Time Series Forecasting Based on Augmented Long Short-Term Memory

Daniel Hsu

July 7, 2017

Abstract

In this paper, we use recurrent autoencoder model to predict the time series in single and multiple steps ahead. Previous prediction methods, such as recurrent neural network (RNN) and deep belief network (DBN) models, cannot learn long term dependencies. And conventional long short-term memory (LSTM) model doesn't remember recent inputs. Combining LSTM and autoencoder (AE), the proposed model can capture long-term dependencies across data points and uses features extracted from recent observations for augmenting LSTM at the same time. Based on comprehensive experiments, we show that the proposed methods significantly improves the state-of-art performance on chaotic time series benchmark and also has better performance on real-world data. Both single-output and multiple-output predictions are investigated.

1 Introduction

Time series forecasting and modeling is an important interdisciplinary field of research, involving among others Computer Sciences, Statistics, and Econometrics. Made popular by Box and Jenkins [1] in the 1970s, traditional modeling procedures combine linear autoregression (AR) and moving average. But, since data are nowadays abundantly available, often complex patterns that are not linear can be extracted. So, the need for nonlinear forecasting procedures arises.

Recently, neural networks with deep architectures have proven to be very successful in image, video, audio and language leaning tasks [6]. In time series forecasting area, though traditionally shallow neural networks are generally adopted, the deep neural networks have also aroused enormous interests among researchers. Deep belief networks (DBN) are frequently employed in current short-term traffic forecasting [7] [8], and pre-training strategies with unsupervised learning algorithms such as Restricted Boltzmann machine (RBM) [9] and Stacked AutoEncoder (SAE) [11] are also used. However, these deep architectures can not capture the long dependencies across data points which are beyond input observations.

RNNs are particularly suitable for modeling dynamical systems as they operate on input information as well as a trace of previously acquired information (due to recurrent connections) allowing for direct processing of temporal dependencies. RNNs can be employed for a wide range of tasks as they inherit their flexibility from plain neural networks. Among all RNN architectures, the most successful one to characterize long-term memory is the long short-term memory network (LSTM) [3], which learns both short-term and long-term memory by enforcing constant error flow through the designed cell state.

Recent advances in deep learning, especially recurrent neural network (RNN) and long short-term memory (LSTM) models [5] [4] [13], have been proposed to tackle this problem. However,

these models still have some disadvantages. Especially LSTM model cannot work well on cases where the prediction is primarily based on recent past observations [5].

In this work, we investigate time series forecasting by combining AutoEncoder (AE) and LSTM. In this model, the input observations are first encoded to latent variables, which is equivalent to feature extraction. In the LSTM cell, the input observations and latent variables are incorporated into the state transition. The decoder is a multi-layer neural network which maps hidden state of LSTM and latent variables of input into the predicted values. We term the proposed model as augmented long short-term memory (A-LSTM). The LSTM cell, encoder and decoder are trained together by ADAM [20]. This model can not only capture long-term dependencies, but also augment the prediction by the extracted features from recent observations at the same time.

The experiments show that this model can not only improve the performance of benchmarks on chaotic time series prediction, but also performs better on real-world data than previous methods such as DBN pre-trained with RBM [9], SAE [11] and LSTM models [13]. Moreover, with 5-step and 10-step experiments, the performance on multi-output prediction task doesn't degrade as the number of output increases, showing the proposed method has strong potential for scalability.

2 Preliminary

2.1 Autoencoder

The Autoencoder (AE) was first introduced as a dimension-reduction model [19]. An autoencoder takes an input vector $\boldsymbol{x}$ and transforms it into a latent representation $\boldsymbol{z}$. The transformation, typically referred as encoder, follows the following equation:

$$\boldsymbol{z} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ correspond to the weighs and bias in the encoder, and σ is the sigmoid function.

The resulting latent representation $\boldsymbol{z}$ is then mapped back into the reconstructed feature space $\boldsymbol{y}$ in the decoder as follows:

$$\boldsymbol{y} = \sigma(\boldsymbol{W}'\boldsymbol{z} + \boldsymbol{b}')$$

where $\boldsymbol{W}'$ and $\boldsymbol{b}'$ correspond to the weighs and bias in the decoder. The autoencoder is trained by minimizing the reconstruction error $\|\boldsymbol{y} - \boldsymbol{x}\|$. Multiple autoencoders can be connected to construct stacked autoencoder [11], which can be used to learn multiple levels of non-linear features. In this work, we utilize AE to extractor features from input observations, which can make LSTM learn more efficiently.

