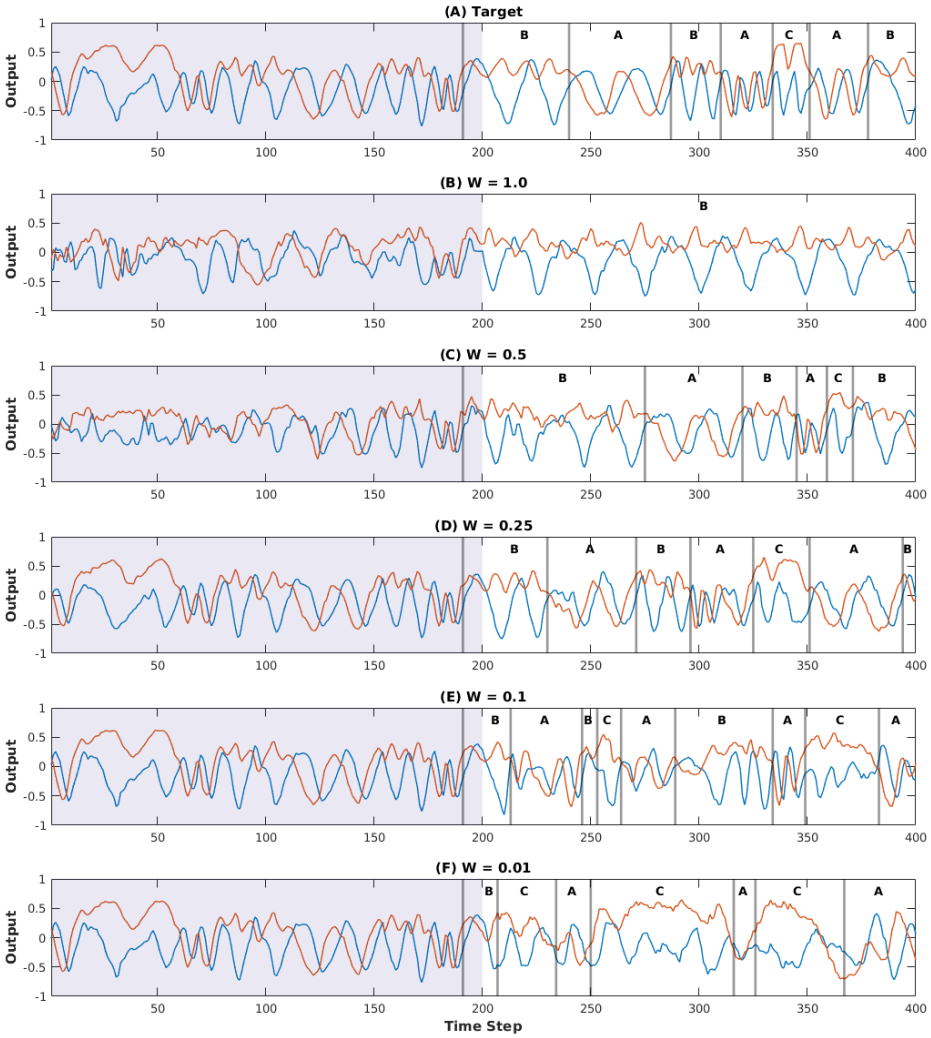

Figure 8: Targets in (A) and predicted outputs using the error regression scheme for VP-RNNs with $W = 1.0$ in (B), $W = 0.5$ in (C), $W = 0.25$ in (D), $W = 0.1$ in (E), and $W = 0.01$ in (F). Gray bars show the steps that transition between primitive patterns. Capital letters above each plot denote corresponding primitive types. In figures (B) to (F), gray areas are temporal windows in which error regression was performed.

0.01.

