Learning to Reproduce Stochastic Time Series Using Stochastic LSTM

Sadaf Gulshad, Dick Sigmund, and Jong-Hwan Kim
School of Electrical Engineering
KAIST
Daejeon 34141, Republic of Korea
Email: {sadaf, dick, johkim}@rit.kaist.ac.kr

***Abstract*—Recurrent neural networks (RNNs) have been widely used for complex data modeling. However, when it comes to long-term time-dependent complex sequential data modeling with stochasticities, RNNs seem to fail because of vanishing gradients problem. Hence, in this paper, we propose a new architecture, stochastic long short term memory (S-LSTM), along with its forward and backward dynamics equations. S-LSTM models stochasticities using Bayesian brain hypothesis, which is a probabilistic model that makes predictions against which samples are tested to update the conclusions about their causes. This is the same as minimizing the difference between inference and posterior densities for suppressing the free energy. During training of S-LSTM, it predicts the mean as well as variance at each time step. The prediction error is minimized by the predicted variance which acts as an inverse weighting factor for prediction error and tries to optimize the maximum likelihood. Our proposed model is evaluated through numerical experiments on noisy Lissajous curves. In the experiments, S-LSTM is found to predict and preserve more stochasticities in the noisy Lissajous curves as compared to LSTM.**

I. Introduction

Artificial neural networks are inspired by the vast and complicated neural system of the human brain. In order to understand artificial neural networks and to improve their efficiency in uncertain environments, the best way is to understand how human brain works in such situations. Understanding the working of the human brain in uncertain situations can give us the clue for the working of the human brain in normal situations [1]. As the real world is nondeterministic, human brain confronts countless uncertainties every day and finds the way to deal with it. In the field of cognitive robotics, intelligent architectures inspired by human brain are used to make robots learn and reason in the real world environment for performing the complex tasks [2]. Neural networks are one among those intelligent architectures and they have found numerous applications in the field of robotics [3]–[6]. Robots also face uncertainties in the real environment. For example, in order to perform the task of moving the arm of a robot from one position to the other, the robot faces uncertainties about the position of an arm at each point. The robot uses either its proprioception or exteroception for locating its hand position but both of these robot's perception methods give noisy outputs no matter how good the sensors are. Another example could be a robot trained to walk inside the laboratory on the plain ground but when the same robot walks outside the laboratory the ground is not the same so it faces uncertainties. These uncertainties create the problem of estimating the state of the environment and the control of robot's motors within a statistical framework [1].

The Bayesian probability theory has been widely used for solving the problems of uncertainties in the environment. This theory gives the probabilistic model that generates the predictions against which sensory data is tested to update the conclusions about their causes. The Bayesian model can be divided into likelihood $p(x|z)$ (the probability of sensory data, given their causes) and a prior $p(z)$ (the prior probability of those causes). Then, the recognition becomes the process of inverting the likelihood $p(z|x)$ (mapping from causes to perception) to access the posterior probability of the causes, given sensory data (mapping from sensations to causes). This inversion is the same as minimizing the difference between the recognition and posterior densities to suppress free energy [7], as shown in Fig. 1.

Although probabilistic models can be used for making predictions and updating the beliefs about their reasons, there are a few factors which make the use of probabilistic models difficult. Firstly, in real world problems as the number of variables become innumerable, we cannot write the distributions over a large number of variables. Secondly,

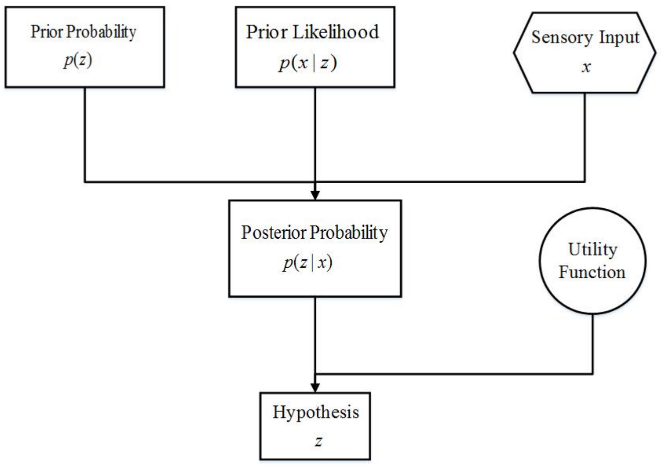


Fig. 1: Bayesian brain model.

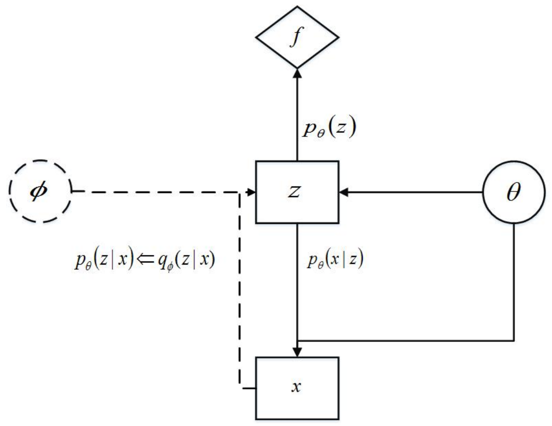


Fig. 2: Directed graphical model of variational Bayes.