2.2 Recurrent Neural Network

RNNs are discrete-time statespace models trainable by specialized weight adaptation algorithms. The input to RNN is a variable-length sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ which can be recursively processed. And when processing each symbol, RNN maintains its internal hidden state $\boldsymbol{h}$. The operation of RNN at each timestep t can be formulated as

$$\boldsymbol{h}_t = f_\theta(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$$

where f is the deterministic state transition function and θ is the parameter of f. The output of RNN is computed using the following equation:

$$\boldsymbol{y}_t = g_\varphi(\boldsymbol{h}_t) \tag{1}$$

where function g can be modeled as a neural network with weights φ. In implementation, the function f can be realized by long short-term memory [3].

Although traditional RNN exhibits a superior capability of modeling nonlinear time series problems in an effective fashion, there are still several issues to be addressed [2]:

- Traditional RNNs cannot train the time series with very long time lags, which is commonly seen in real-world datasets.
- Traditional RNNs rely on the predetermined time lags to learn the temporal sequence processing, but it is difficult to find the optimal window size in an automatic way.

2.3 Long Short-Term Memory

To overcome the aforementioned disadvantages of traditional RNNs, Long Short-Term Memory (LSTM) neural network is proposed in this study to predict time series in single-step and multi-step ahead. LSTM was initially introduced in [3] with the objective of modeling long-term dependencies and determining the optimal time lag for time series problems. A LSTM is composed of one input layer, one recurrent hidden layer, and one output layer. The basic unit in the hidden layer is memory block, containing memory cells with self-connections memorizing the temporal state, and a pair of adaptive, multiplicative gating units to control information flow in the block. It also has input gate and output gate controlling the input and output activations into the block.

The memory cell is primarily a recurrently self-connected linear unit, called Constant Error Carousel (CEC), and the cell state is represented by the activation of the CEC. Because of CEC, the multiplicative gates can learn when to open and close. Then by keeping the network error constant, the vanishing gradient problem can be solved in LSTM. Moreover, a forget gate was added to the memory cell, which can prevent the gradient from exploding when learning long time series. The operation and structure of LSTM can be described as below:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i \boldsymbol{x}_t + \boldsymbol{U}_i \boldsymbol{m}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{s}_t &= \tanh(\boldsymbol{W}_s \boldsymbol{x}_t + \boldsymbol{U}_s \boldsymbol{m}_{t-1} + \boldsymbol{b}_s) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{m}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{y}_t &= \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{m}_{t-1} + \boldsymbol{b}_o) \\
\boldsymbol{c}_t &= \boldsymbol{c}_{t-1} \circ \boldsymbol{f}_t + \boldsymbol{s}_t \circ \boldsymbol{i}_t \\
\boldsymbol{m}_t &= \boldsymbol{s}_t \circ \boldsymbol{o}_t
\end{aligned} \tag{2}$$

where $\boldsymbol{i}_t, \boldsymbol{f}_t$ and $\boldsymbol{y}_t$ are denoted as input gate, forget gate and output gate at time t respectively, and $\boldsymbol{m}_t$ and $\boldsymbol{c}_t$ represent the hidden state and cell state of the memory cell at time t.

3 Related Work and Motivation

Driven by the recent success of deep learning [6], several different Deep Learning approaches can be found in the literature for performing time series predictions. For example, deep belief networks are used in the work of [9] along with restricted Boltzman machine (RBM). [10] also compares the performance of Deep Belief Networks with that of Stacked Denoising Autoencoders. This last type of network is also employed by [11] to predict the temperature of an indoor environment. Another application of Time-Series forecasting can be found in [8], which uses Stacked Autoencoders (SAE) to predict the flow of traffic from a big data dataset. However, as compared in [12], deep learning models such as RBM and SAE perform worse than LSTM

because they cannot capture long-term dependencies across data points, and fixed size window size is another factor for suboptimal performance.

LSTM [3] is another learning structure often used recently in time series prediction. [5] first used LSTM to predict chaotic time series. In [4], an LSTM sequence-to-sequence model was used to predict next values. A survey [18] reviews many applications of LSTM and other RNN architectures to short-term load forecasting problem. However, the sequence-to-sequence RNN models [4], aiming at predicting a sequence of future values on the basis of a sequence of past values, can not handle very long sequences and model periods in time series [15]. And every input sequence was padded to the same length. As indicated in [5], LSTM model such as [13] cannot utilize recent observations effectively, since it spends too much resources on long-term memory.

The time series forecasting with multi-output and multi-step is an active research topic. [25] investigated this problem by employing multiple-output support vector regression (M-SVR) with multiple-input multiple-output (MIMO) prediction strategy. In [26], a cooperative neural evolution method for multi-task learning was adopted, which enables neural networks to be trained with shared knowledge representation.