In considering this question, we compared the divergence between the posterior and the prior in the regression window between the cases of W set to 0.25 and W set to 0.01. Figure 9 shows the target patterns, the reconstructed outputs, the mean of posterior $\mu_t^{(q)}$, and the mean of prior $\mu_t^{(p)}$ for the same test sequence and for the networks

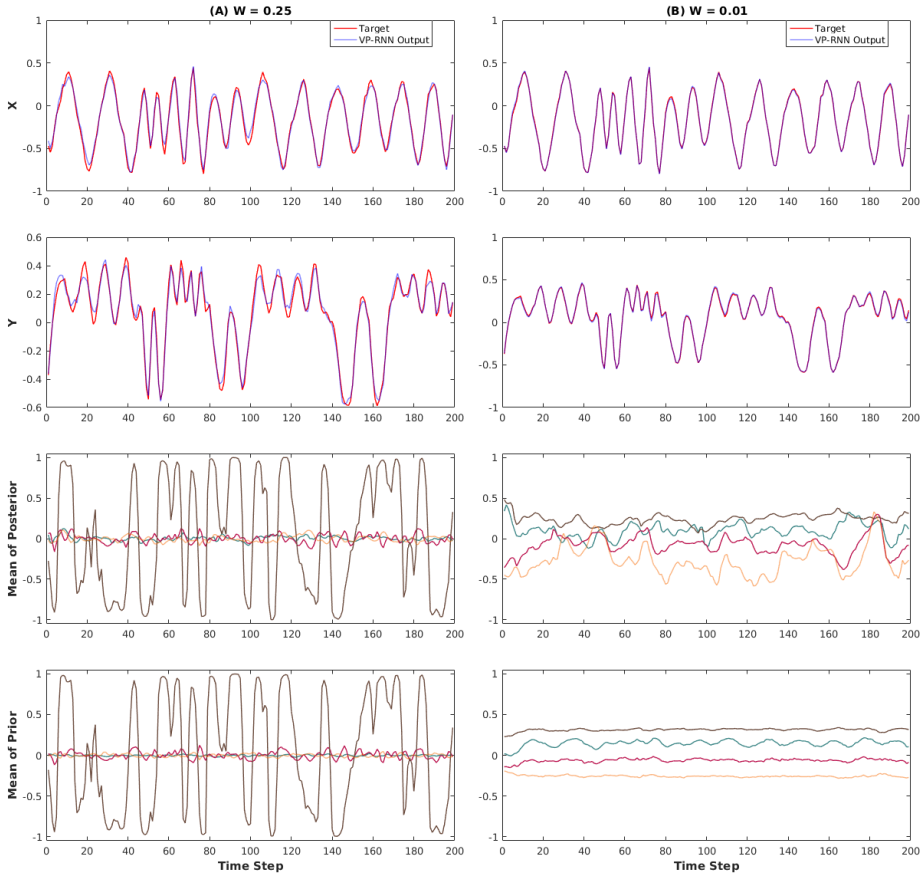

Figure 9: Comparison between VP-RNNs trained with meta-prior W set to 0.25 (A) and 0.01 (B) during the error regression. The first to fourth rows show the X dimensions of target and predicted outputs, Y dimensions of target and predicted output, the $\mu_t^{(q)}$ of the middle context layer, and the $\mu_t^{(p)}$ of the middle context layer, respectively.

trained with W set to 0.25 and 0.01. For the sake of clarity, the X and Y dimensions of $2-D$ patterns are illustrated separately, in which red and blue colors show test patterns and reconstructed patterns, respectively. It can be seen more clearly in this figure that the reconstruction is better for the network trained with W set to the smaller value (0.01). However, the difference between $\mu_t^{(q)}$ as the means in the inferred posterior and $\mu_t^{(p)}$ as the mean in the estimated prior is significantly larger for the network with $W = 0.01$ than the one with $W = 0.25$. This is due to the fact that the learning

process in the former case was imposed with less constraint in the minimization of the KL divergence between the posterior and the prior than the one in the latter case. It is speculated that the best inference of the posterior in the past error regression window can be achieved in a good balance between minimizing the reconstruction error and minimizing the KL divergence between the posterior and the prior by setting the meta-prior W in an intermediate range. This should lead to good future prediction of a long primitive sequence, as like the case of W set to 0.25 shown in Figure 8.