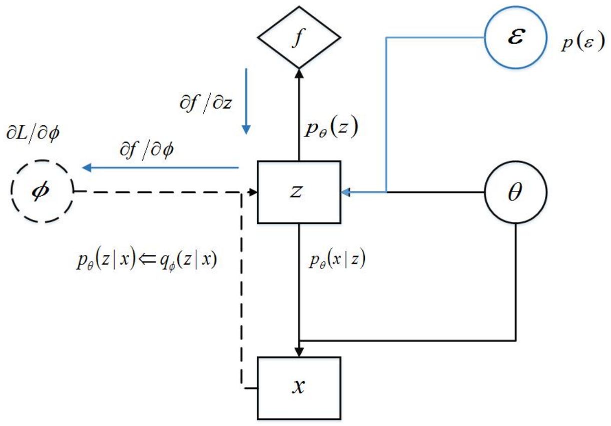


Fig. 3: Backward pass of reparameterization trick.

with the increase in the number of variables, the resources required to represent the probability distributions also increase exponentially [7]. Therefore, in order to solve this problem of representation and reasoning in the complex domain using random variables, graphical models have been introduced.

The directed graphical models used for the representation of Bayesian theory consist of nodes and edges. Each node represents a random variable and the edges represent the dependencies between those random variables. The dependencies represented by the edges are in consistence with the Markov condition i.e. the child is conditioned on parents [8], [9]. Bayesian directed graphical models can be identified as generative models as each variable having no parents is sampled, then its successor is sampled conditioned on the values of its antecedent. Using those samples from the distribution the observable data is generated. Furthermore, by observing the structure of the generated data, it could be suggested that the data is reproduced as a result of some hidden or latent variables [7].

The learning in Bayesian models can be performed through several ways, e.g. gradient descent. However, as the directed graphical models contain latent variables, it becomes difficult to make inferences and do learning in directed probablistic models. Thus, the likelihood of observable variables is tried to be maximized. Nonetheless, this maximization also becomes intractable in case of vast number of hidden variables. In order to solve this problem of intractable posterior due to hidden variables, reparametrization trick solution was proposed in which instead of maximizing the likelihood, the lower bound of the likelihood is maximized [10]. To understand this problem, let us consider the directed graphical model shown in Fig. 2. In the figure, x is the observed variable, z is the hidden variable, and θ is the model parameter. $p_\theta(z)$ is the prior of latent variable and is parameterized by θ. $p_\theta(x|z)$ is the likelihood parameterized by θ. $p_\theta(x|z)p_\theta(z) = p_\theta(x,z)$ is the generative model parameterized by θ. $p_\theta(z|x) = \frac{p_\theta(x,z)}{p_\theta(x)}$ is the intractable true posterior for z given x. $q_\phi(z|x)$ is the recognition model to estimate posterior for z given x with parameter ϕ. Thus, we want to infer a posterior distribution $p_\theta(z|x)$ as

$$p_\theta(z|x) = \frac{p_\theta(x,z)}{p_\theta(x)} = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}. \tag{1}$$

However, $p_\theta(z|x)$ in (1) is intractable so we need good approximations for it. In order to solve the Bayesian inference problem, a parametric model $q_\phi(z|x)$ is introduced [10], where $q_\phi(z|x)$ is the approximate of true distribution $p_\theta(z|x)$. Now we need to minimize the KL divergence between true distribution and approximate distribution in which KL divergence is defined as

$$\begin{aligned}\log p(x) &= D_{KL}(q||p) - \int q_\phi(z|x) \log\Big(\frac{p_\theta(x,z)}{q_\phi(z|x)}\Big)dz \\ &= D_{KL}(q||p) + L(q).\end{aligned} \tag{2}$$

As $\log p(x)$ is fixed in (2), maximizing the variational lower bound $L(q)$ will minimize the KL divergence between $q_\phi(z|x)$ and $p_\theta(z|x)$. This is equivalent to maximizing the free energy by optimizing the parameters [11]. In order to optimize $L(q)$, we need to differentiate it with respect to both generative parameters θ and variational parameters ϕ, but the approximate distribution $q_\phi(z|x)$ is not differentiable as it is estimated by sampling. Therefore, we cannot backpropagate error through it. Thus, in order to solve this problem, sampler parametrization trick was proposed in Kingma's Variational Auto-encoders [12], shown in Fig. 3, as follows:

$$z = f(\phi, \epsilon) \tag{3}$$

where z is converted to a deterministic function of ϕ and noise ϵ. In the case of sampling from Gaussian distribution z will become:

$$z = \mu + \sigma \odot \epsilon, \epsilon \sim N(0, I) \tag{4}$$

where mean μ and variance σ are predicted by the network which is analogous with implementing predictive coding [11]. By using (4), now we can maximize the lower bound in (2).

This paper proposes stochastic long short term memory (S-LSTM). S-LSTM can learn to predict the mean as well as the variance at each time step. The predicted variance acts as an inverse weighting factor for the prediction error which is backpropagated in the process of learning [13], [14]. The formulation of the model is similar to the free energy minimization principle [11]. We also present the complete architecture of S-LSTM along with the equations for its forward and backward dynamics.

This paper is organized as follows. Section II presents the overview of Vanilla LSTM and the procedure of proposed S-LSTM. Section III shows the experiment setup and the results. Finally, concluding remarks follow in Section IV.

II. Model

Firstly, we will describe the Vanilla LSTM introduced in [15]. Then, we will describe the changes in our proposed model: Stochastic LSTM (S-LSTM).

