Evaluation of deep learning models for multi-step ahead time series prediction

Rohitash Chandra[a,1], Shaurya Goyal[b], Rishabh Gupta[c]

[a]*School of Mathematics and Statistics, UNSW Sydney, NSW 2006, Australia*
[b]*Department of Mathematics, Indian Institute of Technology, Delhi, 110016 , India*
[c]*Department of Geology and Geophysics, Indian Institute of Technology,Kharagpur, 721302 , India*

Abstract

Time series prediction with neural networks have been focus of much research in the past few decades. Given the recent deep learning revolution, there has been much attention in using deep learning models for time series prediction, and hence it is important to evaluate their strengths and weaknesses. In this paper, we present an evaluation study that compares the performance of deep learning models for multi-step ahead time series prediction. Our deep learning methods compromise of simple recurrent neural networks, long short term memory (LSTM) networks, bidirectional LSTM, encoder-decoder LSTM networks, and convolutional neural networks. We also provide comparison with simple neural networks use stochastic gradient descent and adaptive gradient method (Adam) for training. We focus on univariate and multi-step-ahead prediction from benchmark time series datasets and compare with results from from the literature. The results show that bidirectional and encoder-decoder LSTM provide the best performance in accuracy for the given time series problems with different properties.

Keywords: Recurrent neural networks, LSTM, Deep Learning,, Time Series Prediction

1. Introduction

Apart from econometric models, machine learning methods became extremely popular for time series prediction or forecasting in the last few decades [1, 2, 3, 4, 5, 6, 7]. Some of the popular categories include one-step, multi-step, and multivariate prediction. Recently some attention has been given to dynamic time series prediction where the size of the input to the model can dynamically change [8]. Just as the term indicates, one-step prediction refers to the use of a model to make a prediction one-step ahead in time whereas a multi-step prediction refers to a series of steps ahead in time from an observed trend in a time series [9, 10]. In the latter case, the *prediction horizon* defines the extent of future prediction. The challenge is to develop models that produce low prediction errors as the prediction horizon increases given the chaotic nature and noise in the dataset [11, 12, 13]. There are two major strategies for multi-step-ahead prediction which include *recursive* and *direct* strategies. The recursive strategy features the prediction from a one-step-ahead prediction model as the input for future prediction horizon [14, 15], where error in the prediction for the next horizon is accumulated in future horizons. The direct strategy encodes the multi-step-ahead problem as a multi-output problem [16, 17] which in the case of neural networks can be represented by multiple neurons in the output later, where each neuron denotes the prediction horizon. The major challenges in multi-step-ahead prediction include highly chaotic time series and those that have missing data which has been approached with non-linear filters and neural networks [18].

Neural networks have been popular for time series prediction for various applications [19]. Different neural network architectures have different strengths and weaknesses and given that time series prediction is concerned with careful integration of knowledge in temporal sequences according to different dimensions, it is important to choose the right neural network architecture and training algorithm.

Recurrent neural networks (RNNs) are known to be better suited for modelling temporal sequences [20, 21, 22, 23] and are also more suitable for modeling dynamical systems when compared to feedforward networks [24, 25, 26]. RNNs have shown to be robust methods for time series prediction [27]. The Elman RNN [20, 28] is one of the earliest architecture trained by backpropagation through-time which is an extension of the backpropagation algorithm for feedforward networks [21]. The limitations in learning by RNNs for long-term dependencies in sequences that span hundreds or thousands of time-steps [29, 30] were addressed by *long short-term memory* networks (LSTMs) [22].

More recently, with the deep learning revolution [23], there has been further improvements such as *gated recurrent unit* (GRU) [31, 32] networks, which provides similar performance than LSTMs, but are simpler to implement. Some of the other extensions include predictive state RNNs [33] that combined RNNs with power of predictive state representations [34]. Bidirectional RNNs connect two hidden layers of opposite directions to the same output where the output layer can get information from past and future states simultaneously [35]. The idea was further extended into bidirectional-LSTMs for phoneme classification [36] which performed better than standard RNNs and LSTMs. More work has been made by combining bidi-

Email address: rohitash.chandra@unsw.edu.au (Rohitash Chandra)

rectional LSTMS with convolutional neural networks (CNNs) for natural language processing with problem of named entity recognition [37]. Further extensions have been made by encoder-decoder LSTM that used a LSTM to map the input sequence to a vector of a fixed dimensionality and used another LSTM to decode the target sequence from the vector for English to French translation task [38].

