LEARNING TEMPORAL EVOLUTION OF PROBABILITY DISTRIBUTION WITH RECURRENT NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose to tackle a time series regression problem by computing temporal evolution of a probability density function to provide a probabilisitic forecast. A Recurrent Neural Network (RNN) based model is employed to learn a nonlinear operator for temporal evolution of a probability density function. We use a softmax layer for a numerical discretization of a smooth probability density functions, which transforms a function approximation problem to a classification task. We propose explicit and implicit regularization strategies to impose a smoothness condition on the estimated probability distribution. A Monte Carlo procedure to compute the temporal evolution of the distribution for a multiple-step forecast is presented. The evaluation of the proposed algorithm on two synthetic and two real data sets shows advantage over the compared baselines.

1 INTRODUCTION

Application of the deep learning in manufacturing processes has attracted a great attention as one of the core technologies in Industry 4.0 (Lasi et al., 2014). In many manufacturing processes, e.g. blast furnace, smelter, and milling, the complexity of the overall system makes it almost impossible or impractical to develop a simulation model from the first principles. Hence, system identification from sensor observations has been a long-standing research topic (Wang et al., 2016). Still, when the observation is noisy and there is no prior knowledge on the underlying dynamics, there is only a very limited number of methods for the reconstruction of nonlinear dynamics.

In this work, we consider the following class of problems, where the system is driven by a complex underlying dynamical system, e.g.,

$$\frac{\partial y}{\partial t} = \mathcal{F}(y(t), y(t-\tau), \boldsymbol{u}(t)). \tag{1}$$

Here, $y(t)$ is a continuous process, $\mathcal{F}$ is a nonlinear operator, τ is a delay-time parameter, and $\boldsymbol{u}(t)$ is an exogenous forcing, such as control parameters. At time step t, we then observe a noisy measurement of $y(t)$ which can be defined by a following noise model

$$\hat{y}_t = y(t)\nu_t + \epsilon_t, \tag{2}$$

where ν_t is a multiplicative and ϵ_t is an additive noise processes. In (1) and (2), we place no assumption on function $\mathcal{F}$, do not assume any distributional properties of noises ν_t and ϵ_t, but assume the knowledge of the control parameters $\boldsymbol{u}(t)$.

Since the noise components, ν_t and ϵ_t, are stochastic processes, the observation $\hat{y}_t$ is a random variable. In this work, we are interested in computing temporal evolution of the probability density function (PDF) of $\hat{y}$, given the observations up to time step t, i.e., $p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$ for $n \geq 1$, where $\widehat{\boldsymbol{Y}}_{0:t} = (\hat{y}_0, \cdots, \hat{y}_t)$ is a trajectory of the past observations of $\hat{y}$ and $\boldsymbol{U}_{0:t+n-1} = (\boldsymbol{u}_0, \cdots, \boldsymbol{u}_{t+n-1})$ consists of the history of the known control actions, $\boldsymbol{U}_{0:t-1}$, and a future control scenario, $\boldsymbol{U}_{t:t+n-1}$. We show, in Section 3, a class of problems, where simple regression problem of forecasting the value of $\hat{y}_{t+n}$ is not sufficient or not possible, e.g. , chaotic systems. Note that the computation of time evolution of a PDF has been a long-standing topic in statistical physics. For a simple Markov process, there are well-established theories based on the Fokker-Planck equation. However, it is very difficult to extend those theories to a more general problem, such as delay-time dynamical systems, or apply it to complex nonlinear systems.

Modeling of the system (1) has been extensively studied in the past, in particular, under the linearity assumptions of $\mathcal{F}$ and certain noise models (e.g., Gaussian ϵ_t and $\nu_t = 1$ in (2)). The approaches based on auto-regressive processes (Lütkepohl, 2005) and Kalman filter (Harvey, 1990) are good examples. Although these methods do estimate the predictive probability distribution and enable the computation of the forecast uncertainty, the assumptions on the noise and linearity in many cases make it challenging to model real nonlinear dynamical systems.

A recent success of deep learning created a flurry of new approaches for time series modeling and prediction. The ability of deep neural networks, such as RNN, to learn complex nonlinear spatio-temporal relationships in the data enabled these methods to outperform the classical time series approaches. For example, in the recent works of Qin et al. (2017); Hsu (2017); Dasgupta & Osogami (2017), the authors proposed different variants of the RNN-based algorithms to perform time series predictions and showed their advantage over the traditional methods. Although encouraging, these approaches lack the ability to estimate the probability distribution of the predictions since RNN is a deterministic model and unable to fully capture the stochastic nature of the data.

To enable RNN to model the stochastic properties of the data, Chung et al. (2015) augmented RNN with a latent random variable included in the hidden state and proposed to estimate the resulting model using variational inference. In a similar vein, the works of Archer et al. (2015); Krishnan et al. (2017) extend the traditional Kalman filter to handle nonlinear dynamics when the inference becomes intractable. Their approach is based on formulating the variational lower bound and optimizing it under the assumption of Gaussian posterior.

Another recent line of works enabled stochasticity in the RNN-based models by drawing a connection between Bayesian variation inference and a dropout technique. In particular, (Gal & Ghahramani, 2016) showed that the model parameter uncertainty (which then leads to uncertainty in model predictions), that traditionally was estimated using variational inference, can be approximated using a dropout method (a random removal of some connections in the network structure). The prediction uncertainty is then estimated by evaluating the model outputs at different realizations of the dropout weights. Following the ideas of Gal & Ghahramani (2016), Zhu & Laptev (2017) proposed additional ways (besides modeling the parameter uncertainty) to quantify the forecast uncertainty in RNN, which included the model mis-specification error and the inherent noise of the data.

1.1 Overview of the proposed work

We propose an RNN-model to compute the temporal evolution of a PDF, $p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$. To avoid the difficulties in directly estimating the continuous function, we use a numerical discretization technique, which converts the function approximation problem to a classification task (see Section 2.2). We note that the use of the traditional cross-entropy (CE) loss in our formulated classification problem can be problematic since it is oblivious to the class ordering. To address this, we additionally propose two regularizations for CE to account for a geometric proximity between the classes (see Sections 2.2.1 and 2.2.2). The probability distribution of one-step-ahead prediction, $p(\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t})$ can now be simply estimated from the output softmax layer of RNN (see Section 2.2), while to propagate the probability distribution farther in time, for a multiple-step forecast, we propose a sequential Monte Carlo (SMC) method (see Section 2.3). Observe, that the proposed modeling approach enables us to represent a continuous PDF of any arbitrary shape, including the ability to handle the multiplicative data noises in (2). Since the probability distribution is computed, the RNN-model can also be used for a regression task by computing the expectation (see Section 2.3). Hereafter, we use DE-RNN for the proposed RNN model, considering the similarity with the density-estimation task.