A. Vanilla LSTM

Vanilla LSTM consists of input block x_t; three gates which are input gate i_t, forget gate f_t, and output gate o_t; cell unit c_t; and output block h_t. The output block is recurrently connected to all gates and a new memory cell at the next time step. Fig. 4 shows the Vanilla LSTM network model in which mean squared error of output and training data is backpropagated through the network. The mean-square error function is given as follows:

$$MSE = (\hat{y_t} - y_t)^2. \quad (5)$$

The forward dynamics of Vanilla LSTM are shown as follows:

$$i_t = \sigma(W_{xi}{\cdot}x_t + W_{hi}{\cdot}h_{t-1} + W_{ci}{\cdot}c_{t-1} + b_i) = \sigma(a_{i_t}) \quad (6)$$

$$f_t = \sigma(W_{xf}{\cdot}x_t + W_{hf}{\cdot}h_{t-1} + W_{cf}{\cdot}c_{t-1} + b_f) = \sigma(a_{f_t}) \quad (7)$$

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot tanh(W_{xc}{\cdot}x_t + W_{hc}{\cdot}h_{t-1} + b_c) \\ &= f_t \odot c_{t-1} + i_t \odot tanh(a_{c_t}) \end{aligned} \quad (8)$$

$$o_t = \sigma(W_{xo}{\cdot}x_t + W_{ho}{\cdot}h_{t-1} + W_{co}{\cdot}c_{t-1} + b_o) = \sigma(a_{o_t}) \quad (9)$$

$$h_t = o_t{\odot}tanh(c_t) \quad (10)$$

where W is the weight, σ is sigmoid activation function, $\cdot$ is matrix multiplication and $\odot$ is element-wise multiplication. Finally, the output (mean) of the network is predicted as follows:

$$y_t = W_{hy}{\cdot}h_t + b_y. \quad (11)$$

B. S-LSTM

S-LSTM works under S-CTRNN [13] framework in which training minimizes the free energy by maximizing likelihood function using gradient ascent method. Different with Vanilla LSTM which only predicts the mean h_t, S-LSTM predicts both mean h_t and variance v_t. Fig. 5 shows S-LSTM network model. Instead of backpropagating the mean squared error of output and training data through the network, S-LSTM backpropagates the negative logarithm of likelihood function from mean, variance, and training data through the network. The negative logarithm of the likelihood function is given as follows:

$$\ln L_{out} = \sum_{t=1}^{T} \left(-\frac{\ln(2\pi v_t)}{2} - \frac{(y_t - \hat{y_t})^2}{2v_t} \right). \quad (12)$$

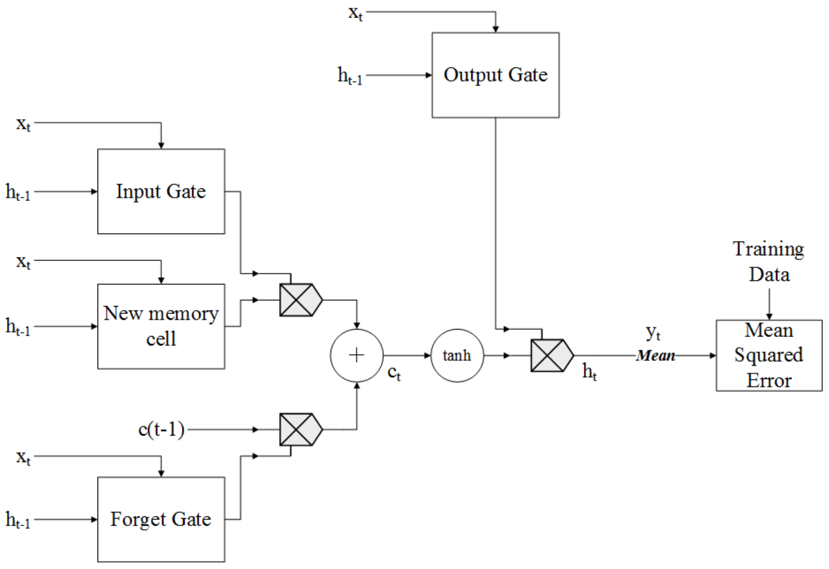


Fig. 4: Original model: Vanilla LSTM.