Motivated by disadvantages of previous models above, we propose a LSTM model, where the hidden state transition is dependent on both input feature extracted by the autoencoder and the previous hidden state. The prediction output of the proposed model is decoded from both hidden state and latent representation of the input. So, our model can utilize long-term dependencies and recent observations at same time. And it can work on long sequences with variable lengths. We conduct performance comparison on both single-output and multi-output forecasting experiments.

We notice that there are some recent work [16] [17] which stack AE and RNN or LSTM together to predict time series. Different from our work, the AE and LSTM are trained separately in their models, and the decoder of AE is directly cut off, which may yield suboptimal weights for AE and LSTM cell. And they don't have results on multi-output prediction. The proposed method can outperform these models on chaotic and real-world datasets.

4 Augmented Long Short Term Memory Model

In this work, we incorporate autoencoder (AE) into LSTM model to improve time series forecasting performance. Here, $\boldsymbol{x}_t, \boldsymbol{z}_t$ and $\boldsymbol{h}_t$ denote the input observations, latent variables of the input and the hidden state of LSTM at time t respectively. The hidden state $\boldsymbol{h}_t \triangleq (\boldsymbol{c}_t, \boldsymbol{m}_t)$ includes both the cell state $\boldsymbol{c}_t$ and hidden state $\boldsymbol{m}_t$ of the memory cell as described in (2). In the proposed model, the LSTM is augmented by the latent variables learned from AE and corresponding feature extractors.

4.1 Encoder Model

Different from conventional autoencoder in [17], the latent variable extracted at time t is dependent on both the input observations and the previous hidden state of LSTM cell. It can be described as below:

$$\boldsymbol{z}_t = \varphi_\tau^{\text{enc}}(\varphi^{\boldsymbol{x}}(\boldsymbol{x}_t), \boldsymbol{h}_{t-1}) \tag{3}$$

where the decoder $\varphi_\tau^{\text{enc}}$ and feature extractor $\varphi^{\boldsymbol{x}}(\boldsymbol{x}_t)$ are both implemented as a two-layer neural network. Here, the input $\boldsymbol{x}_t$ is not directly processed by the encoder together with hidden state. The encoder only read the extracted feature of input $\boldsymbol{x}_t$, which can reduce some redundant

information in the input observations. It is especially important when processing time series with low sampling rate.

4.2 Decoder Model

This model applies AE in a recurrent setting. At each timestep, different from conventional AE, the output of the decoder, i.e. predicted value, is dependent on both the current latent variables of input and the previous hidden state of LSTM. This process can be depicted as:

$$\hat{\boldsymbol{y}}_t = \varphi_\tau^{\text{dec}}(\varphi_\tau^{\boldsymbol{z}}(\boldsymbol{z}_t), \boldsymbol{h}_{t-1}) \tag{4}$$

where the decoder $\varphi_\tau^{\text{dec}}$ and the feature extractor $\varphi_\tau^{\boldsymbol{z}}$ can be modeled by two-layer neural networks. In order to improve expressive capability, we also add feature extractor $\varphi_\tau^{\boldsymbol{z}}$ to latent variables, which can learn multi-scale and hierarchical features embedded in the time series. It can significantly improve the prediction performance on complex and highly dynamic time series.

Inside the LSTM cell, the hidden state is updated as follows:

$$\boldsymbol{h}_t = f_\theta(\varphi_\tau^{\boldsymbol{x}}(\boldsymbol{x}_t), \varphi_\tau^{\boldsymbol{z}}(\boldsymbol{z}_t), \boldsymbol{h}_{t-1}) \tag{5}$$

where transition function f is essentially the operation described in (2). We implement function f by multi-layer neural networks with weights θ. Different from previous LSTM work [13] [17] [18], the state transition incorporates the feature of input and feature of latent variables and previous state, rather than the original input and latent variables.

4.3 Objective Function

The target function is the prediction error regularized by the L2 norm of all trainable weights. We can describe this target as below:

$$\sum_{t=1}^{T} \|\hat{\boldsymbol{y}}_t - \boldsymbol{y}\| + \beta \|\boldsymbol{W}\|_2 \tag{6}$$

where $\hat{\boldsymbol{y}}_t$ and $\boldsymbol{y}$ are the predicted values and ground truth at time t, $\boldsymbol{W}$ denotes the weights of all neural netwokrs in the proposed model, and β is the trade-off coefficient for the regularizor. With all functions implemented by neural networks, the accumulative ELBO (6) is minimized by ADAM [20] algorithm. β is chosen to be specific to the experiment.