In summary, the contributions of this work are as follows: (i) formulate the classical regression problem for time series prediction as a predictive density-estimation problem, which can be solved by a classification task (ii) propose an approach to compute the time evolution of probability distribution using SMC on the distributions from DE-RNN (iii) proposed two regularizations for CE loss to capture the ordering of the classes in the discretized PDF. We evaluated the proposed algorithm on two synthetic and two real datasets, showing its advantage over the baselines. Note that DE-RNN has a direct relevance to a wide range of problems in physics and engineering, in particular, for uncertainty quantification and propagation (Zhang & Karniadakis, 2017).

2 LSTM FOR NOISY DYNAMICAL SYSTEM

In this Section we present the details of the proposed algorithm using a specific form of RNN, called Long Short-Term Memory (LSTM) network. Although in the following presentation and experiments we used LSTM, other networks, e.g., GRU (Chung et al., 2014), can be used instead.

2.1 REVIEW OF LONG SHORT-TERM MEMORY NETWORK

The Long Short-Term Memory network (LSTM) (Hochreiter & Schmidhuber, 1997; Gers et al., 2000) consists of a set of nonlinear transformations of input variables $\boldsymbol{z}_t \in \mathbb{R}^m$;

$$\text{Gating functions: } \boldsymbol{G}_{i,f,o} = \boldsymbol{\varphi}_S \circ \mathcal{L}(\boldsymbol{z}_t), \tag{3}$$

$$\text{Internal state: } \boldsymbol{s}_t = (\boldsymbol{1} - \boldsymbol{G}_f) \odot \boldsymbol{s}_{t-1} + \boldsymbol{G}_i \odot (\boldsymbol{\varphi}_T \circ \mathcal{L}(\boldsymbol{z}_t)), \tag{4}$$

$$\text{Output: } \boldsymbol{h}_t = \boldsymbol{G}_o \odot \boldsymbol{s}_t. \tag{5}$$

Here, $\boldsymbol{\varphi}_S$ and $\boldsymbol{\varphi}_T$, respectively, denote the sigmoid and hyperbolic tangent functions, $\mathcal{L}$ is the linear layer, which includes the bias, $\boldsymbol{s}_t \in \mathbb{R}^{N_c}$ is the internal state, $\boldsymbol{h}_t \in \mathbb{R}^{N_c}$ is the output of the LSTM network, N_c is the number of the LSTM units, and $\boldsymbol{a} \odot \boldsymbol{b}$ denote a component-wise multiplication.

Interesting observation can be made about equation (4). We can re-write equation (4) as

$$\boldsymbol{s}_{t+1} = [1 - f(\boldsymbol{z}_t)]\, \boldsymbol{s}_t + g(\boldsymbol{z}_t), \tag{6}$$

for some function f and g. With a simple re-scaling, this equation can be interpreted as a first-order Euler scheme for a linear dynamical system,

$$\frac{\partial \boldsymbol{s}}{\partial t} = -f(\boldsymbol{z})\boldsymbol{s} + g(\boldsymbol{z}). \tag{7}$$

Thus, LSTM can be understood as a series expansion, where a complex nonlinear dynamical system is approximated by a combination of many simpler dynamical systems.

Usually, LSTM network is supplemented by feed-forward neural networks, e.g.,

$$\boldsymbol{z}_t = \mathcal{F}_{in}(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}), \quad \boldsymbol{P}_{t+1} = \mathcal{F}_{out}(\boldsymbol{h}_t), \tag{8}$$

in which $\boldsymbol{x}_t$ is the input feature. Using (5), we can denote by Ψ_e and Ψ_d a collection of the operators from input to internal state (encoder) and from internal state to the output $\boldsymbol{o}$ (decoder):

$$\boldsymbol{s}_t = \Psi_e(\boldsymbol{x}_t, \boldsymbol{s}_{t-1}), \;\; \boldsymbol{P}_{t+1} = \Psi_d(\boldsymbol{s}_t). \tag{9}$$

2.2 DISCRETE APPROXIMATION OF PROBABILITY DENSITY FUNCTION

In this Section we first consider the problem of modeling the conditional PDF, $p(\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t})$. Although $\hat{y}_{t+1}$ has a dependence on the past trajectories of both $\hat{y}$ and $\boldsymbol{u}$, using the "state space" LSTM model argument from Section 2.1, the conditional PDF can be modeled as a Markov process

$$p(\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t}) = p(\hat{y}_{t+1}|\hat{y}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t-1}) = p(\hat{y}_{t+1}|\boldsymbol{s}_t). \tag{10}$$

Hence, to simplify the problem, we consider a task of estimating the PDF of a random variable y, given input x, i.e., $p(y|x)$. The obtained results can then be directly applied to the original problem of estimating $p(\hat{y}_{t+1}|\boldsymbol{s}_t)$.

Let $\boldsymbol{\alpha} = (\alpha_0, \cdots, \alpha_K)$ denote a set of real numbers, such that $\alpha_{i-1} < \alpha_i$ for $i = 1, \cdots, K$, which defines K disjoint intervals, $\mathcal{I}_i = (\alpha_{i-1}, \alpha_i)$. Then, a discrete probability distribution can be defined;

$$p(k|x) = \int_{\mathcal{I}_k} p(y|x)dy, \text{ for } k = 1, \ldots, K, \tag{11}$$

where it is clear that $p(k|x)$ is a numerical discretization of the continuous PDF, $p(y|x)$. Using the LSTM from Section 2.1, the discrete probability $p(k|x)$ can be modeled by the softmax layer (P) as an output of Ψ_d in (9) such that

$$p(k|x) = P_k, \text{ for } k = 1, \ldots, K. \tag{12}$$

Thus, the original problem of estimating a smooth function, $p(y|x)$, is transformed into a classification problem of estimating $p(k|x)$ in discrete domain. Obviously, the size of the bin, $|\mathcal{I}_j|$, affects the fidelity of the approximation. The effects of the bin size are presented in Section 3.1. There is a similarity between the discretization and the idea of Lin et al. (2007). However, it should be noted that the same discretization technique, often called "finite volume method", has been widely used in the numerical simulations of partial differential equations for a long time.