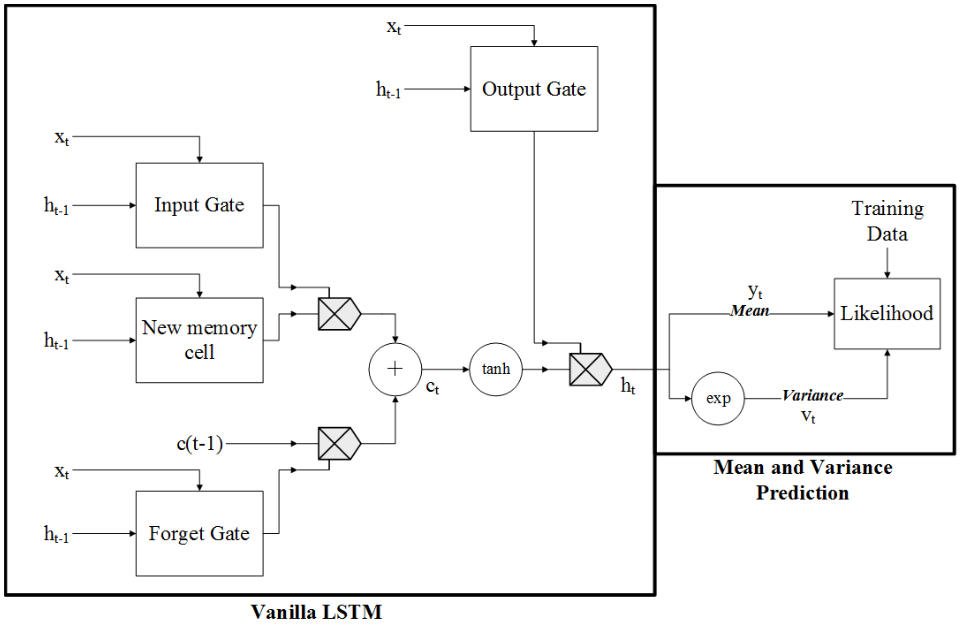


Fig. 5: Proposed model: S-LSTM. This figure shows that S-LSTM does not predict only mean, but also the variance of the data. The backpropagation of the proposed architecture is performed through maximizing the likelihood function of the mean and variance.

Later, the model parameters (weights) are updated in accordance with:

$$W_t = W_{t-1} + \Delta W_t \tag{13}$$

where

$$\Delta W_t = \alpha\left(\frac{\partial \ln L_{out}}{\partial W} + \eta \Delta W_{t-1}\right). \tag{14}$$

1) S-LSTM Forward Dynamics: The forward dynamics of S-LSTM are similar to the one in Vanilla LSTM as shown in (6) - (10). However, besides predicting the mean, S-LSTM also adds variance prediction by exponential activation function to the output block h_t. The mean and variance prediction equations are as follows:

$$y_t = W_{hy} \cdot h_t + b_y \tag{15}$$

$$v_t = \exp(W_{hv} \cdot h_t + b_v). \tag{16}$$

2) S-LSTM Backward Dynamics: Error backpropagation in S-LSTM utilizes negative logarithm of likelihood function in (12). Therefore, at first we need to find the partial differential equations for each learnable parameter of mean, $\partial \ln L_{out}/\partial y_t$, and variance, $\partial \ln L_{out}/\partial v_t$, by differentiating (12) respect to mean y_t, and variance v_t as follows:

$$\frac{\partial \ln L_{out}}{\partial y_t} = \frac{(y_t - \hat{y}_t)}{v_t} \tag{17}$$

$$\frac{\partial \ln L_{out}}{\partial v_t} = -\frac{1}{2} + \frac{(y_t - \hat{y}_t)^2}{2v_t}. \tag{18}$$

The negative logarithm of the likelihood function is also backpropagated to the output block h_t as follows:

$$\begin{aligned}\frac{\partial \ln L_{out}^{y_t}}{\partial h_t} = \sum_k W_{hy,k} \cdot \frac{\partial \ln L_{out}}{\partial y_{t,k}} + \sum_d W_{ho,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{o_{t+1},d}} \\ + \sum_d W_{hc,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{c_{t+1},d}} + \sum_d W_{hf,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{f_{t+1},d}} \\ + \sum_d W_{hi,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{i_{t+1},d}}\end{aligned} \tag{19}$$

$$\begin{aligned}\frac{\partial \ln L_{out}^{v_t}}{\partial h_t} = \sum_k W_{hv,k} \cdot \frac{\partial \ln L_{out}}{\partial v_{t,k}} + \sum_d W_{ho,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{o_{t+1},d}} \\ + \sum_d W_{hc,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{c_{t+1},d}} + \sum_d W_{hf,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{f_{t+1},d}} \\ + \sum_d W_{hi,d} \cdot \frac{\partial \ln L_{out}}{\partial a_{i_{t+1},d}}.\end{aligned} \tag{20}$$

After computing the changes in output block h_t, backpropagation proceeds to the output gate to update the corresponding weights using the following equations:

$$\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial o_t} = \frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial h_t} \frac{h_t}{o_t} \tag{21}$$

where

$$\frac{\partial h_t}{\partial o_t} = \sigma'(o_t) \odot tanh(c_t). \tag{22}$$

For weights to the memory cell, forget gate, and input gate, we need to calculate the influence of cell unit c_t, changes in advance as follows:

$$\begin{aligned}\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial c_t} &= W_{co}\cdot\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{o_t}} + o_t \odot tanh'(c_t)\\ &\odot\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial h_t} + f_{t+1}\odot\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial c_{t+1}}\\ &+ W_{cf}\cdot\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{f_{t+1}}} + W_{ci}\cdot\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{i_{t+1}}}.\end{aligned} \tag{23}$$

Then, weights update for memory cell, input gate and forget gate can be done respectively by utilizing (15) as shown below.