5 Simulation Results

This section presents a comprehensive experimental study that evaluates the performance of augmented LSTM method (A-LSTM) in terms of predictive ability. The performance is compared with cutting-edge time-series prediction models. For single-step ahead prediction task, we compare the proposed method with the deep belief network trained with restricted Boltzmann Machine (RBM) [9], stacked autoencoder model (SAE) [11], and autoencoder stacked on LSTM (Auto-LSTM) [17]. For multi-step ahead prediction task, the performance comparison is conducted with multi-output support vector regression (MSVR) [25] and co-evolutionary multi-task learning (CMTL) [26]. We conduct 5-step and 10-step ahead prediction experiments, and the scalability is also discussed. All these methods above have been introduced in Section 3.

The experiments are conducted using three simulated and two real-world time series datasets. The simulated time series are chaotic nonlinear systems such as Mackey and Glass [21], Lorenz

[22], and Rossler [23]. One real world dataset is individual household electric power consumption dataset [24]. All the respective time series data sets are scaled in the range $[0, 1]$ in order to be used for sigmoid units in the feedforward neural network. The performance is measured by the root-mean-squared error (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

where $y_i, \hat{y}_i$ are observed and predicted data.

5.1 Chaotic Processes Prediction

Across four simulations in this section, the length of training sequence is 1000, and the length of testing sequence is 500. The optimization is performed by ADAM with learning rate 0.0005.

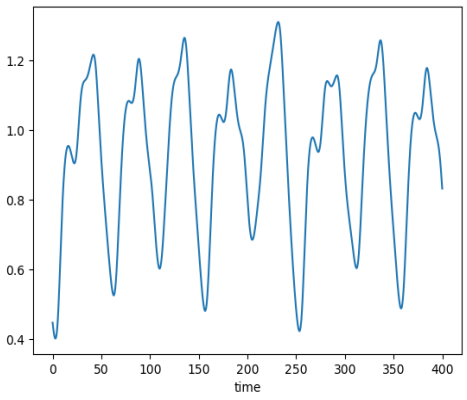
5.1.1 Mackey and Glass Time Series

We first experiment on Mackey-Glass time series [21], which is derived from differential equation as below:

$$\frac{dx}{dt} = \beta x(t) + \frac{\alpha x(t-\delta)}{1 + x(t-\delta)^{10}} \tag{7}$$

The parameters are chosen to be $\alpha = 0.2, \beta = -0.1$ and $\delta = 17$. The dataset is constructed by second-order Runge-Kutta method with the step size of 0.1, with plot as below.

Figure 1: Plot of Mackey and Glass Time Series.



For one-step ahead prediction, the input is a sequence of 5 samples with stride of 6. For 5-step ahead prediction, the input is a sequence of 15 samples with stride of 6. For 10-step ahead prediction, the input is a sequence of 10 samples with stride of 6. The RMSE and its variance of the proposed method (ALSTM), RBM, SAE and Auto-LSTM are shown in tables in the following.

	ALSTM	RBM	SAE	Auto-LSTM
RMSE	0.004778	0.010218	0.010233	0.008231

Table 1: Performance Comparison for One-step Ahead Prediction

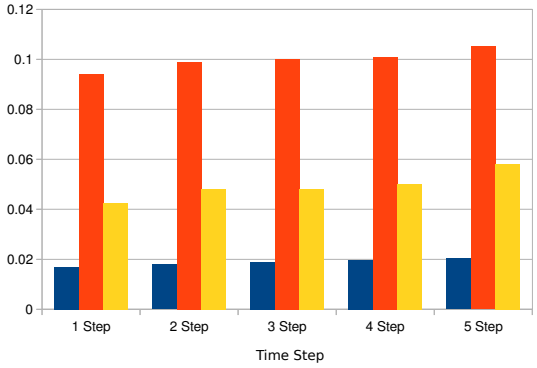


Figure 2: RMSE for 5-step Ahead Prediction

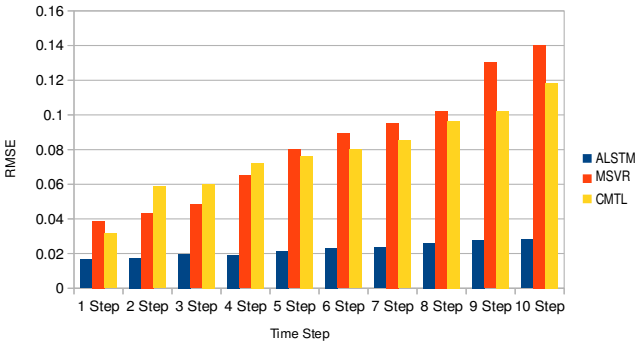


Figure 3: RMSE for 10-step Ahead Prediction

	ALSTM	MSVR	CMTL
1 Step	0.010839	0.022342	0.021562
2 Step	0.011292	0.026897	0.019023
3 Step	0.012740	0.035123	0.015021
4 Step	0.013451	0.045213	0.010228
5 Step	0.013305	0.054234	0.009342

Table 2: Variance for 5-step Ahead Prediction

The variance of ALSTM for 10-step ahead prediction is 0.011232, which is comparable with CMTL and MSVR The complete variance table is omitted.