On the other hand, backpropgation neural network which is also slowly known as shallow learning as opposed to deep learning has a number of improvements for training and generalisation. Deep learning methods such as convolutional neural networks, presented idea of regularisation using dropouts during training that has been helpful for generalisation [39]. Adaptive gradient methods such as the Adam optimiser has become popular for training shallow networks [40]. Apart from these, other ways of training feedforward networks such as evolutionary algorithms have been used for time series problems [41, 8]. Furthermore, RNNs have also been training by evolutionary algorithms with applications for time series prediction [42, 43].

Noting these advances, it is important to evaluate the performance for a challenging problem which in our case is multi-step time series prediction. We note that limited work has been done comparison done between FNN and RNNs for time series [44, 45]. We note that while most of the LSTM applications have been natural language processing and signal processing applications such as phoneme recognition, there is no work that evaluates their performance for time series prediction, particularly multi-step ahead prediction. Since the underlying feature of LSTMs is in handing temporal sequences, it is worthwhile to investigate about their predictive power, i.e. as the prediction horizon increases. The recent advances in technologies such as *Tensorflow* have improved computational efficiency of RNNs [46] and enabled easier implementation for providing comprehensive evaluation of their performance.

In this paper, we present an evaluation study that compares the performance of selected deep learning models for multi-step ahead time series prediction. Our deep learning methods compromise of standard LSTMs, bidirectional LSTMS, encoder-decoder LSTMs, and CNNs. Our shallow learning models use stochastic gradient descent and prominent adaptive gradient method known as Adam optimiser for training. We examine univariate time series prediction with selected models and learning algorithms for benchmark time series datasets. We also compare our results with other related machine learning methods for multi-step time series problems from the literature.

The rest of the paper is organised as follows. Section 2 presents a background and literature review of related work. Section 3 presents the details of the different deep learning models, and Section 4 presents experiments and results. Section 5 provides a discussion and Section 6 concludes the paper with discussion of future work.

2. Related Work

2.1. Multi-step time series prediction

The recursive strategy of multi-step-ahead prediction features the predicted value of the current prediction horizon as inputs to next prediction horizon which iterates, hence it is also known as iterated strategy. One of the first attempts for recursive strategy was using state-space Kalman filter and smoothing [47] followed by recurrent neural networks [48]. Later, a dynamic recurrent network used current and delayed observations as inputs to the network which reported excellent generalization performance [49]. Then the non-parametric Gaussian process model was used to incorporate the uncertainty about intermediate regressor values [50]. The *Dempster–Shafer* regression technique for prognosis of data-driven machinery used iterative strategy with promising performance [51]. Lately, reinforced real-time recurrent learning was used with iterative strategy for flood forecasts [12].

The direct strategies for multi-step-ahead prediction feature all the prediction horizons during training typically as multiple outputs. Initial progress was made using recurrent neural networks trained by backpropagation through-time algorithm [13]. A review of single-output vs. multiple-output approaches showed direct strategy more promising choice over recursive strategy [17]. A multiple-output support vector regression (M-SVR) achieved better forecasts when compared to standard SVR using direct and iterated strategies [52].

The third strategy features the combination of recursive and direct strategies. Initial work featured multiple SVR models that were trained independently based on the same training data and with different targets [14]. An optimally pruned extreme learning machine (OP-ELM) was used using recursive, direct and a combination of the two strategies in an ensemble approach where the combination gave better performance than standalone methods [53]. Chandra et. al [54] presented a recursive neural network inspired by multi-task learning and cascaded neural networks for multi-step ahead prediction, training by evolutionary algorithm which gave very promising performance when compared to the literature . Ye and Dai [55] presented a multitask learning method which considers different prediction horizons as different tasks and explores the relatedness among horizons while forecasting them in parallel. The method consistently achieved lower error values over all horizons when compared to other related iterative and direct prediction methods.

A comprehensive study on the different strategies was given using a large experimental benchmark (NN5 forecasting competition) [56], and further comparison for macroeconomic time series where it was reported that the iterated forecasts typically outperformed the direct forecasts [57] and relative performance of the iterated forecasts improved with the forecast horizon, with further comparison that presented an encompassing representation for derivation auto-regressive coefficients [58]. A study on the properties shows that direct strategy provides prediction values that are relatively robust to breaks and the benefits increases with the prediction horizon [59].