- Memory Cell

$$\begin{aligned}\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{c_t}} &= \frac{\partial c_t}{\partial a_{c_t}}\frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}\\ &= i_t \odot tanh'(a_{c_t}) \odot \frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}\end{aligned} \tag{24}$$

- Forget Gate

$$\begin{aligned}\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{f_t}} &= \frac{\partial c_t}{\partial a_{f_t}}\frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}\\ &= \sigma'(a_{f_t}) \odot c_{t-1} \odot \frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}\end{aligned} \tag{25}$$

- Input Gate

$$\begin{aligned}\frac{\partial \ln L_{out}^{(y_t,v_t)}}{\partial a_{i_t}} &= \frac{\partial c_t}{\partial a_{i_t}}\frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}\\ &= \sigma'(a_{i_t}) \odot tanh(c_t) \odot \frac{\partial \ln L_{out}^{y_t,v_t}}{\partial c_t}.\end{aligned} \tag{26}$$

Finally, using (17) - (26) all weights can be updated as follows:

$$W_{hy} = W_{hy} + \eta\frac{\partial \ln L_{out}}{\partial y_t} \tag{27}$$

$$W_{hv} = W_{hv} + \eta\frac{\partial \ln L_{out}}{\partial v_t} \tag{28}$$

$$W_{xo} = W_{xo} + \eta\frac{\partial \ln L_{out}}{\partial a_{o_t}} \tag{29}$$

$$W_{xc} = W_{xc} + \eta\frac{\partial \ln L_{out}}{\partial a_{c_{t+1}}} \tag{30}$$

$$W_{xf} = W_{xf} + \eta\frac{\partial \ln L_{out}}{\partial a_{f_{t+1}}} \tag{31}$$

$$W_{xi} = W_{xi} + \eta\frac{\partial \ln L_{out}}{\partial a_{i_{t+1}}} \tag{32}$$

$$W_{ho} = W_{ho} + \eta\frac{\partial \ln L_{out}}{\partial a_{o_{t+1}}} \tag{33}$$

$$W_{hc} = W_{hc} + \eta\frac{\partial \ln L_{out}}{\partial a_{c_{t+1}}} \tag{34}$$

$$W_{hf} = W_{hf} + \eta\frac{\partial \ln L_{out}}{\partial a_{f_{t+1}}} \tag{35}$$

$$W_{hi} = W_{hi} + \eta\frac{\partial \ln L_{out}}{\partial a_{i_{t+1}}} \tag{36}$$

$$W_{co} = W_{ho} + \eta\frac{\partial \ln L_{out}}{\partial a_{o_t}} \tag{37}$$

$$W_{cf} = W_{cf} + \eta\frac{\partial \ln L_{out}}{\partial a_{f_{t+1}}} \tag{38}$$

$$W_{ci} = W_{ci} + \eta\frac{\partial \ln L_{out}}{\partial a_{i_{t+1}}}. \tag{39}$$

III. Experiment

Details about dataset, environment, and results of the experiment are presented in this section.

A. Experiment Dataset

To verify our proposed model, we conducted experiments, in which S-LSTM is trained to generate stochastic time series Lissajous curves. These sinusoidal curves have their applications in robotics, for example for determining the angle with which we should move the arm of the robot to reach the goal. In this experiment, Lissajous curves are described by the following parametric equations:

$$Y_{t,1} = -0.4\cos\left(\frac{2\pi t}{P} - 1\right) \tag{40}$$

$$Y_{t,2} = 0.8\sin\left(\frac{2\pi t}{P}\right) \tag{41}$$

$$Y_{t,3} = -0.4\cos\left(\frac{4\pi t}{P} - 1\right) \tag{42}$$

$$Y_{t,4} = 0.8\sin\left(\frac{4\pi t}{P}\right) \tag{43}$$

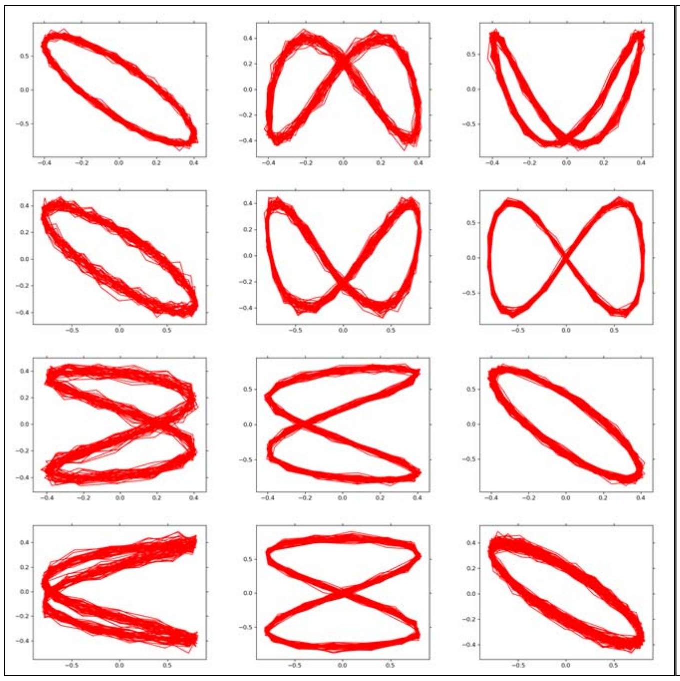
where period P used is 25 and time step t is from 0 to 999.