5.1.2 Lorenz Time Series

The Lorenz attractor [22] is a nonlinear 3-D system that provides a simplified model of convection in the atmosphere. The Lorenz equations are given by:

$$\begin{aligned} \frac{dx}{dt} &= \sigma(y-x) \\ \frac{dy}{dt} &= \rho x - y - xz \\ \frac{dz}{dt} &= xy - \beta z \end{aligned}$$

In this experiment, the parameters are set to be $\sigma = 10, \rho = 28$ and $\beta = 8/3$. The 3-D attractor and x, y components are plotted as below.

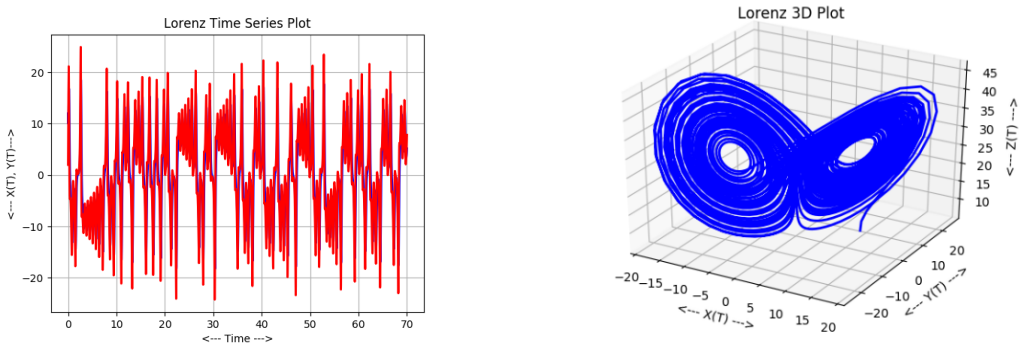


Figure 4: Plots of Lorenz Attractor

The performance of one-step ahead prediction is shown in the following table.

	ALSTM	RBM	SAE	Auto-LSTM
RMSE	0.006752	0.043278	0.020632	0.018992

Table 3: Performance Comparison for One-step Ahead Prediction

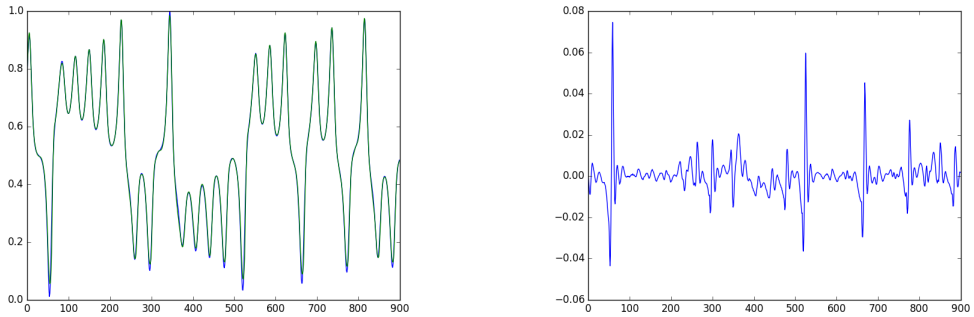


Figure 5: One-step prediction performance on test data. Left: plots of confidence interval and mean of generating distribution. Right: plot of error, which is the difference between the mean of generating distribution and ground truth.

For 5-step ahead, we use 6 samples with stride of 6. For 10-step ahead, we use 10 samples with stride of 4. The results for 5-step and 10-step ahead prediction are shown below.