Some applications of the different machine learning methods that apply multi-step-ahead prediction for real-world problems. These include 1.) auto-regressive models for predicting critical levels of abnormality in physiological signals [60], 2.) flood forecasting using recurrent neural networks [61, 62], 3.) emissions of nitrogen oxides using a neural network and related ap-

proaches [63], 4.) photo-voltaic power forecasting using hybrid support vector machine [64], 5.) Earthquake ground motions and seismic response prediction [65], and 6. central-processing unit (CPU) load prediction [66]. More recently, Wu [67] employed an adaptive-network-based fuzzy inference system with uncertainty quantification the prediction of short-term wind and wave conditions for marine operations. Wang and Li [68] presented multi-step ahead prediction for wind speed which was based on optimal feature extraction, deep learning with LSTMs, and error correction strategy. The method showed lower error values for one, three and five-step ahead predictions in comparison to related methods. Wang and Li [69] also presented another hybrid approach to the multi-step ahead wind speed prediction with empirical wavelet transformation for feature extraction, autoregressive fractionally integrated moving average to detect long memory characteristics and swarm-based backpropagation neural network.

2.2. Deep learning: LSTM and applications for time series

Deep learning naturally features robust spacial temporal information processing [70, 23] and has been popular for modelling temporal sequences. Deep learning has became very successful for computer vision [71], reinforcement learning for games [72], and big data related problems. Deep learning typically refer to recurrent networks (RNNs), convolutional neural networks (CNNs) [73, 72], deep belief networks, and LSTM networks which are a special class of RNNs addressing long-term dependency problem in time series datasets [23]. RNNs have been popular for forecasting time series with their ability to capture temporal information [27, 74, 10, 75, 76]. In terms of uncertainity quantification in predictions, Mirikitani and Nikolaev used [77] variational inference for Bayesian RNNs for time series forecasting.

CNNs have gained attention recently in forecasting time series. Wang et. al [78] used CNNs with wavelet transform probabilistic wind power forecasting. Xingjian et. al [79] used CNNs in conjunction with LSTMs to capture spatial-temporal sequences for forecasting precipitation. Amarasinghe et al. [80] employed CNNs for energy load forecasting, and Huang and Kuo [81] combined CNNs and LSTMs for air pollution quality forecasting. Sudriani et al. [82] employed LSTMs for forecasting discharge level of a river for managing water resources. Ding et al. [83] employed CNNs to evaluate different events on stock price behavior, and Nelson et al. [84] used LSTMs to forecast stock market trends. Chimmula and Zhand employed LSTMs for forecasting COVID-19 transmission in Canada [85]. The authors predicted the possible ending point of the outbreak around June 2020, Canada reached the daily new cases peak by 2nd May[1] which further reduced.

[1] https://www.worldometers.info/coronavirus/country/canada/

3. Methodology

3.1. Data reconstruction

The original time series data is reconstructed for multi-step-ahead prediction. Taken's theorem expresses that the reconstruction can reproduce important features of the original time series [86]. Therefore, given an observed time series $x(t)$, an embedded phase space $Y(t) = [(x(t), x(t-T), ..., x(t-(D-1)T)]$ can be generated; where T is the time delay, D is the embedding dimension (window size) given $t = 0, 1, 2, ..., N - DT - 1$, and N is the length of the original time series. A study needs to be done to determine good values for D and T in order to efficiently apply Taken's theorem [87]. We note that Taken's proved that if the original attractor is of dimension d, then $D = 2d + 1$ would be sufficient [86].

3.2. Shallow learning via simple neural networks

We refer to the the backpropagation neural network and multiplayer perception as simple neural network which has been typically trained by the stochastic gradient descent (SGD) algorithm. SGD maintains a single learning rate for all weight updates which does not change during the training. Adaptive moment estimation (Adam) learning algorithm [88] extends the stochastic gradient descent by maintaining and adapting the learning rate for each network weight as learning unfolds. Using first and second moments of the gradients, Adam computes individual adaptive learning rates which is inspired by the *adaptive gradient algorithm* (AdaGrad) [89]. In the literature, Adam has shown better results when compared to stochastic gradient descent and AdaGrad, and our experiments will consider evaluating it further for multi-step time series prediction. Adam's learning procedure for iteration t is formulated as