The discretization naturally leads to the conventional cross-entropy (CE) minimization. Suppose we have a data set, $\boldsymbol{D}_R = \{(y_i, x_i); y_i \in \mathbb{R}, x_i \in \mathbb{R}, \text{and } i = 1, \ldots, N\}$. We can define a mapping , $\mathcal{C} : \mathbb{R} \to \mathbb{N}_+$, such that $\mathcal{C}(y) = k,$ if $y \in \mathcal{I}_k$. Then, $\boldsymbol{D}_R$ can be easily converted to a new data set for target labels, $\boldsymbol{D}_C = \{(c_i, y_i, x_i); c_i \in \mathbb{N}_+, y_i \in \mathbb{R}, x_i \in \mathbb{R}, \text{and } i = 1, \ldots, N\}$, where $c_i = \mathcal{C}(y_i)$. $\boldsymbol{D}_C$ provides a training data set for the following CE minimization problem,

$$CE = -\sum_{n=1}^{N}\sum_{k=1}^{K} \delta_{c_n k} \log P_k^n = -\sum_{n=1}^{N} \log P_{c_n}^n. \tag{13}$$

Note, however, that the CE minimization does not explicitly guarantee the smoothness of the estimated distribution. Since CE loss function depends only on P_i of a correct label, $\delta_{c_n k}$, as a result, in the optimization problem every element P_i, except for the one corresponding to the correct label, P_{c_n}, is penalized in the same way, which is natural in the conventional classification tasks where a geometric proximity between the classes is not relevant. In the present study, however, the softmax layer, or class probability, is used as a discrete approximation to a smooth function. Hence, it is expected that P_{c_n} and $P_{c_n \pm 1}$ (i.e., the nearby classes) should be close to each other. To address this issue, in the following Sections 2.2.1 and 2.2.2, we propose two types of regularizations to impose the class proximity structure in the CE loss.

2.2.1 Explicit Regularization of Cross-Entropy Loss

To explicitly impose the smoothness between the classes, we propose to use a regularized cross-entropy (RCE) minimization, defined by the following loss function

$$\text{RCE} = \sum_{n=1}^{N} \left\{ \sum_{k=1}^{K} -\delta_{c_n k} \log P_k^n + \lambda \left(\boldsymbol{L}\boldsymbol{P}^n\right)^T \boldsymbol{L}\boldsymbol{P}^n \right\}, \tag{14}$$

where λ is a penalty parameter and the Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{K-2,K}$ is

$$\boldsymbol{L} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \end{bmatrix}. \tag{15}$$

RCE is analogous to the penalized maximum likelihood solution for density estimation (Silverman, 1986). Assuming a uniform bin size, $|\mathcal{I}_0| = \cdots = |\mathcal{I}_K| = \delta y$, the Laplacian of a distribution can be approximated by a Taylor expansion $p''(y|x)|_{y=\alpha_{i-1/2}} \simeq (P_{i-1} - 2P_i + P_{i+1})/\delta y^2$, where $\alpha_{i-1/2} = 0.5(\alpha_{i-1} + \alpha_i)$. Then, it is clear that

$$\left(\boldsymbol{L}\boldsymbol{P}^n\right)^T \boldsymbol{L}\boldsymbol{P}^n \sim \int \left[p''(y|x)\right]^2 dy. \tag{16}$$

In other words, RCE aims to smooth out the distribution by penalizing local minima or maxima.

2.2.2 Implicit Regularization of Cross-Entropy Loss

Alternative to adding an explicit regularization to CE, the smoothness can be achieved by enforcing a spatial correlation in the network output. Here, we use an one-dimensional convolution layer to enforce smoothness. Let $\widetilde{\boldsymbol{o}} \in \mathbb{R}^K$ be an output vector of DE-RNN, which is used as an input to the softmax layer. We can add a convolution layer, $\boldsymbol{o} \in \mathbb{R}^K$, on top of $\widetilde{\boldsymbol{o}}$, such that

$$o_i = \sum_{j=1}^{K} \frac{1}{h} \exp\left[-\frac{1}{2}\left(\frac{i-j}{h}\right)^2\right] \widetilde{o}_j, \quad \text{for } i = 1, \cdots, K, \tag{17}$$

where the penalty parameter h determines the smoothness of the estimated distribution. Using (17), the LSTM model can now be trained by the standard CE. The implicit regularization, here we call convolution CE (CCE), is analogous to a kernel density estimation.

2.3 COMPUTING TIME EVOLUTION OF PROBABILITY DISTRIBUTION

In the next-step prediction of DE-RNN, the inputs are $(\hat{y}_t, \boldsymbol{u}_t)$, and the output is the probability distribution,

$$\boldsymbol{P}_{t+1} = \Psi_d(\boldsymbol{s}_t) = \Psi_d \circ \Psi_e(\hat{y}_t, \boldsymbol{u}_t, \boldsymbol{s}_{t-1}).$$

Note that $\boldsymbol{D}_C$ is used only in the training stage. Then, the moments of the predictive distribution can be easily evaluated, e.g.,

$$E[\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t}] = \boldsymbol{\alpha}_{1/2}^T \boldsymbol{P}_{t+1},\ Var[\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t}] = (\boldsymbol{\alpha}_{1/2}^2)^T \boldsymbol{P}_{t+1} - E[\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t}]^2, \tag{18}$$

where $\boldsymbol{\alpha}_{1/2} = (\alpha_{1/2}, \alpha_{1+1/2}, \cdots, \alpha_{K-1/2})^T$, $\boldsymbol{\alpha}_{1/2}^2 = \boldsymbol{\alpha}_{1/2} \odot \boldsymbol{\alpha}_{1/2}$, and $\alpha_{i-1/2} = 0.5(\alpha_{i-1} + \alpha_i)$.

Next, we consider a multiple-step forecast, which corresponds to computing a temporal evolution of the probability distribution, i.e., $p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$ for $n > 1$.

Applying the results of Section 2.2, once the distribution of $\hat{y}_{t+1}$ in (10) is computed, the distribution of $\hat{y}_{t+2}$ can be similarly obtained as $p(\hat{y}_{t+2}|\boldsymbol{s}_{t+1})$. Observe that $\boldsymbol{s}_{t+1}$ is computed from a deterministic function of $\boldsymbol{s}_t$, $\boldsymbol{u}_{t+1}$, and $\hat{y}_{t+1}$, i.e.,

$$\boldsymbol{s}_{t+1} = \Psi_e(\hat{y}_{t+1}, \boldsymbol{u}_{t+1}, \boldsymbol{s}_t).$$

Here, $\boldsymbol{u}_{t+1}$ and $\boldsymbol{s}_t$ are already known, while $\hat{y}_{t+1}$ is a random variable, whose distribution $p(\hat{y}_{t+1}|\boldsymbol{s}_t)$ is computed from the deterministic function $\Psi_d(\boldsymbol{s}_t)$. Then, $\boldsymbol{s}_{t+1}$ is also a random variable. The distribution, $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{u}_{t+1})$, can be obtained by applying a change of variables on $p(\hat{y}_{t+1}|\boldsymbol{s}_t)$ with a nonlinear mapping Ψ_e. Repeating this process, the multiple-step-ahead predictive distribution can therefore be computed as

$$p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1}) = \int \cdots \int p(\hat{y}_{t+n}|\boldsymbol{s}_{t+n-1}) \prod_{i=1}^{n-1} p(\boldsymbol{s}_{t+i}|\boldsymbol{s}_{t+i-1}, \boldsymbol{u}_{t+i})\, d\boldsymbol{s}_{t+i}. \tag{19}$$

Since the high dimensional integration in (19) is intractable, we propose to approximate it by a sequential Monte Carlo method. The Monte Carlo procedure is outlined in Algorithm 1.