Discussion and Conclusion

The current paper proposed a novel stochastic RNN model of variational Bayes inference in learning, inferring, and generating temporal patterns. Learning in the model network was performed by maximizing the lower bound on the marginal likelihood of target temporal patterns. The lower bound was computed as the weighted sum between the expectation of prediction error and the KL divergence between the prior and the approximated posterior. The prior was computed for each step in terms of mean and standard deviation by means of mapping from the hidden state in the previous time step and its posterior was inferred by means of regressing the prediction error as inspired by the predictive-coding framework.

The current study attempted to examine how uncertainty or probabilistic structure hidden in observed temporal patterns can be extracted in the model through learning. For this purpose, two different classes of simulation experiments were conducted. The first experiment involved learning of discrete sequences generated by a simple proba-

bilistic finite state machine. The second experiment involved continuous temporal patterns composed of probabilistic transitions between a set of hand-generated movement primitives where the model extended with hierarchy by introducing multiple timescale properties was used. Both experiments showed qualitatively similar results where the way of representing the uncertainty or probabilistic nature in the sequences depended significantly on the weighting of the KL divergence, referred to as the meta-prior W. When W was set to a large value, deterministic chaos developed during learning tended to imitate uncertainty. On the other hand, when W set to a small value, a random process developed where uncertainty in the sequence data was imitated by sampling noise. Importantly, both experimental results indicated that the most generalization was achieved by setting W with an intermediate value between these two extremes.

Another interesting finding was the role of the inference of the posterior in the error regression window by iterative search for the purpose of predicting relatively long-term future sequences. In the second experiment, the VP-RNN extended with hierarchy was evaluated for look-ahead prediction in primitive sequences. We found that relatively long sequences of primitive transitions were successfully predicted despite some discrepancies in details in pattern profiles when W was set to an adequate intermediate value. This, however, required that a sufficient length of the past window (200 steps in the current experiment) was taken for posterior inference with a large number of iterations for error regression (1000 iterations in the current experiment). We speculate that an optimal inference for the posterior in the past window achieved through iterative computation under a good balance between minimizing reconstruction error and divergence between the posterior and the prior can result in accurate prediction of future

primitive sequences.

The iterative computation for achieving optimal inference of the posterior using the prediction error signal is considered a hallmark in the predictive-coding framework. However, most of variational Bayes RNN models do not employ such computational procedures and they infer the posterior at each time step just by means of a single recurrent mapping of the hidden state in the previous step, fed with inputs with the current time step using sequence auto-encoders (Chung et al., 2015). Both quantitative and qualitative analysis between these two approaches should be conducted in future studies.

An obvious, but crucial question, might be how to determine an optimal value for the meta-prior W rather than having it set arbitrarily by experimenters. It should be interesting to consider how W should be modulated in the course of learning, especially when the model is applied to account for developmental processes of enactive agents. Pezzulo and colleagues (Pezzulo, Rigoli, and Friston, 2015; Seth and Tsakiris, 2018) have investigated possible correspondence between active inference and homeostasis by considering that active inference by error minimization is analogous to preservation of values in generating habitual behaviors in the homeostasis framework or to the error feedback control scheme (Baltieri and Buckley, 2017). Pezzulo and colleagues proposed that the course of developmental stages along with the hierarchy could be manipulated by regulating the overall degree of precision achieved at each level. For example, habitual behaviors can be developed first by achieving good precision in the lower level in the early phase of development, whereas goal-directed control of actions can be developed afterward by doing so in the higher level in the later phase by adequately

manipulating dominant parameters such as the meta-prior in the current proposal, at each layer. Clark (Clark, 2016) suggested that such staged development, along with hierarchy could account for the so-called u-shaped development (Karmiloff-Smith, 1994) where a set of habitual behaviors developed in the earlier phase is later re-described for the purpose of their more general use, such as generating goal-directed actions by flexibly combining them in the higher level.