$$\begin{aligned}
\Theta_{t-1} &= [\mathbf{W_{t-1}}, \mathbf{b_{t-1}}] \\
g_t &= \nabla_\Theta J_t(\Theta_{t-1}) \\
m_t &= \beta_1.m_{t-1} + (1-\beta_1).g_t \\
v_t &= \beta_2.v_{t-1} + (1-\beta_2).g_t^2 \\
\hat{m}_t &= m_t/(1-\beta_1^t) \\
\hat{v}_t &= v_t/(1-\beta_2^t) \\
\Theta_t &= \Theta_{t-1} - \alpha.\hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)
\end{aligned} \tag{1}$$

where m_t, v_t are the respective first and second moment vectors for iteration t; β_1, β_2 are constants $\in [0, 1]$, α is the learning rate, and ϵ is a close to zero constant.

3.3. Simple recurrent neural networks

The Elman RNN [28] is a prominent example of *simple recurrent networks* that feature a context layer to act as memory and incorporate current state for propagating information into future states, given future inputs. The use context layer to store the output of the state neurons from computation of the previous time steps makes them applicable for time-varying patterns in data. The context layer maintains memory of the prior hidden layer result as shown in Figure **??**.

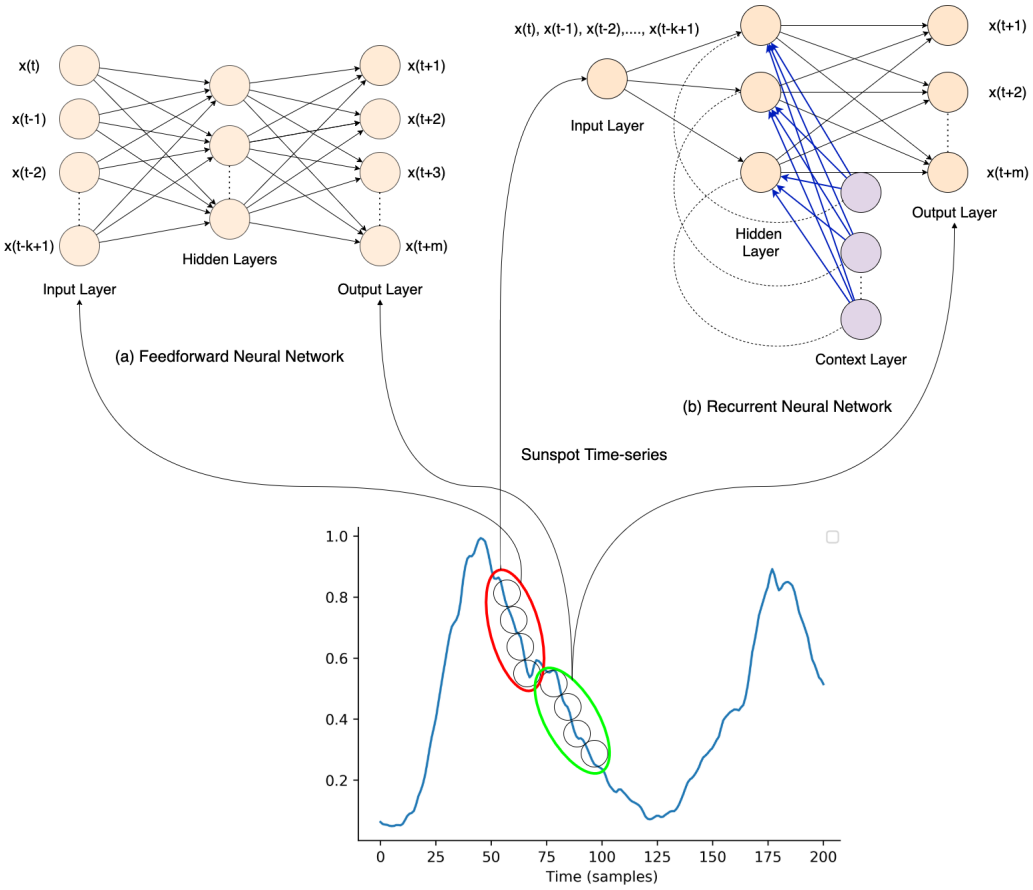


Figure 1: Feed Forward Neural Network and Elman RNN for time series prediction

A vectorised formulation can be given as follows

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{2}$$

where; x_t input vector, h_t hidden layer vector, y_t output vector, W represent the weights for hidden and output layer, U is the context state weights, b is the bias, and σ_h and σ_y are the respective activation functions. Backpropagation through time (BPTT) [21] which is an extension of the backpropagation algorithm is a prominent method for training simple recurrent networks which features gradient descent with the major difference that the error is backpropagated for a deeper network architecture that features states defined by time.