In order to provide stochasticities in the Lissajous curves, we added Gaussian noises ϵ_t to the parametric equations defined in (40) - (43). The standard deviations of the Gaussian noises σ_t for training and testing are given in Table I. Combining the parametric equations and Gaussian noise with standard deviation provided, twelve noisy Lissajous curves for each training and testing, shown in Fig. 6, are generated as follows:

$$\begin{aligned}(x_{t,1}^{1}, y_{t,1}^{1}) &= (Y_{t,1} + \epsilon_{t,1}, Y_{t,2} + \epsilon_{t,2}),\\ (x_{t,1}^{2}, y_{t,1}^{2}) &= (Y_{t,1} + \epsilon_{t,1}, Y_{t,3} + \epsilon_{t,2}),\\ &\ldots\ldots\ldots\ldots\ldots\ldots\\ (x_{t,1}^{12}, y_{t,1}^{12}) &= (Y_{t,4} + \epsilon_{t,1}, Y_{t,3} + \epsilon_{t,2}).\end{aligned} \tag{44}$$

TABLE I: Standard Deviation of Gaussian Noise Used in the Experiment Dataset

Standard deviation	For training	For testing
$\sigma_{t,1}$	0.01	0.03
$\sigma_{t,2}$	0.03	0.01


(a)

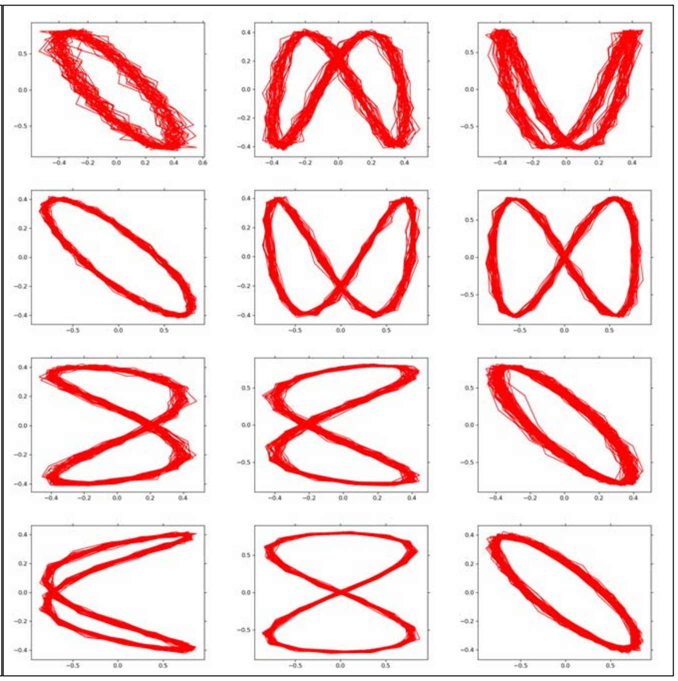

(b)

Fig. 6: Twelve noisy Lissajous curves generated for each (a) training and (b) testing dataset.

B. Experiment Environment

In the experiment, S-LSTM was compared with Vanilla LSTM to learn and generate the noisy Lissajous curves described in the previous subsection. The implementation library used for simulation is Google Brain's TensorFlow library.

The network size and learning rate for both LSTM and S-LSTM used in the experiment were 3×2 and 0.0006 respectively with 100,000 training epochs and 10,000 testing epochs. The optimization method used for gradient ascent was AdaGrad [16]. The learning curve for error of both networks was evaluated by the following mean squared error:

$$MSE = \frac{1}{NM} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (X(i,j) - Y(i,j))^2 \qquad (45)$$

where N and M are the size of data, X is the training data and Y is the output of the network.

C. Experiment Results

Fig. 7 shows the comparison between the output of LSTM and S-LSTM with the same training data set in Fig. 6a. We can see that both LSTM and S-LSTM provide good generations of the twelve noisy Lissajous curves. On the other hand, we can also observe that all of the noisy Lissajous curves generated by S-LSTM in Fig. 7b have more randomness than the curves generated by LSTM in Fig. 7a. This indicates that S-LSTM outperformed LSTM in storing more stochasticities of the training dataset. Also by comparing the learning curves of both LSTM and S-LSTM in Fig. 8, we can observe that S-LSTM converged faster than LSTM in most cases.

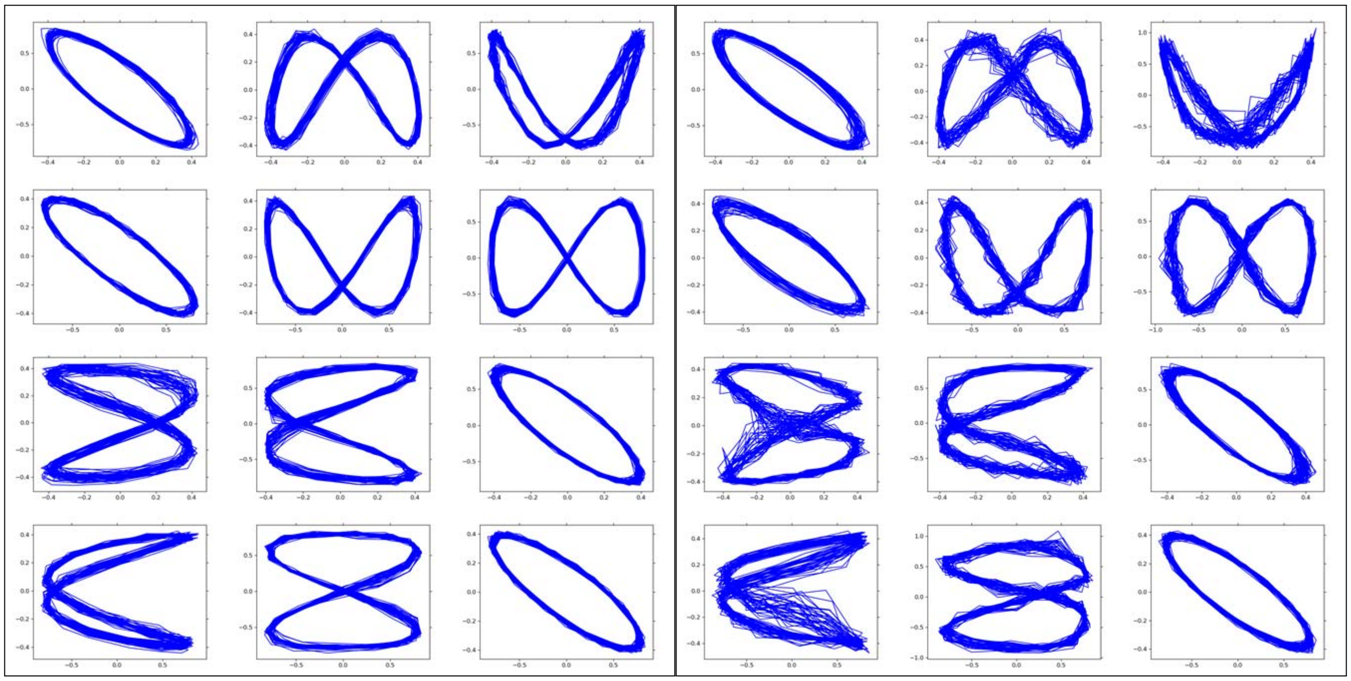
Whereas for testing as shown in Fig. 9, both LSTM and S-LSTM seems to provide outputs which resemble the testing data set in Fig. 6b. However, there are slightly different types of stochasticites in the generated curves due to different Gaussian noises added for training and testing data sets.