Furthermore, it has been speculated that regulation of the overall degree of precision or uncertainty could contribute to access the problem of balancing exploration and exploitation (Butz and Kutter, 2016; Pezzulo et al., 2015; Oudeyer and Smith, 2016). Butz and Kutter (Butz and Kutter, 2016) suggest that epistemic drives (reducing uncertainty in relevant task dimensions) might be pursued before extrinsic goals. Future studies should explore the aforementioned perspectives more precisely by examining what types of rules for modulating the essential parameters such as the meta-prior in the proposed model contribute to adequate development of adopted enactive agents.

Finally, it has been hypothesized that overestimation of precision in prediction could cause development of autism spectrum disorders (ASD) (Quattrocki and Friston, 2014). Van de Cruys et al. (Van de Cruys et al., 2014) have suggested that ASD might be caused by overly strong top-down prior potentiation to minimize prediction error (thus increasing precision), which can enhance capacities for rote learning while losing the capacity to generalize what is learned, a pathology typical of ASD. It has been presumed that in social interaction, ASD patients frequently suffer from over-amplified error in predicting behaviors of others by means of overestimated precision due to overfitting in learning. For this reason, such patients may tend to indulge in their own repetitive behaviors,

which generate only a tolerable amount of error. It is expected that future studies for simulating these pathologies using models such as the current proposal could deepen our understanding of the underlying mechanisms, not only in normal development, but also in abnormal ones, as in the case of ADS.

Appendix

A Posterior Computation

$$q_\phi(Z_t \mid \tilde{d}_{t-1}, e_{t:T}) = \mathcal{N}(Z_t; \mu_t^{(q)}, \sigma_t^{(q)}) \quad \text{where} \quad [\mu_t^{(q)}, \log \sigma_t^{(q)}] = f^{(q)}(\tilde{d}_{t-1}, A_t^{(q)}) \quad (1)$$

The equation for μ and $\log \sigma$ of posterior can be written as

$$\begin{cases} \mu_{t,i} = \tanh\left(\sum_{j=1}^{N_Z} w_{ij}^{\mu d} \tilde{d}_{t-1,j} + b_i^{\mu} + A_{t,i}^{\mu}\right) \\ \log \sigma_{t,i} = \sum_{j=1}^{N_Z} w_{ij}^{\sigma d} \tilde{d}_{t-1,j} + b_i^{\sigma} + A_{t,i}^{\sigma} \end{cases} \quad (2)$$

where $\mu_{t,i}$ is the mean state value of the i_{th} latent neural unit Z at time t, $w_{ij}^{\mu d}$ is the connectivity weight from the i_{th} mean unit to the j_{th} d unit, b_i^{μ} is the bias of the i_{th} mean unit, $\log \sigma_{t,i}$ is the log of the standard deviation state value of the i_{th} latent neural unit Z at time t, $w_{ij}^{\sigma d}$ is the connectivity weight from the i_{th} $\log \sigma$ unit to the j_{th} d unit, b_i^{σ} is the bias of the i_{th} standard deviation unit, and N_Z is the number of Z units. The notation "(q)" was omitted from the equation for the sake of simplicity. A^{μ} and A^{σ} are obtained as follows

$$\begin{cases} A_{t,i}^{\mu} = A_{t,i}^{\mu} + \alpha \frac{\partial L}{\partial A_{t,i}^{\mu}} \\ A_{t,i}^{\sigma} = A_{t,i}^{\sigma} + \alpha \frac{\partial L}{\partial A_{t,i}^{\sigma}} \end{cases} \quad (3)$$

Based on Eq. 2, we can rewrite Eq. 3 to have the derivatives with respect to mean

and standard deviation values as

$$\begin{cases} A^{\mu}_{t,i} = A^{\mu}_{t,i} + \alpha \left(1 - \tanh^2 \left((\sum_{j=1}^{N_Z} w^{\mu d}_{ij} \tilde{d}_{t-1,j} + b^{\mu}_{i} + A^{\mu}_{t,i}) \right) \right) \left(\frac{\partial L}{\partial \mu_{t,i}} \right) \\ A^{\sigma}_{t,i} = A^{\sigma}_{t,i} + \alpha \frac{\partial L}{\partial \log \sigma_{t,i}} \end{cases} \tag{4}$$