Algorithm 1 Sequential Monte Carlo method for LSTM multi-step-ahead prediction

Input: $\widehat{\boldsymbol{Y}}_{0:t}$, $\boldsymbol{U}_{0:t}$, number of Monte Carlo samples, N_s, and forecast horizon n
Output: $p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$ (density estimation from $\hat{\boldsymbol{y}}_{t+n}$)
Initialization: Set LSTM states to $\boldsymbol{s}_0 = \boldsymbol{h}_0 = \boldsymbol{0}$
Perform a sequential update of LSTM up to time t from the noisy observations ($\widehat{\boldsymbol{Y}}_{0:t}$).

$$\boldsymbol{s}_i = \Psi_e(\hat{y}_i, \boldsymbol{u}_i, \boldsymbol{s}_{i-1}) \ \text{ for } \ i = 1, \cdots, t.$$

Make N_s replicas of the internal state, $\boldsymbol{s}_t^1 = \cdots = \boldsymbol{s}_t^{N_s} = \boldsymbol{s}_t$.
repeat
Compute the predictive distribution of $\hat{y}_{t+1}^i$ for each sample

$$\boldsymbol{P}_{t+1}^i = \Psi_d(\boldsymbol{s}_t^i), \ \text{ for } \ i = 1, \cdots, N_s.$$

Sample the target variable at $t+1$, $\hat{y}_{t+1}^i$, from the distribution
1. Sample the class label from the discrete distribution: $c^i \sim \boldsymbol{P}_{t+1}^i$
2. Sample $\hat{y}_{t+1}^i$ in $\mathcal{I}_c$: $\hat{y}_{t+1}^i \sim \mathcal{U}(\alpha_{c^i-1}, \alpha_{c^i})$

Update the internal state of LSTM

$$\boldsymbol{s}_{t+1}^{i+1} = \Psi_e(\hat{y}_{t+1}^i, \boldsymbol{u}_{t+1}, \boldsymbol{s}_t^i).$$

until (all $\hat{\boldsymbol{y}}_{t+n}$ are sampled)

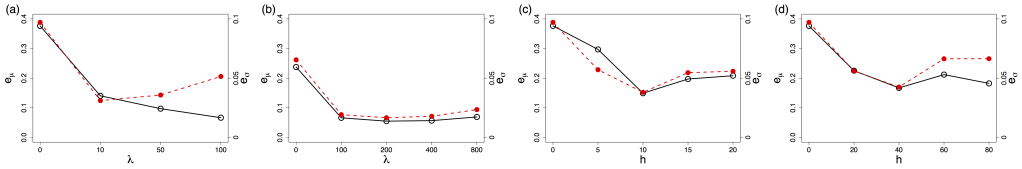

Figure 1: NRMSE of the next-step prediction of the CIR process by RCE (a,b) and CCE (c,d). The bin size is (a,c) $\delta y = 0.08sd[y]$ and (b,d) $0.04sd[y]$. The hollow circles (○) denote NRMSE in the expectation (e_μ) and the solid circles (●) are the standard deviation (e_σ).

3 EXPERIMENTS

In this section, DE-RNN is tested against two synthetic and two real data sets. The LSTM architecture used in all of the numerical experiments is identical. Two feed-forward networks are used before and after the LSTM;

$$\boldsymbol{z}_t = \mathcal{L}\left(\varphi_T \circ \mathcal{L}(y_t, \boldsymbol{u}_t) + \boldsymbol{h}_{t-1}\right), \qquad \boldsymbol{P}_{t+1} = \varphi_{SM} \circ \mathcal{L}(\varphi_T \circ \mathcal{L}(\varphi_{SP} \circ \mathcal{L}(\boldsymbol{h}_t))), \tag{20}$$

in which φ_{SP} and φ_{SM} denote the softplus and softmax functions, respectively. The size of the bins is kept uniform, i.e., $|\mathcal{I}_1| = \cdots = |\mathcal{I}_K| = \delta y$. The LSTM is trained by using ADAM Kingma & Ba (2015) with a minibath size of 20 and a learning rate of $\eta = 10^{-3}$.

3.1 COX-INGERSOLL-ROSS PROCESS

First, we consider a modified Cox-Ingersoll-Ross (CIR) process, which is represented by the following stochastic differential equation,

$$dy(t) = -0.5y(t)dt + \sqrt{0.5 + |y(t)|}dW, \tag{21}$$

in which W is the Weiner process. The original CIR process is used to model the valuation of interest rate derivatives in finance. Equation (21) is solved by the forward Euler method with the time step size $\delta t = 0.1$. The simulation is performed for $T = (0, 160000]\delta t$ to generate the training data and $T = (160000, 162000]\delta t$ is used for the testing. Note that the noise component of CIR is multiplicative, which depends on $y(t)$.

The experiments are performed for two different bin sizes, $dy = 0.08sd[y]$ and $0.04sd[y]$, in which $sd[y]$ denotes the standard deviation of the data. The DE-RNN has 64 LSTM cells. Figure 1 shows the errors in the expectation and the standard deviation with respect to the analytical solution;

$$E_{y\sim p_T}[y_{t+1}|y_t] = y_t \exp(-0.5\delta t), \qquad sd_{y\sim p_T}[y_{t+1}|y_t] = \sqrt{(0.5 + |y_t|)\delta_t}. \tag{22}$$

Here, p_T denotes the true distribution of the CIR process. The normalized root mean-square errors (NRMSE) are defined as

$$e_\mu = \frac{\langle (E_{y\sim p_L}[y_{t+1}|y_t] - E_{y\sim p_T}[y_{t+1}|y_t])^2\rangle^{1/2}}{\langle (y_t - E_{y\sim p_T}[y_{t+1}|y_t])^2\rangle^{1/2}}, \tag{23}$$

$$e_\sigma = \frac{\langle (sd_{y\sim p_L}[y_{t+1}|y_t] - sd_{y\sim p_T}[y_{t+1}|y_t])^2\rangle^{1/2}}{sd[y]}, \tag{24}$$

in which $\langle\cdot\rangle$ denotes an average over the testing data and p_L is the distribution from the LSTM. The error in the expectation is normalized against a zeroth-order prediction, which assumes $y_{t+1} = y_t$.

In Figure 1, it is clearly shown that the prediction results are improved when a regularization is used to impose a smoothness condition. Comparing Figures 1 (a) and (b), for RCE, e_μ and e_σ become smaller when a smaller δy is used. As expected, e_σ increases when λ is large. But, for the smaller bin size, $\delta y = 0.04sd[y]$, both e_μ and e_σ are not so sensitive to λ. Similar to RCE, e_μ and e_σ for CCE decrease at first as the penalty parameter h increases. However, in general, RCE provides a better prediction compared to CCE.