3.4. LSTM neural networks

Simple recurrent networks have the [22] limitation of learning long-term dependencies with problems in vanishing and exploding gradients [90] in simple recurrent neural networks. LSTMs employ using memory cells and gates for much better capabilities in remembering the long-term dependencies in temporal sequences as shown in Figure 2

LSTM units are trained in a supervised fashion on a set of training sequences using an adaptation of the BPTT algorithm that considers the respective gates [22]. Neuroevolution provides an alternate training method that does not request gradients [91]. Policy gradient methods in reinforcement learning framework can be used in case where no training labels are present [92].

LSTM networks calculate a hidden state h_t as

$$\begin{aligned} i_t &= \sigma(x_t U^i + h_{t-1} W^i) \\ f_t &= \sigma(x_t U^f + h_{t-1} W^f) \\ o_t &= \sigma(x_t U^o + h_{t-1} W^o) \\ \tilde{C}_t &= \tanh\left(x_t U^g + h_{t-1} W^g\right) \\ C_t &= \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \\ h_t &= \tanh(C_t) * o_t \end{aligned} \tag{3}$$

where, i_t, f_t and o_t refer to the input, forget and output gates, at time t, respectively. x_t and h_t refer to the number of input features and number of hidden units, respectively. W and U is the weight matrices adjusted during learning along with b which is the bias. The initial values are $c_0 = 0$ and $h_0 = 0$. All the gates have the same dimensions d_h, the size of your hidden state. $\tilde{C}_t$ is a "candidate" hidden state, and C_t is the internal memory of the unit as shown in Figure 2. Note that we denote (*) as element-wise multiplication.

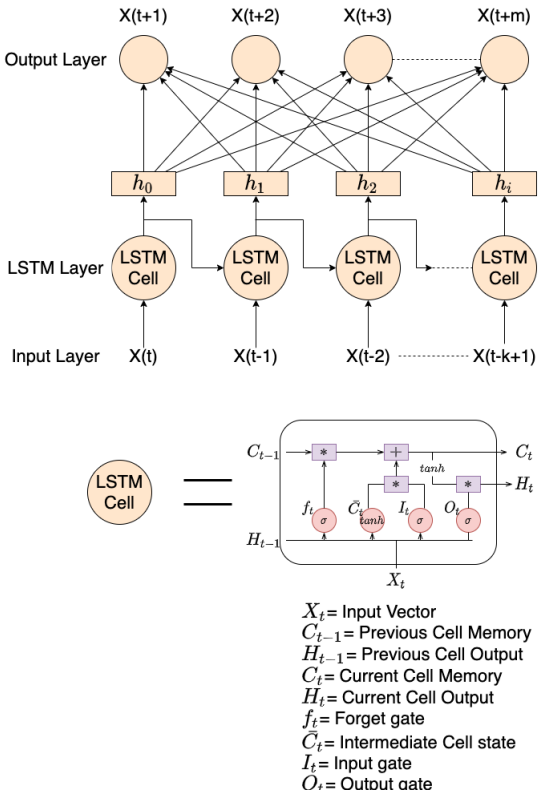


Figure 2: Long Short-Term Memory (LSTM) neural networks

3.5. Bi-directional LSTM networks

A major shortcoming of conventional RNNs is that they only make use of previous context state for determining future states. Bidirectional RNNs (BD-RNNs) [93] process information in both directions with two separate hidden layers, which are then propagated forward to the same output layer. BD-RNNs hence consist of placing two independent RNNs together to allow the networks to have both backward and forward information about the sequence at every time step. BD-RNN computes the forward hidden sequence h_f, the backward hidden sequence h_b, and the output sequence y by iterating the backward layer from $t = T$ to $t = 1$, the forward layer from $t = 1$ to $t = T$ and then updating the output layer.