B KL Divergence

We let each posterior and prior distributions be a Gaussian with a diagonal covariance matrix, so:

$$KL[q_\phi(Z_t) \mid\mid P_{\theta_Z}(Z_t)] = E_{q_\phi}[\log q_\phi(Z_t)] - E_{q_\phi}[\log P_{\theta_Z}(Z_t)] \tag{1}$$

$$q_\phi(Z_t) = \frac{1}{\sqrt{2\pi(\sigma^q_t)^2}} e^{\frac{-(Z_t - \mu^q_t)^2}{2(\sigma^q_t)^2}} \tag{2}$$

$$P_{\theta_Z}(Z_t) = \frac{1}{\sqrt{2\pi(\sigma^p_t)^2}} e^{\frac{-(Z_t - \mu^p_t)^2}{2(\sigma^p_t)^2}} \tag{3}$$

For simplicity, we removed parenthesis from p and q. Based on Equations 1, 2, and 3:

$$E_{q_\phi}[\log q_\phi(Z_t)] = E_{q_\phi}[-\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma^q_t)^2} + \frac{-(Z_t - \mu^q_t)^2}{2(\sigma^q_t)^2}] \tag{4}$$

$$\begin{aligned} E_{q_\phi}[\log P_{\theta_Z}(Z_t)] &= E_{q_\phi}[-\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma^p_t)^2} + \frac{-(Z_t - \mu^p_t)^2}{2(\sigma^p_t)^2}] \\ &= E_{q_\phi}[-\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma^p_t)^2} + \frac{-(Z_t)^2 + 2Z_t\mu^p_t - (\mu^p_t)^2}{2(\sigma^p_t)^2}] \end{aligned} \tag{5}$$

Variance and $E[Z_t^2]$ can be written as:

$$\sigma_t^2 = E[(Z_t - \mu_t)^2], \qquad E[Z_t^2] = \mu_t^2 + \sigma_t^2 \tag{6}$$

so, Equations 4 and 5 can be rewritten as:

$$\begin{aligned} E_{q_\phi}[\log q_\phi(Z_t)] &= -\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma_t^q)^2} + \frac{-\sigma_t^2}{2(\sigma_t^q)^2} \\ &= -\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma_t^q)^2} - \frac{1}{2} \end{aligned} \tag{7}$$

$$E_{q_\phi}[\log P_{\theta_Z}(Z_t)] = -\frac{1}{2}\log 2\pi - \frac{1}{2}\log{(\sigma_t^p)^2} + \frac{-(\mu_t^q)^2 - (\sigma_t^q)^2 + 2\mu_t^q\mu_t^p - (\mu_t^p)^2}{2(\sigma_t^p)^2} \tag{8}$$

Now, Equation 1 can be rewritten as:

$$\begin{aligned} KL[q_\phi(Z_t) \,||\, P_{\theta_Z}(Z_t)] &= -\frac{1}{2}\log{(\sigma_t^q)^2} - \frac{1}{2} + \frac{1}{2}\log{(\sigma_t^p)^2} - \frac{-(\mu_t^q)^2 - (\sigma_t^q)^2 + 2\mu_t^q\mu_t^p - (\mu_t^p)^2}{2(\sigma_t^p)^2} \\ &= -\frac{1}{2}(1 + \log{(\sigma_t^q)^2} - \log{(\sigma_t^p)^2} + \frac{-(\mu_t^q)^2 - (\sigma_t^q)^2 + 2\mu_t^q\mu_t^p - (\mu_t^p)^2}{(\sigma_t^p)^2}) \end{aligned} \tag{9}$$

C Lyapunov Exponent Computation

For the sake of simplicity, let us consider a VP-RNN consisting of 2 d units and 1 Z unit. We need to first compute Jacobian matrices at each time step as

$$J_t = \begin{bmatrix} \frac{\partial Z_{t+1,1}}{\partial Z_{t,1}} & \frac{\partial Z_{t+1,1}}{\partial d_{t,1}} & \frac{\partial Z_{t+1,1}}{\partial d_{t,2}} \\ \frac{\partial d_{t+1,1}}{\partial Z_{t,1}} & \frac{\partial d_{t+1,1}}{\partial d_{t,1}} & \frac{\partial d_{t+1,1}}{\partial d_{t,2}} \\ \frac{\partial d_{t+1,2}}{\partial Z_{t,1}} & \frac{\partial d_{t+1,2}}{\partial d_{t,1}} & \frac{\partial d_{t+1,2}}{\partial d_{t,2}} \end{bmatrix}$$