NRMEs are listed in Table 1. For a comparison, the predictions by AR(1) and KF are shown. The CIR process is essentially a first-order autoregressive process. So, it is not surprising to see that

Table 1: NRMSE of the mean and standard deviation of the next-step prediction. The DE-RNN results are compared with the first-order autoregressive model (AR) and Kalman filter (KF).

	CE	RCE	CCE	AR(1)	KF
e_μ	0.238	0.0549	0.149	0.029	0.029
e_σ	0.066	0.017	0.038	0.228	0.228

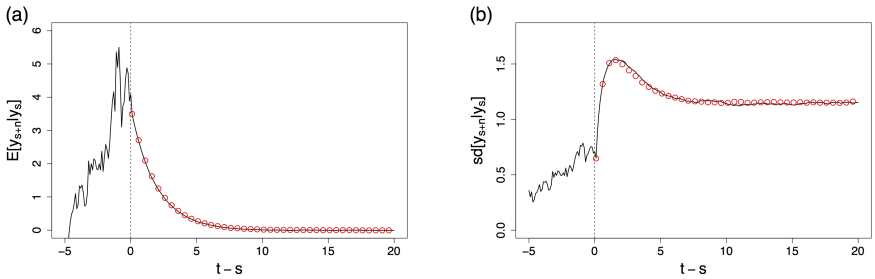

Figure 2: 200-step forecast of (a) expectation and (b) standard deviation of the CIR process. The circles denote the solution of Eqn (21) from a Monte Carlo method with 10^7 samples.

AR(1) and KF, which are designed for the first-order AR process, outperforms DE-RNN for the prediction of the expectation. However, e_σ of AR(1) and KF are much larger than that of DE-RNN, because those models assume an additive noise. Note that e_σ of RCE and CCE are less than 4%, suggesting that DE-RNN can model the complex noise process very well.

In Figure 2, a 200-step forecast by DE-RNN is compared with a Monte-Carlo solution of equation (21). DE-RNN is trained with $\delta y = 0.04sd[y]$ and $\lambda = 200$. For the DE-RNN forecast, the testing data is supplied to DE-RNN for the first 100 time steps, i.e., for $t = -10$ to $t = 0$, and the SMC multiple-step forecast is performed for the next 200 time steps with 20,000 samples. It is shown that the multiple-step forecast by DE-RNN agrees very well with the MC solution of the CIR process. Note that, in Figure 2 (b), the noise process, as reflected in $sd[y_t]$, is a function of y_t, and hence the multi-step forecast of the noise increases rapidly first and then decreases before reaching a plateau. The SMC forecast can accurately capture the behavior. Such kind of behavior can not be represented if a simple additive noise is assumed.

3.2 Mackey-Glass time series

For the next test, we applied DE-RNN for a time series generated from the Mackey-Galss equation (Mackey & Glass, 1977);

$$\frac{dy}{dt} = \frac{\alpha y(t-\tau)}{1 + y^\beta(t-\tau)} - \gamma y(t). \tag{25}$$

We use the parameters adopted from Gers (2001), $\alpha = 0.2$, $\beta = 10$, $\gamma = 0.1$, and $\tau = 17$.

The Mackey-Glass equation is solved by using a third-order Adams-Bashforth method with a time step size of 0.02. The time series is generated by down-sampling, such that the time interval between consecutive data is $\delta t = 1$. A noisy observation is made by adding a white noise;

$$\hat{y}_t = y_t + \epsilon_t.$$

ϵ_t is a zero-mean Gaussian random variable with the noise level $sd[\epsilon_t] = 0.3sd[y]$. A time series of the length $1.6 \times 10^5 \delta t$ is generated for the model trainig and another time series of length $2 \times 10^3 \delta t$ is made for the validation. DE-RNN is trained for $\delta y = 0.04sd[y]$ and consists of 128 LSTM cells.

Figure 3 (a) shows the noisy observation and the expectation of the next-step prediction, $E[\hat{y}_{t+1}|\hat{y}_t]$, in a phase space. It is shown that DE-RNN can filter out the noise and reconstruct the original dynamics accurately. Even though the noisy data are used as an input, $E[\hat{y}_{t+1}|\hat{y}_t]$ accurately represents the original attractor of the chaotic system, indicating a strong de-noising capability of the LSTM.

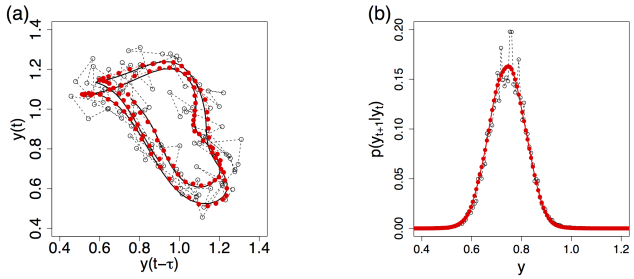

Figure 3: (a) Next-step prediction (•) and the noisy observation (○) for the Mackey-Glass equation. The solid line denotes the ground truth, $y(t)$. (b) The next-step probability distribution, $p(\hat{y}_{t+1}|\hat{y}_t)$, from the standard CE (○) and CCE (•) with $h = 5$.

Table 2: NRMSEs of the Mackey-Galss time series. DE-RNN results are compared with autoregressive integrated moving average (ARIMA) and Kalman filter (KF).

	RCE				CCE		ARIMA	KF
	$\lambda = 0$	$\lambda = 50$	$\lambda = 100$	$\lambda = 200$	$h = 5$	$h = 10$		
e_μ	0.143	0.141	0.143	0.142	0.133	0.143	0.668	0.916
e_σ	0.032	0.023	0.027	0.038	0.013	0.020	0.191	0.351

The estimated probability distribution is shown in Figure 3 (b). Without a regularization, the standard CE results in a noisy distribution, while the distribution from CCE shows a smooth Gaussian shape.

The prediction errors are shown in table 2. NRMSEs are defined as,

$$e_\mu = \frac{\langle (E[\hat{y}_{t+1}|\hat{y}_t] - y_{t+1})^2 \rangle^{1/2}}{\langle (\hat{y}_t - y_{t+1})^2 \rangle^{1/2}}, \qquad e_\sigma = \frac{sd[\hat{y}_{t+1}|\hat{y}_t]}{sd[\epsilon_t]} - 1, \tag{26}$$

in which $dy_t = y_{t+1} - y_t$. NRMSEs are computed with respect to the ground truth. Again, e_μ compares the prediction error to the zeroth-order prediction. In this example, the errors are not so sensitive to the regularization parameters. The best result is achieved by CCE. DE-RNN can make a very good estimation of the noise. The error in the noise component, e_σ, is only $2\% \sim 5\%$. Unlike the CIR process, NRMSEs from KF and ARIMA are much larger than those of DE-RNN. Because the underlying process is a delay-time nonlinear dynamical system, those linear models can not accurately approximate the complex dynamics.