Originally proposed for world-embedding in natural language processing, bi-directional LSTM networks (BD-LSTM) [94], can access longer-range context or state in both directions similar to BD-RNNs. BD-LSTM would intake inputs in two ways, one from past to future and one from future to past which different from conventional LSTM since by running information backwards, state information from the future is preserved. Hence, with two hidden states combined, in any point in time the network can preserve information from both past and future as shown in Figure 3. BD-LSTM networks have been used in several real-world sequence processing problems such as phoneme classification [94], continuous speech recognition [95] and speech synthesis [96].

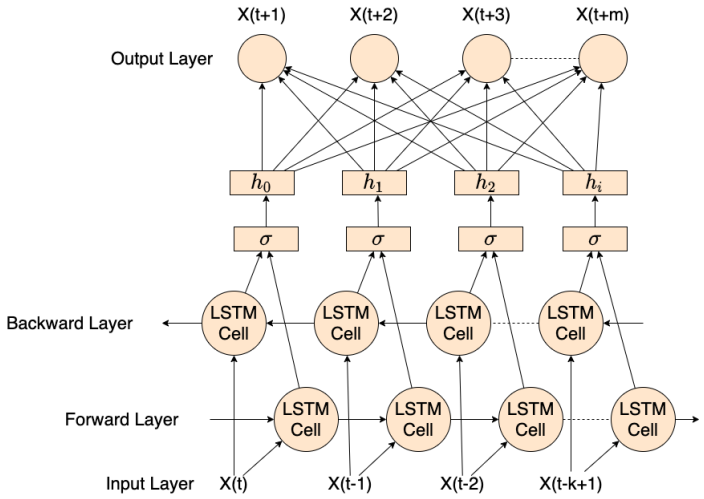


Figure 3: Bi-directional LSTM

3.6. Encoder-Decoder LSTM networks

Sutskever et. al [97] introduced the encoder-decoder LSTM network (ED-LSTM) which is a sequence to sequence model for mapping a fixed-length input to a fixed-length output where the length of the input and output may differ, which is applicable in automatic language translation tasks (English to French for example). In the case of multi-step series prediction and multivariate analysis, both the input and outputs are of variable lengths. Hence, the input is the sequence of video frames $(x_1, ..., x_n)$, and the output is the sequence of words $(y_1, ..., y_m)$. Therefore, we estimate the conditional probability of an output sequence $(y_1, ..., y_m)$ given an input sequence $(x_1, ..., x_n)$ i.e. $p(y_1, ..., y_m|x_1, ..., x_n)$.

ED-LSTM networks handle variable-length input and outputs by first encoding the input sequences, one at a time, using a latent vector representation, and then decoding from that representation. In the encoding phase, given an input sequence, the ED-LSTM computes a sequence of hidden states In decoding phase, it defines a distribution over the output sequence given the input sequence as shown in Figure 4.

3.7. CNNs

CNNs introduced by introduced by LeCun [98, 99] are prominent deep learning architecture inspired by the natural visual system of mammals. CNNs could classify handwritten digits and could be trained using backpropagation algorithm [100] which later has been prominent in many computer vision and image processing tasks. More recently, CNNs have been applied for time series prediction [80, 79, 78] with promising results.

CNNs learn spatial hierarchies of features by using multiple building blocks, such as convolution, pooling layers, and fully connected layers. Figure 5 shows an example of a CNN used for time series prediction, given a univariate time series input and multiple output neurons representing different prediction horizons. We note that multivariate time series is more appropriate

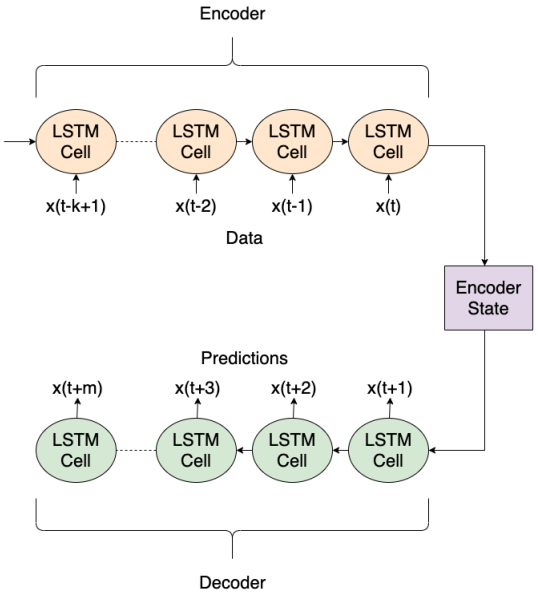


Figure 4: Encoder-Decoder LSTM

to take advantage of feature extraction via the convolutional and the pooling layers.