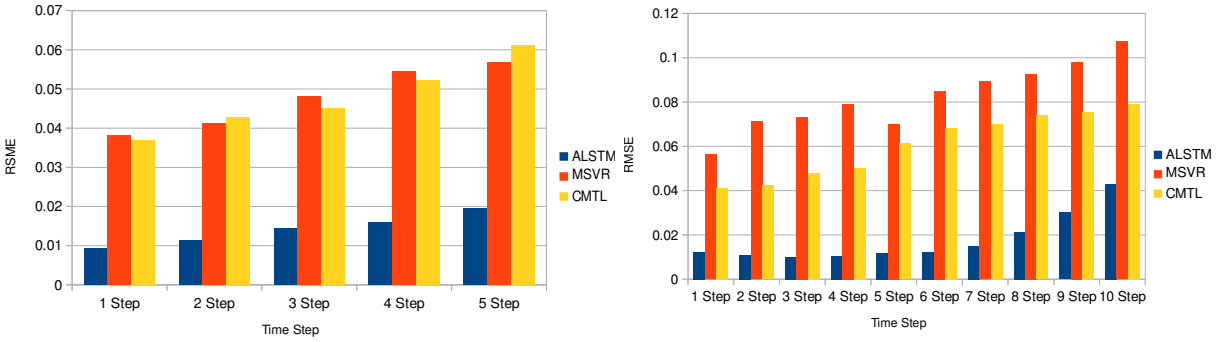


Figure 6: RMSE for 5-step and 10-step Ahead Prediction on Lorenz

	ALSTM	MSVR	CMTL
1 Step	0.011467	0.015583	0.014023
2 Step	0.010589	0.018234	0.012882
3 Step	0.012034	0.018199	0.010568
4 Step	0.012589	0.019202	0.010228
5 Step	0.014892	0.022021	0.011772

Table 4: Variance for 5-step Ahead Prediction

5.1.3 Rossler Time Series

The Rossler attractor [23] is generated by a system of three differential equations as below:

$$\begin{aligned} \frac{dx}{dt} &= -z - y \\ \frac{dy}{dt} &= x + ay \\ \frac{dz}{dt} &= b + z(x - c) \end{aligned}$$

where coefficients are set to be $a = 0.2, b = 0.2$ and $c = 4.6$. The figure below plots the 3-D Rossler Attractor and its x, y components.

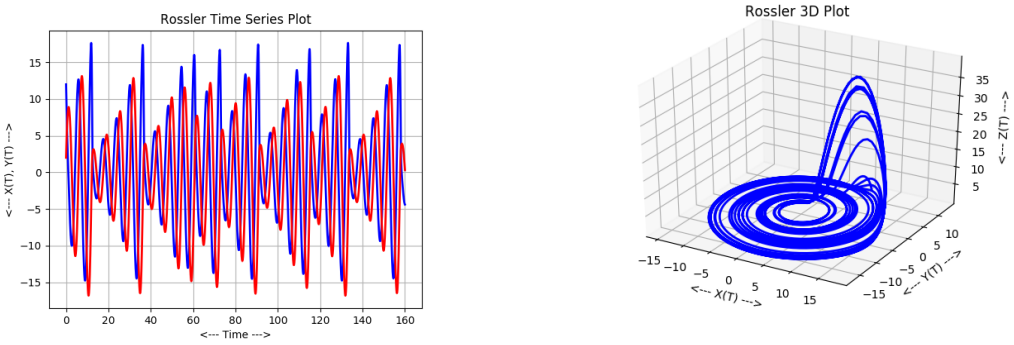


Figure 7: Plots of Rossler Attractor. Right: blue line represents x and red line shows y.

For one-step ahead prediction, we use 5 samples with stride of 6. The plots of prediction results are shown in the following figure.

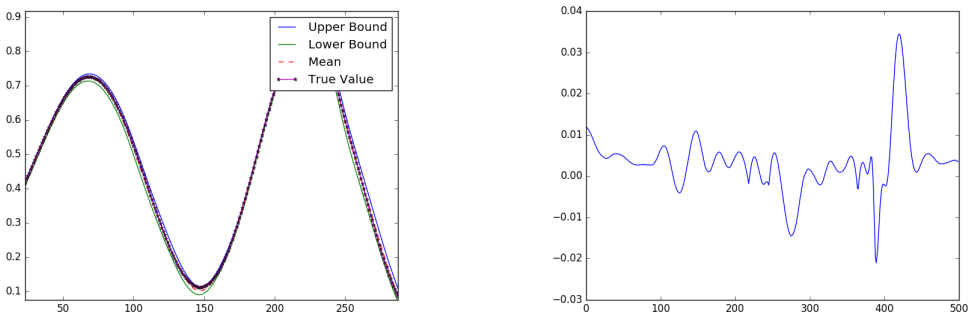


Figure 8: One-step prediction performance on test data. Left: plots of confidence interval, mean of generating distribution and ground truth. Right: prediction error.

The performance comparison for one-step ahead prediction is shown as below:

	ALSTM	RBM	SAE	Auto-LSTM
RMSE	0.006824	0.013278	0.010632	0.012992
Variance	0.010558	0.009832	0.011342	0.012421

The performance comparison for 5-step and 10-step ahead predictions are shown below.