IV. Conclusion

In this paper, we proposed a novel variant of LSTM, called S-LSTM, along with its equations for forward and backward dynamics. S-LSTM produces stochastic time series by predicting mean and variance which in turn performs free energy minimization. In the experiment, LSTM and S-LSTM were compared to learn and generate stochastic time series. We observed that S-LSTM was proven to be able to preserve and generate more stochasticities than LSTM. For further developments, our proposed architecture could be modified for designing variational auto-encoder and could be used for solving Bayesian inference problems.

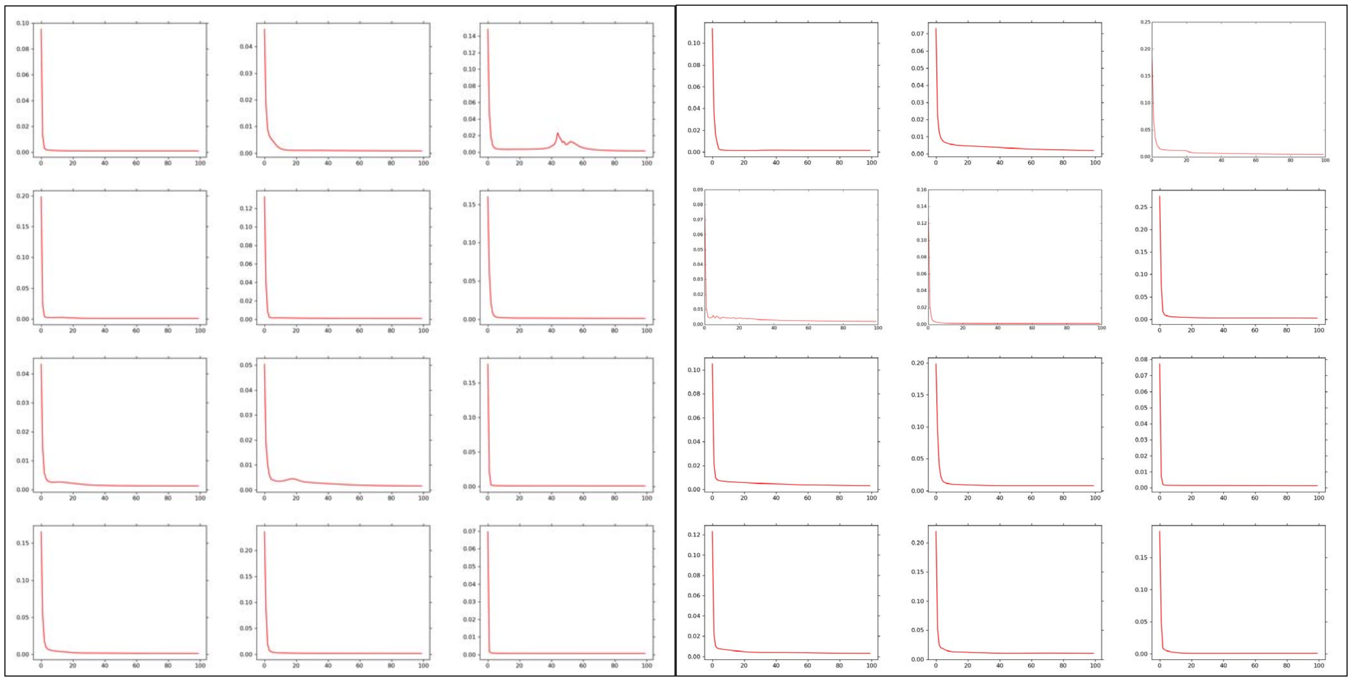
V. Acknowledgment

This work was supported by the ICT R&D program of MSIP/IITP [2016-0-00563, Research on adaptive machine learning technology development for intelligent autonomous digital companion].