4. Experiments and Results

We present experiments and results that consider simple neural networks featuring SGD and Adam learning, and and deep learning methods that feature RNNs, LSTM networks, ED-LSTM, BD-LSTM and CNNs.

4.1. Experimental Design

The selected benchmark problems are are combination of simulated and real-world time series. The simulated time series are Mackey-Glass[101], Lorenz [102], Henon [103], and Rossler [104]. The real-world time series are Sunspot [105], Lazer [106] and ACI-financial time series [107]. They have been used in our previous works and are prominent benchmarks for time series problems [108, 109, 110]. The Sunspot time series indicates solar activities from November 1834 to June 2001 and consists of 2000 data points [105]. The ACI-finance time series contains closing stock prices from December 2006 to February 2010, featuring 800 data points [107]. The Lazer time series is from the *Santa Fe competition* that consists of 500 points [106].

The respective time series were pre-processed into a state-space vector [86] with embedding dimension $D = 5$ and time-lag $T = 1$ for 10-step-ahead prediction. We considered respective neural network models with number of hidden neurons, selected learning rate and other papers and in trial experiments to determine appropriate models. Table 1 gives details for the topology of the respective models in terms of input, hidden and output layers. In the respective datasets, we used first 1000 data points from which the first 60% was used for training and remaining for testing. All the respective time series are scaled in the range [0,1].

We use the root-mean-squared error (RMSE) in Equation 4 as the main performance measures for different prediction horizons.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4}$$

where $y_i, \hat{y}_i$ are the observed data, predicted data, respectively. N is the length of the observed data. We apply RMSE in Equation 4 for each prediction horizon, and also report the mean error for all the respective prediction horizons.

4.2. Results

We report the mean and 95 % confidence interval of RMSE for each prediction horizon for the respective problem for train and test datasets from 30 experimental runs with different initial neural network weights.

The results are shown in Figure 9 to Figure 12 for the simulated time series problems (Table 8 to 11 in Appendix). Figure 6 to 8 show results for the real-world time series problem (Table 5 to 7 in Appendix). We define robustness as the confidence interval which must be as low as possible to indicate high confidence in prediction. We consider scalability as the ability to provide consistent performance to some degree of error given the prediction horizon increases.

Note that the results are given in terms of the RMSE where the lower values indicate better performance. Note that each problem reports 10-step-ahead prediction results for 30 experiments with RMSE mean and 95% confidence interval as histogram and error bars, shown in Figures 6 to 12.

We first consider results for real world time series that naturally feature noise (ACI-Finance, Sunspot, Lazer). Figure 6 shows the results for the ACI-fiancee problem. We observe that the test performance is better than the train performance in Figure 6 (a) where deep learning models provide more reliable performance. The prediction error increases with the prediction horizon, and the deep learning methods do much better than simple learning methods (FNN-SGD and FNN-Adam). We find that LSTM provides the best overall performance as shown in Figure 6 (b). The overall test performance shown in Figure 6 (a) indicates that FNN-Adam and LSTM provide similar performance, which are better than rest of the problems. Figure 13 shows ACI-finance prediction performance of the best experiment run with selected prediction horizons that indicate how the prediction deteriorates as prediction horizon increases.

Next, we consider the results for the Sunspot time series shown in Figure 7 which follows a similar trend as the ACI-finance problem in terms of the increase in prediction error along with the prediction horizon. Also, the test performance is better than the train performance as evident from Figure 7 (a). The LSTM methods (LSTM, ED-LSTM, BD-LSTM) gives better performance than the other methods as can be observed from Figure 7 (a) and 7 (b). Note that the FNN-SGD gives the worst performance and the performance of RNN is better than