It can be noted that X does not exist in the Jacobian matrices because in the generative model, $X_{1:T}$ are not given to the context layers. We can now resort the approximation of the image ellipsoid $J_T J_{T-1} ... J_1 U$ of the unit sphere by a computational algorithm. More details can be seen in (Alligood et al., 1996). Here, the computation of the first largest Lyapunov exponent is only given. Let us start with an orthonormal basis r = $[1.0\ 0.0\ 0.0]^T$, and use the Gram-Schmidt orthogonalization procedure, so we have

Algorithm 1 Lyapunov Exponent Computation

1: $LE = 0.0$
2: **for** `<t = 1 to T>` **do**
3: $y_t = J_t\, r$
4: $r = \frac{y_t}{\|\mathbf{y_t}\|}$
5: $LE \ += \ \log \|\mathbf{y_t}\|$
6: **end for**
7: $LE = \frac{LE}{T}$

where $\|.\|$ and LE denote Euclidean length, and first largest Lyapunov exponent, respectively. T was $50,000$ in our experiments.

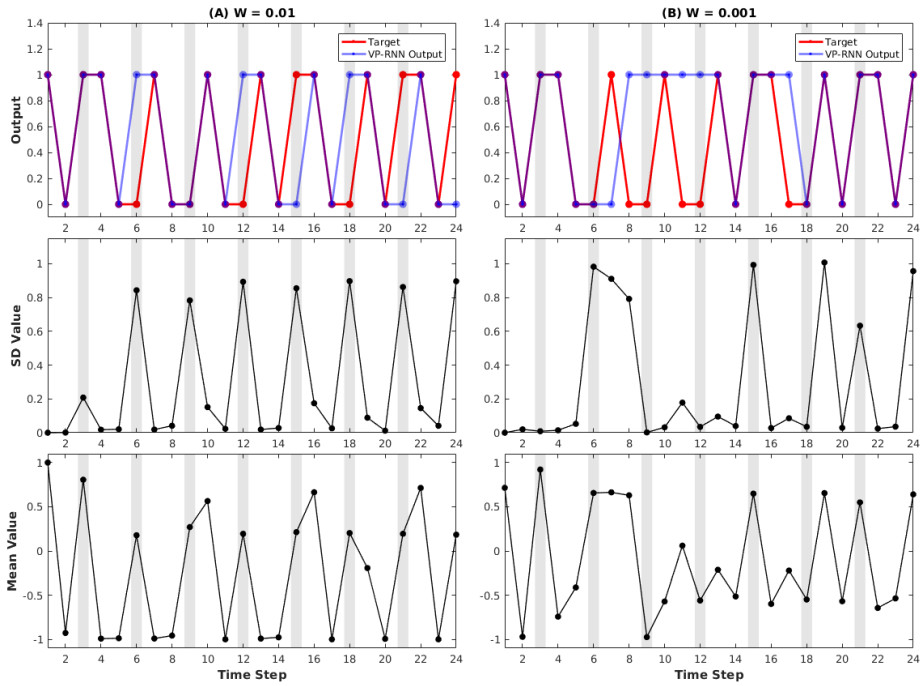

Figure 10: The target and the regenerated outputs, the mean $\mu^{(p)}$, and the standard deviation $\sigma^{(p)}$ of two VP-RNNs trained with meta-prior W set to 0.01 (A) and 0.001 (B). The gray bars show the time steps corresponding to uncertain states.