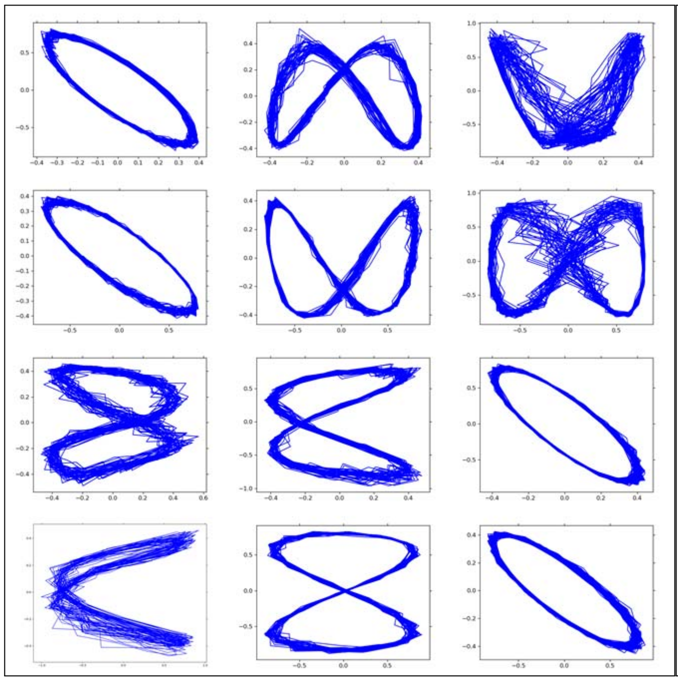
(a) (b)

Fig. 7: Training output of (a) LSTM and (b) S-LSTM. We can observe that LSTM outputs are more deterministic while the outputs of S-LSTM are more stochastic as S-LSTM is preserving the variance along with the mean of the training data.



(a) (b)

Fig. 8: Learning curves of (a) LSTM and (b) S-LSTM training.


(a)

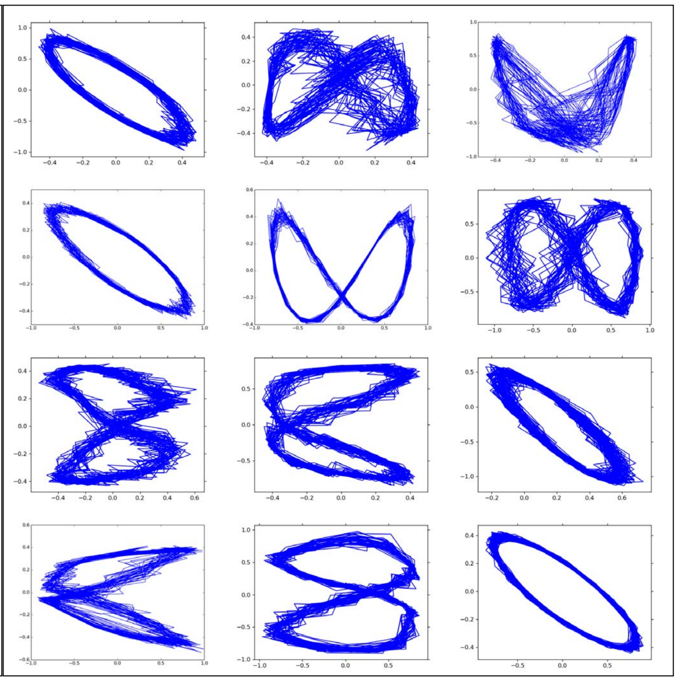

(b)

Fig. 9: Testing output of (a) LSTM and (b) S-LSTM. Same as the training output, S-LSTM predictions store more uncertainties compared to those of LSTM though in a few experiments S-LSTM seem to be unable to preserve their original shapes accurately.

REFERENCES

[1] K. P. Körding and D. Wolpert, "Bayesian decision theory in sensorimotor control," *Trends in Cognitive Sciences*, vol. 10, no. 7, pp. 319–326, 2006.

[2] J.-H. Kim, S.-H. Choi, I.-W. Park, and S. A. Zaheer, "Intelligence technology for robots that think," *IEEE Computational Intelligence Magazine*, vol. 8, no. 3, pp. 70–84, 2013.

[3] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.

[4] G.-M. Park and J.-H. Kim, "Deep adaptive resonance theory for learning biologically inspired episodic memory," in *Proceedings IEEE International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, Canada, Jul. 2016.

[5] J.-Y. Park, Y.-H. Yoo, D.-H. Kim, and J.-H. Kim, "Integrated adaptive resonance theory neural model for episodic memory with task memory for task performance of robots," in *Proceedings IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, BC, Canada, Jul. 2016.

[6] M.-J. Kim, S.-H. Baek, S.-H. Cho, and J.-H. Kim, "Approach to integrate episodic memory into cogency-based behavior planner for robots," in *Proceedings IEEE International Conference on Systems, Man, and Cybernetics*, Budapest, Hungary, Oct. 2016.

[7] T. Griffiths and A. Yuille, "A primer on probabilistic inference," *The Probabilistic Mind: Prospects for Bayesian Cognitive Science*, pp. 33–57, 2008.

[8] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[9] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*. MIT press, 2000.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[11] K. Friston, "The free-energy principle: a unified brain theory?" *Nature Reviews Neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[12] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[13] S. Murata, J. Namikawa, H. Arie, S. Sugano, and J. Tani, "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*, vol. 5, no. 4, pp. 298–310, 2013.

[14] J. Bayer and C. Osendorfer, "Learning stochastic recurrent networks," *arXiv preprint arXiv:1411.7610*, 2014.

[15] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, Jul. 2011.