A multiple-step forecast of the Mackey-Glass time series is shown in Figure 4. In the validation time series, the observations in $t \in [1, 100]\delta t$ are supplied to the DE-RNN to develop the internal state, and a 500-step forecast is made for $t \in [101, 600]\delta t$. In Figure 4 (a), it is shown that a multiple-step forecast by a standard regression LSTM approximates $y(t)$ very well initially, e.g, for $t < 80\delta t$, but

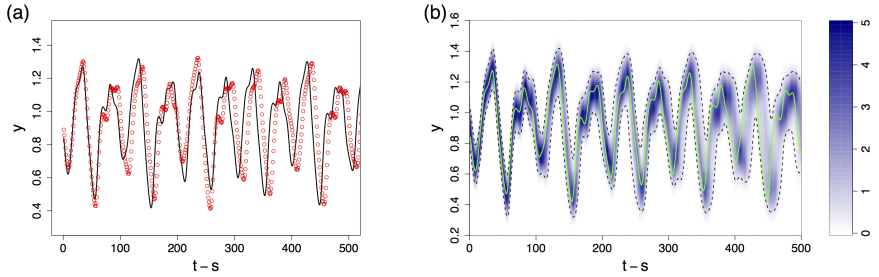

Figure 4: (a) 500-step forecast by a regression LSTM (○) and the ground truth (—). (b) The color contours denote a 500-step forecast of the probability distribution, $p(\hat{y}_{n+s}|\hat{y}_s)$, and the dashed lines are 95%-CI. The ground truth is shown as the solid line (—).

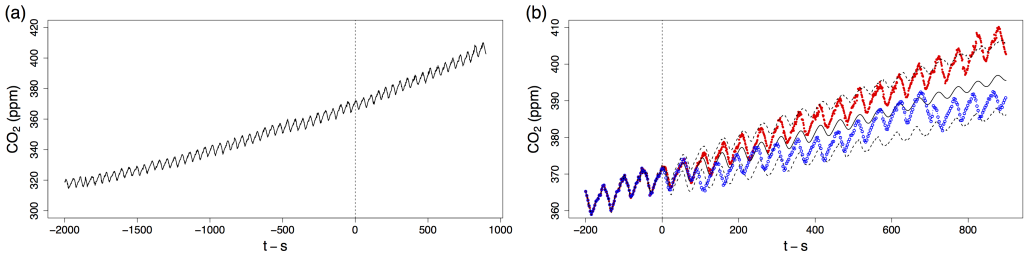

Figure 5: (a) Mauna Loa CO_2 observation. The vertical line denotes the boundary of the training and testing data. (b) 17-year forecast of the CO_2 concentration: Apr-01-2000 Sep-23-2017. The solid and dashed lines denote the expectation and 95%-CI, respectively. The observation is shown as the solid circles (●) and a regression LSTM is the hollow circles (○). The time unit is a week.

eventually diverges for larger t. Because of the Mackey-Glass time series is chaotic, theoretically it is impossible to make a long time forecast. But, in the DE-RNN forecast, $y(t)$ is bounded by the 95%-confidence interval (CI) even for the 500-step forecast. Note that the uncertainty, denoted by 95%-CI grows only at a very mild rate in time. In fact, it is observed that CI is not a monotonic function of time. In DE-RNN, the 95%-CI may grow or decrease following the dynamics of the system, while for the conventional time series models, such as ARIMA and KF, the uncertainty is a non-decreasing function of time.

3.3 Mauna Loa CO_2 observation

In this experiments, DE-RNN is tested against the atmospheric CO_2 observation at Mauna Loa Observatory, Hawaii (Keeling et al., 2001). The CO_2 data set consists of weekly-average atmospheric CO_2 concentration from Mar-29-1958 to Sep-23-2017 (Figure 5 a). The data from Mar-29-1958 to Apr-01-2000 is used to train DE-RNN and a 17-year forecast is made from Apr-01-2000 to Sep-23-2017. This CO_2 data has been used in previous studies (Gal & Ghahramani, 2016; Rasmussen & Williams, 2006). In DE-RNN, 64 LSTM cells and $\delta y = 0.1sd[dy]$ are used.

The 17-year DE-RNN forecast, with 1,000 MC samples, is shown in Figure 5 (b). DE-RNN well represents the growing trend and the oscillatory patten of the CO_2 data. The CO_2 data is non-stationary, where the rate of increase of CO_2 is an increasing function of time. Since DE-RNN is trained against the history data, where the rate of CO_2 increase is smaller than the current, it is expected that the forecast will underestimate the future CO_2. $E[\hat{y}_{n+s}|\hat{y}_s]$ agrees very well with the observation for the first 200 weeks, but eventually underestimates CO_2 concentration. It is interesting to observe that the upper bound of the 95%-CI grows more rapidly than the expectation and provides a good approximation of the observation. For a comparison, the forecast by a regression LSTM is also shown. Similar to the chaotic Mackey-Glass time series, the regression LSTM makes a good prediction for a short time, e.g., $t < 100$ weeks, but eventually diverges from the observation. Note that the lower bound of 95%-CI encompasses the regression LSTM.

3.4 CPU temperature forecast

In the last experiment, IBM Power System S822LC and NAS Parallel Benchmark (NPB) are used to generate the temperature trace. Figure 6 (a) shows the temperature of a CPU. The temperature sensor generates a discrete data, which has a resolution of 1°C. The CPU temperature is controlled by three major parameters; CPU frequency, CPU utilization, and cooling fan speed. In this experiment, we have randomized the CPU frequencies and job arrival time to mimic the real workload behavior, while the fan speed is fixed to 3300RPM. The time step size is $\delta t = 2$ seconds. Accurate forecast of CPU temperature for a future workload scenario is essential in developing an energy-efficient control strategy for the thermal management of a cloud system.

Figure 6 (c) and (d) show multiple-step forecasts of the CPU temperature by RCE and a regression LSTM, respectively. The bin size is $\delta y = 0.18$°C, which is smaller than the sensor resolution. In the forecast, the future control parameters are given to DE-RNN. In other words, DE-RNN predicts the probability distribution of future temperature with respect to a control scenario, i.e.,