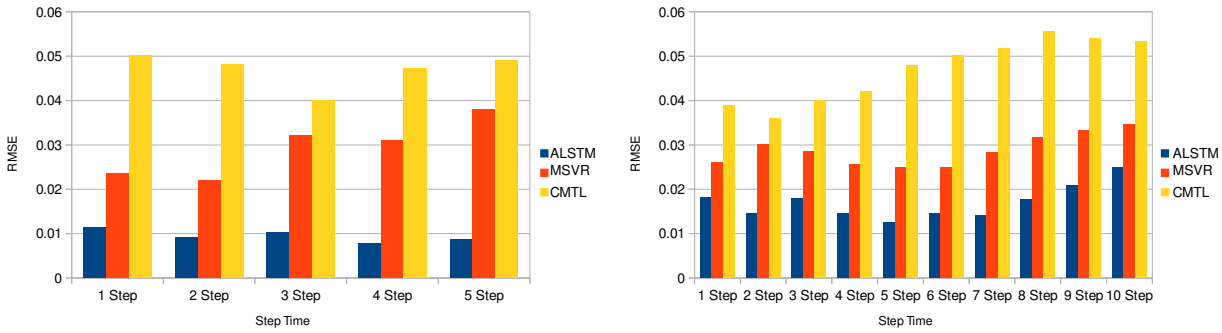


Figure 9: RMSE for 5-step and 10-step Ahead Prediction on Rossler

The variance of ALSTM is 0.103214, and the complete table for variance is omitted.

5.2 Power Consumption Data

This dataset contains measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years [24], which is equivalent to 7000 data points. The first 1000 data points are used for training and the remaining for testing.

The experiment shows that ALSTM model can also work well on real-world data. The following plots show the estimated value and its error, compared with ground truth.

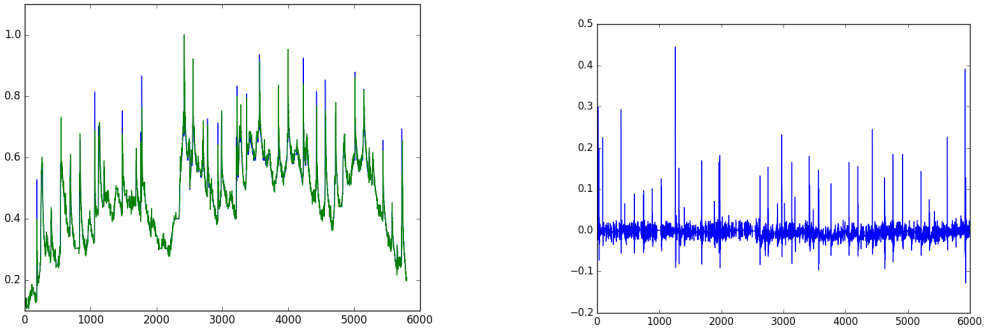


Figure 10: One-step prediction performance on test data. Left: plots of predicted data (green) and ground truth (blue). Right: plot of prediction error.

	ALSTM	RBM	SAE	Auto-LSTM
RMSE	0.011562	0.035586	0.030211	0.022520

Table 5: One-step Ahead Prediction Preformance Comparison

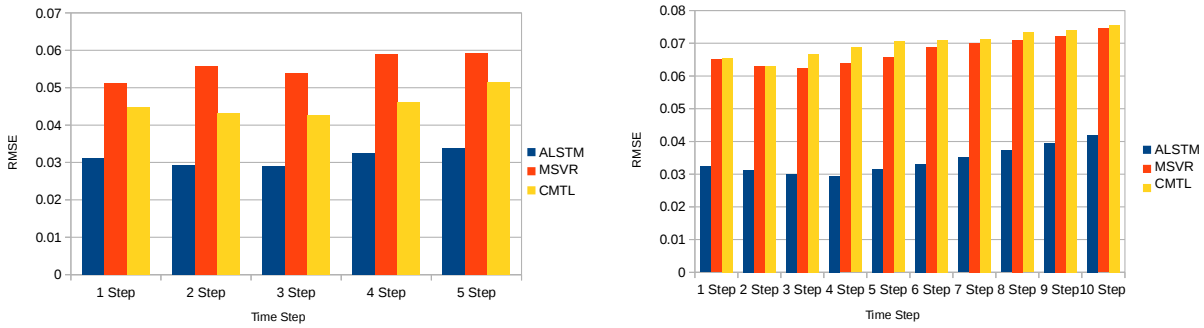


Figure 11: RMSE for 5-step and 10-step Ahead Prediction on Power Data

Comparing the performance of experiments with different number of output, we can see that performance of ALSTM model doesn't degrade in multi-output cases, which proves the scalability in practical applications.