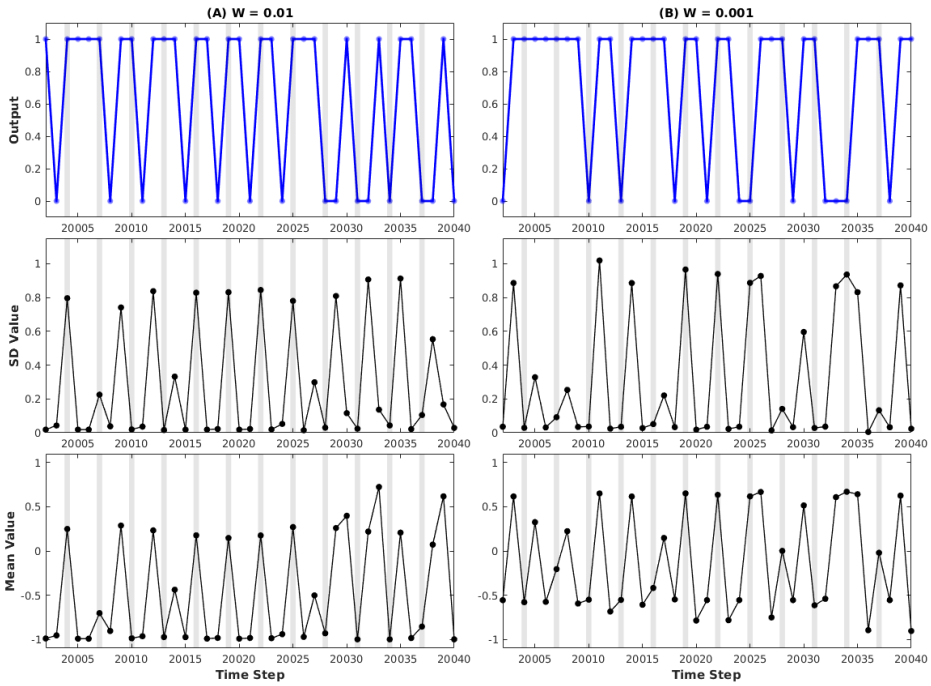

Figure 11: The generated output, the mean $\mu^{(p)}$, and the standard deviation $\sigma^{(p)}$ from time steps $20,002$ to $20,040$ of two VP-RNNs trained with the meta-prior W set to 0.01 (A) and 0.001 (B). The gray bars show the time steps corresponding to uncertain states.

References

Ahmadi, A. & Tani, J. (2017a). Bridging the gap between probabilistic and deterministic models: a simulation study on a variational bayes predictive coding recurrent neural network model. In *International conference on neural information processing* (pp. 760–769). Springer.

Ahmadi, A. & Tani, J. (2017b). How can a recurrent neurodynamic predictive coding model cope with fluctuation in temporal patterns? robotic experiments on imitative interaction. *Neural Networks*, *92*, 3–16.

Alligood, K. T., Sauer, T. D., & Yorke, J. A. (1996). *Chaos*. Springer.

Baltieri, M. & Buckley, C. L. (2017). An active inference implementation of phototaxis. In *Proceedings of the european conference on artificial life 14* (Vol. 14, pp. 36–43). MIT Press.

Bayer, J. & Osendorfer, C. (2014). Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Butz, M. V. & Kutter, E. F. (2016). *How the mind comes into being: introducing cognitive science from a functional and computational perspective*. Oxford University Press.

Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., … Abbeel, P. (2016). Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

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

Clark, A. (2015). *Surfing uncertainty: prediction, action, and the embodied mind*. Oxford University Press.

Clark, A. (2016). Precisions, slopes, and representational re-description. commentary to "Exploring Robotic Minds by Predictive Coding Principle by Jun Tani". *The Newsletter of the Technical Committee on Cognitive and Developmental Systems*, *13*(2).

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Fabius, O. & van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*.

Fraccaro, M., Sønderby, S. K., Paquet, U., & Winther, O. (2016). Sequential neural models with stochastic layers. In *Advances in neural information processing systems* (pp. 2199–2207).

Friston, K. (2005). A theory of cortical responses. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *360*(1456), 815–836.

Friston, K. (2010). The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, *11*(2), 127.