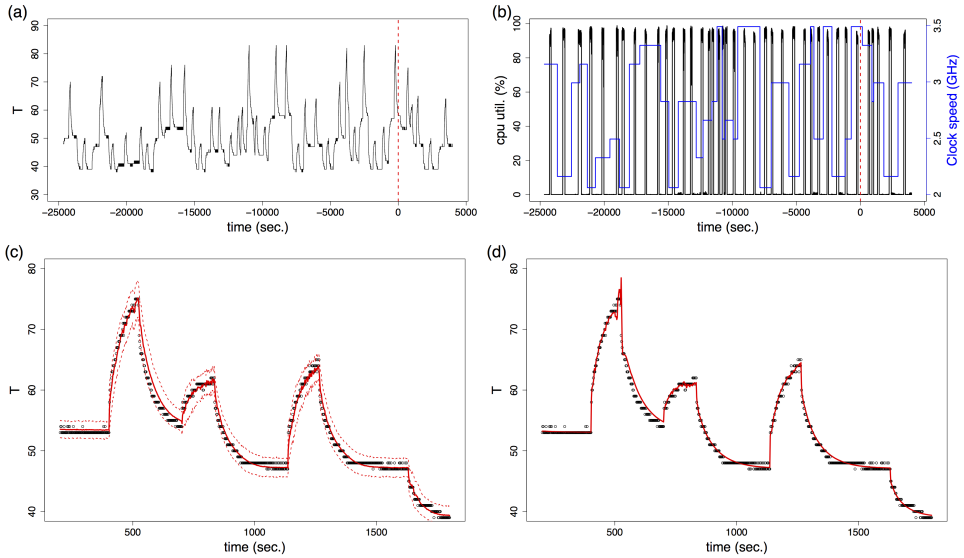

Figure 6: (a) Temperature of a CPU in °C. (b) Control parameters; CPU utilization (black) and Clock speed (blue). Multiple-step forecasts by (c) RCE and (d) regression LSTM. The solid and dashed lines in (c) denote $E[\hat{y}_{n+s}|\hat{y}_s]$ and 95%-CI, respectively. The circles are the observations.

Table 3: l_∞ error for RCE, CCE and regression LSTM in °C.

	RCE	CCE	LSTM
$\|E[\hat{y}_{n+s}\|\hat{y}_s] - \hat{y}_{n+s}\|_\infty$	4.61	2.70	9.48

$p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$. The forecast is made by using 5,000 Monte Carlo samples. Here, 1800-step forecast is made, $t = 0 \sim 3,600$ sec. and only the results in $t \in (50, 1800)$ sec. is shown. While the regression LSTM makes a very noisy prediction near the local peak temperature at $t \simeq 500$, RCE provides a much more stable forecast. Table 3 shows the l_∞-errors, i.e., maximum absolute difference. The maximum error is observed near the peak temperature at $t \simeq 500$. ARIMA and KF are also tested for the multiple-step forecast, but the results are not shown because their performance is much worse than the LSTMs. The changes in the temperature are associated with step changes of some control parameters. Such abrupt transitions seem to cause the large oscillation in the regression LSTM prediction. But, for RCE or CCE, the prediction is made from an ensemble of Monte Carlo samples, which makes it more robust to such abrupt changes. Also, note that for $t < 200$ sec., RCE prediction ($\simeq 53.4$°C) is in between the two discrete integer levels, 53°C and 54°C, which correctly reflects the uncertainty in the measurement, while the regression LSTM ($\simeq 53.1$°C) more closely follows one of the two levels.

4 Concluding Remarks

We present DE-RNN to compute the time evolution of a probability distribution for complex time series data. DE-RNN employs LSTM to learn multiscale, nonlinear dynamics from the noisy observations, which is supplemented by a softmax layer to approximate a probability density function. To assign probability to the softmax output, we use a mapping from $\mathbb{R}$ to $\mathbb{N}_+$, which leads to a cross-entropy minimization problem. To impose a geometric structure in the distribution, two regularization strategies are proposed. The regularized cross-entropy method is analogous to the penalized maximum likelihood estimate for the density estimation, while the convolution cross-entropy method is motivated by the kernel density estimation. The proposed algorithm is validated against two synthetic data set, for which we can compare with the analytical solutions, and two real data sets. In this study, for simplicity, the problem is formulated for a univariate time series. But, it is straightforward to extend the methodology to a multivariate time series.

REFERENCES

E. Archer, M. Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville C, and Y. Bengio. A recurrent latent variable model for sequential data. In *Advances in Neural Information Processing Systems*, pp. 2980–2988, 2015.

S. Dasgupta and T. Osogami. Nonlinear dynamic boltzmann machines for time-series prediction. In *AAAI Conference on Artificial Intelligence*, 2017.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pp. 1050–1059, 2016.

F. A. Gers. *Ph.D Thesis*. EPFL, 2001.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Comput.*, 12:2451 – 2471, 2000.

A. C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9:1735 – 1780, 1997.

D. Hsu. Time series forecasting based on augmented long short-term memory. *arXiv preprint arXiv:1707.00666*, 2017.

C. D. Keeling, S. C. Piper, R. B. Bacastow, M. Wahlen, T. P. Whorf, M. Heimann, and H. A. Meijer. Exchanges of atmospheric CO_2 and $^{13}CO_2$ with the terrestrial biosphere and oceans from 1978 to 2000. I. Global aspects. In *SIO Reference Series*, pp. No. 01–06. Scripps Institution of Oceanography, 2001.

D. P. Kingma and J. L. Ba. ADAM: A method for stochastic optimization. In *3rd International Conference on Learning Representation, San Diego, CA, USA*, 2015.

R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *AAAI Conference on Artificial Intelligence*, 2017.

H. Lasi, P. Fettke, H. Kemper, T. Feldand, and M. Hoffmann. Industry 4.0. *Business & Information Systems Engineering*, 6(4):239–242, 2014.

J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

H. Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005.

M. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197: 287–289, 1977.

Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 2627–2633, 2017.

C. E. Rasmussen and K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman & Hall, 1986.

W.-X. Wang, Y.-C. Lai, and C. Grebogi. Data based identification and prediction of nonlinear and complex dynamical systems. *Phys. Reports*, 644:1–76, 2016.

Z. Zhang and G. E. Karniadakis. *Numerical methods for stochastic partial differential equations with white noise*. Springer, 2017.

L. Zhu and N. Laptev. Deep and confident prediction for time series at Uber. *arXiv preprint arXiv:1709.01907*, 2017.

A APENDIX

In this Section we present a few extra details of the proposed DE-RNN algorithm. Figure 7 shows the architecture of DE-RNN as was used during the experiments (presented in Section 3). In Figure 8 we show the process of computing one-step-ahead predictions (presented in Section 2.2). Note that since DE-RNN estimates approximation of the predictive probability distribution, the predicted value, e.g., for time step $t+1$, is the discrete approximation of $E[\hat{y}_{t+1}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t}]$, i.e., the expectation of $\hat{y}_{t+1}$ given all the observations and control inputs up to time t. Finally, in Figure 9 we show the details of the multi-step forecast (presented in Section 2.3, in Algorithm 1). Using sequential Monte Carlo method, the discrete approximation of the predictive distribution $p(\hat{y}_{t+n}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+n-1})$ is estimated using N_s samples.