6 Conclusion

This paper shows the advantage of ALSTM model on time series prediction. With comprehensive experiments, we prove that the ALSTM model can outperform cutting-edge algorithm on time series prediction and improve the state-of-art performance on both chaotic benchmarks and real-world datasets. Moreover, the predictions from ALSTM model have comparable variance with previous methods, resulting in small confidence intervals. Comparing performance of different number of prediction outputs, we can see the performance of 10-step prediction doesn't degrade too much, showing that the proposed method has strong ability of scaling to multi-output prediction. In the future, we are going to apply ALSTM to predict the financial data which has more variance and uncertainty.

References

[1] G. Box and G. Jenkins, Time Series Analysis: Forecasting and Control. San Francisco, CA: Holden-Day, 1970.

[2] F.-A. Gers, J. Schmidhuber, F. Cummins. Learning to Forget: Continual Prediction with LSTM. Technical Report IDSIA-01-99, 1999.

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. Neural Computation, 9(8):1735-1780, 1997.

[4] A. Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[5] F.A. Gers, D. Eck, and J. Schmidhuber, Applying LSTM to time series predictable through time-window approaches. Neural Nets WIRN Vietri-01. Springer, London, 2002. 193-200.

[6] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning . Nature, 521(7553):436-444, 2015.

[7] W. Huang, G. Song, H. Hong and K. Xie. Deep architecture for traffic flow prediction: deep belief networks with multi-task learning. IEEE Transactions on Intelligent Transportation Systems 15(5) 2191-2201, 2014.

[8] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: a deep learning approach. IEEE Transactions on Intelligent Transportation Systems,16(2):865-873, 2015.

[9] T. Kuremoto, S. Kimura, K. Kobayashi, and M. Obayashi. Time series forecasting using a deep belief network with restricted Boltzmann machines. Neurocomputing 137 (2014): 47-56.

[10] J.-T. Turner. Time series analysis using deep feed forward neural networks. Ph.D. thesis, University of Maryland, Baltimore County, 2014.

[11] P. Romeu, F. Zamora-Martnez, P. Botella-Rocamora, J. Pardo. Time-series forecasting of indoor temperature using pre-trained deep neural networks. In: Artificial Neural Networks and Machine LearningICANN 2013, pp. 451-458, Springer ,2013.

[12] P. Kanestrom. Traffic flow forecasting with deep learning. MS thesis. NTNU, 2017.

[13] X. Ma, et al. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transportation Research Part C: Emerging Technologies 54 : 187-197, 2015.

[14] G. Hinton, and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. Science,313.5786:504-507, 2006.

[15] Y. Cinar, H. Mirisaee, P. Goswami, etc. Time Series Forecasting using RNNs: an Extended Attention Mechanism to Model Periods and Handle Missing Values. arXiv preprint arXiv:1703.10089.

[16] H.-M. Nguyen, S. Woo, J. Im, T. Jun, and D. Kim. A Workload Prediction Approach Using Models Stacking Based on Recurrent Neural Network and Autoencoder. High Performance Computing and Communications; IEEE 14th International Conference on Smart City, 2016.

[17] A. Gensler, J. Henze, B. Sick, and N. Raabe. Deep Learning for solar power forecastingAn approach using AutoEncoder and LSTM Neural Networks. In Systems, Man, and Cybernetics (SMC), 2016 IEEE International Conference on (pp. 002858-002865), Oct. 2016.

[18] F.-M. Bianchi, et al. An overview and comparative analysis of Recurrent Neural Networks for Short Term Load Forecasting. arXiv preprint arXiv:1705.04378, 2017.

[19] Bengio, Y., 2007. Learning deep architectures for AI. Technical Report 1312. Dept. IRO, Universite de Montreal.

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[21] M. Mackey and L. Glass. Oscillation and chaos in physiological control systems, Science 197 (4300), pages 287-289, 1977.

[22] E. Lorenz. Deterministic non-periodic flows, J. Atmos. Sci., vol. 20, pages 267-285, 1963.

[23] O. E. Rossler. An equation for continuous chaos. Physics Letters A, 57(5), 397-398, 1976.

[24] UCI Machine Learning Repository. Available at http://archive.ics.uci.edu/ml/datasets

[25] Y. Bao, T. Xiong, and Z. Hu. Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing 129 (2014): 482-493.

[26] R. Chandra, Y.-S. Ong, and C.-K. Goh. Co-evolutionary multi-task learning for dynamic time series prediction." arXiv preprint arXiv:1703.01887, 2017.