Friston, K. (2018). Does predictive coding have a future? *Nature neuroscience*, *21*(8), 1019.

Friston, K. J., Daunizeau, J., Kilner, J., & Kiebel, S. J. (2010). Action and behavior: a free-energy formulation. *Biological cybernetics*, *102*(3), 227–260.

Geiger, D., Verma, T., & Pearl, J. (1990). Identifying independence in bayesian networks. *Networks*, *20*(5), 507–534.

Goyal, A., Sordoni, A., Côté, M.-A., Ke, N., & Bengio, Y. (2017). Z-forcing: training stochastic recurrent networks. In *Advances in neural information processing systems* (pp. 6713–6723).

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Jordan, M. I. (1997). Serial order: a parallel distributed processing approach. In *Advances in psychology* (Vol. 121, pp. 471–495). Elsevier.

Karl, M., Soelch, M., Bayer, J., & van der Smagt, P. (2016). Deep variational bayes filters: unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*.

Karmiloff-Smith, B. A. (1994). Beyond modularity: a developmental perspective on cognitive science. *European journal of disorders of communication*, *29*(1), 95–105.

Kingma, D. P. & Ba, J. (2014). Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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

Kingma, D. P. & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lee, T. S. & Mumford, D. (2003). Hierarchical bayesian inference in the visual cortex. *JOSA A*, *20*(7), 1434–1448.

Murata, S., Namikawa, J., Arie, H., Sugano, S., & Tani, J. (2013). Learning to reproduce fluctuating time series by inferring their time-dependent stochastic properties: application in robot learning via tutoring. *IEEE Transactions on Autonomous Mental Development*, *5*(4), 298–310.

Murata, S., Yamashita, Y., Arie, H., Ogata, T., Sugano, S., & Tani, J. (2017). Learning to perceive the world as probabilistic or deterministic via interaction with others: a neuro-robotics experiment. *IEEE Trans. Neural Netw. Learning Syst. 28*(4), 830–848.

Oudeyer, P.-Y. & Smith, L. B. (2016). How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, *8*(2), 492–502.

Pezzulo, G., Rigoli, F., & Friston, K. (2015). Active inference, homeostatic regulation and adaptive behavioural control. *Progress in neurobiology*, *134*, 17–35.

Quattrocki, E. & Friston, K. (2014). Autism, oxytocin and interoception. *Neuroscience & Biobehavioral Reviews*, *47*, 410–430.

Rao, R. P. & Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, *2*(1), 79.

Rao, R. P. & Sejnowski, T. J. (2000). Predictive sequence learning in recurrent neocortical circuits. In *Advances in neural information processing systems* (pp. 164–170).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1985). *Learning internal representations by error propagation*. California Univ San Diego La Jolla Inst for Cognitive Science.

Seth, A. K. & Tsakiris, M. (2018). Being a beast machine: the somatic basis of selfhood. *Trends in cognitive sciences*.

Shabanian, S., Arpit, D., Trischler, A., & Bengio, Y. (2017). Variational bi-lstms. *arXiv preprint arXiv:1711.05717*.

Tani, J. & Fukumura, N. (1995). Embedding a grammatical description in deterministic chaos: an experiment in recurrent neural learning. *Biological Cybernetics*, *72*(4), 365–370.

Tani, J. & Ito, M. (2003). Self-organization of behavioral primitives as multiple attractor dynamics: a robot experiment. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *33*(4), 481–488.

Tani, J., Ito, M., & Sugita, Y. (2004). Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using rnnpb. *Neural Networks*, *17*(8-9), 1273–1289.

Tani, J. & Nolfi, S. (1999). Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, *12*(7-8), 1131–1141.

Van de Cruys, S., Evers, K., Van der Hallen, R., Van Eylen, L., Boets, B., de-Wit, L., & Wagemans, J. (2014). Precise minds in uncertain worlds: predictive coding in autism. *Psychological review*, *121*(4), 649.

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Yamashita, Y. & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS computational biology*, *4*(11), e1000220.

Zhao, T., Zhao, R., & Eskenazi, M. (2017). Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.