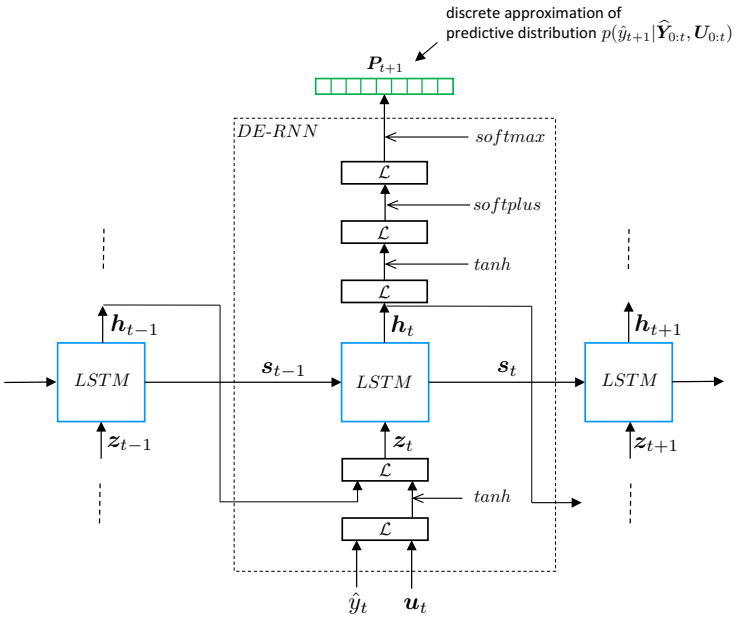

Figure 7: Architecture of the DE-RNN algorithm.

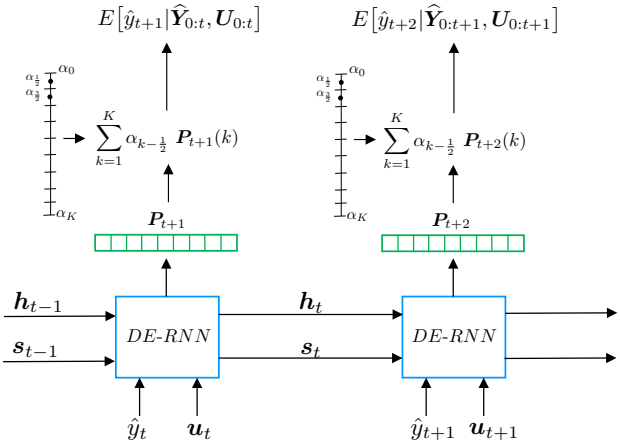

Figure 8: Details of the computation for the one-step-ahead predictions. At a given step the model computes a discrete approximation of the expectation for the next step observation.

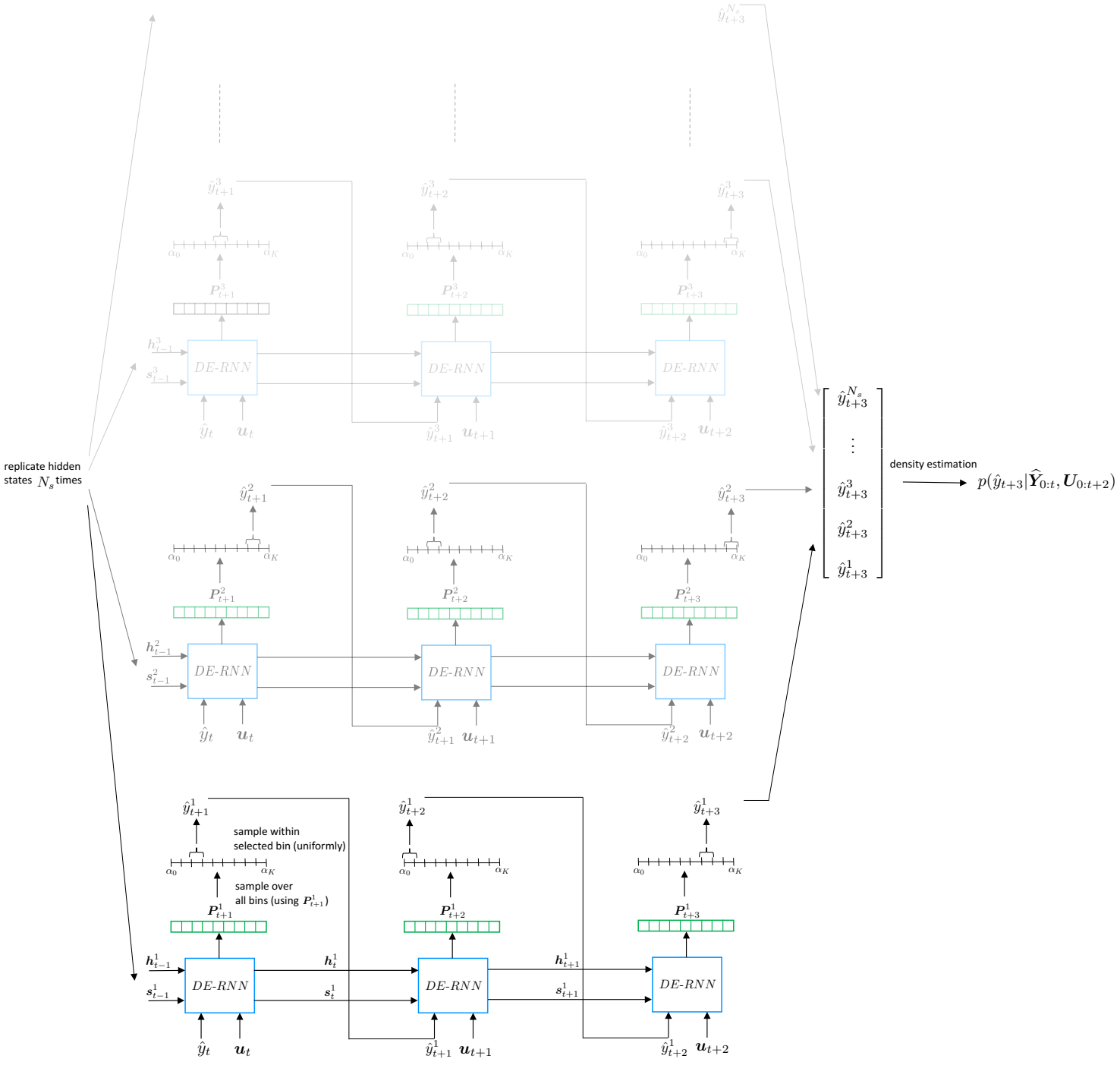

Figure 9: Visualization of the multi-step-ahead forecast made by DE-RNN. The shown illustration is for computing 3-step-ahead predictive probability distribution $p(\hat{y}_{t+3}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+2})$. Ns independent DE-RNN instances, $i = 1, \ldots, N_s$, are initialized identically with the hidden states from the time step of the last observation. Each DE-RNN instance i uses a two-stage sampling procedure to propagate its own value of $\hat{y}^i$ in time. At the end, N_s samples of $\hat{y}^i_{t+3}$ are used to compute a discrete approximation of the predictive distribution $p(\hat{y}_{t+3}|\widehat{\boldsymbol{Y}}_{0:t}, \boldsymbol{U}_{0:t+2})$.