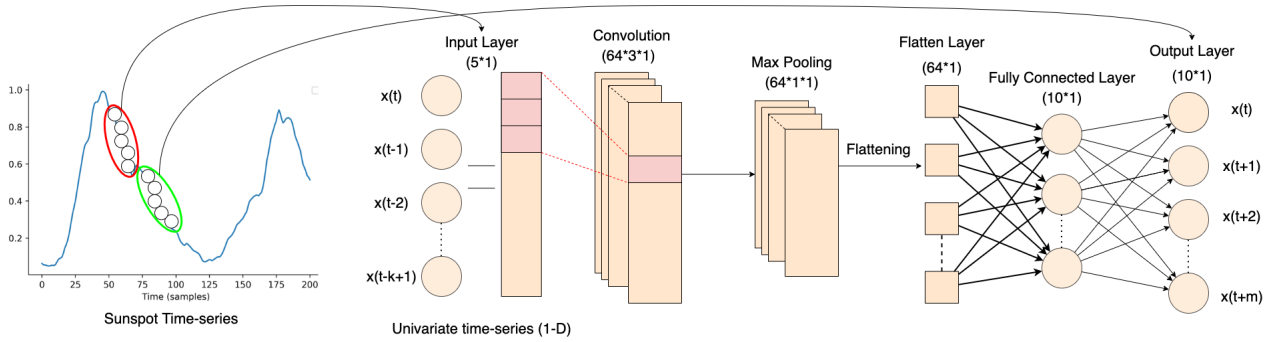


Figure 5: One-dimensional Convolutional Neural Network for time series

	Input	Hidden Layers	Output	Comments
FNN-Adam	5	1	10	Hidden Layer size= 10, Optimizer= adam, Activation= relu, Epochs=1000
FNN-SGD	5	1	10	Hidden Layer size= 10, Optimizer= SGD, Activation= relu, Epochs=1000
LSTM	5	1	10	Hidden Layer of 10 cells, Optimizer= adam, Activation= relu, Epochs=1000
BD-LSTM	5	1	10	Forward and Backward layer of 10 Cells each, Optimizer= adam, Activation= relu, Epochs=1000
ED-LSTM	5	4	10	Two LSTM, Repeat Vector and a Time distributed layer, Optimizer= adam, Activation= relu, Epochs=1000
RNN	5	2	10	Hidden Layers consist of 10 Cells and 10 neurons resp., Optimizer= SGD, Epochs=1000
CNN	5	4	10	Convolution, Pooling, Flatten and Dense Layer, Optimizer= adam, Activation= relu, Epochs=1000, Filters=64, Convolutional Window size=3, Max-Pooling Window size=2

Table 1: Configuration of models

that of CNN, FNN-SGD, FNN-Adam but poorer than LSTM methods. Figure 14 shows Sunspot prediction performance of the best experiment run with selected prediction horizons.

The results for Lazer time series is shown in Figure 8, which exhibits a similar trend in terms of the train and test performance as the other real-world time series problems. Note that the Lazer problem is highly chaotic (as visually evident in Figure 16), which seems to be the primary reason behind the difference in performance for the prediction horizon in contrast to other problems as displayed in Figure 8 (b). It is striking that none of the methods appear to be showing any trend for the prediction accuracy along the prediction horizon, as seen in previous problems. In terms of scalability, all the methods appear to be performing better in comparison with the other problems. The performance of CNN is better than that of RNN, which is different from other real-world time series. Figure 16 shows Lazer prediction performance of the best experiment run using ED-LSTM with selected prediction horizons. We note that due to the chaotic nature of the time series, the prediction performance is visually not clear.

We now consider simulated time series that do not feature noise (Henon, Mackey-Glass, Rosssler, Lorenz). The Henon time series in Figure 9 shows that ED-LSTM provides the best performance. Note that there is a more significant difference between the three LSTM methods when compared to other problems. The trends are similar to the ACI-finance and the Sunspot problem given the prediction horizon performance in Figure 9 (a) and 9 (b) where the simple learning methods (FNN-SGD and FNN-Adam) appear to be more scalable than the other methods along the prediction horizon although they perform poorly. Figure 17 and Figure 15 shows Mackey-Glass and Henon prediction performance of the best experiment run using ED-LSTM with selected prediction horizons. The Henon prediction in Figure 15 indicates that it is far more chaotic than Mackey-Glass and hence it faces more challenges. We show them since these are cases with no noise when compared to real-world time series previously shown that has a larger deviation or deterioration in prediction performance as the prediction horizon increases (Figures 13 and Figure 14).

For the Lorenz, Mackey-Glass and Rossler simulated time series, the deep learning methods are performing far better than the simple learning methods as can be seen in Figures 10, 11 and 12. The trend along the prediction horizon is similar to previous problems, i.e., the prediction error increases along with the prediction horizon. If we consider scalability, the deep learning methods are more scalable in the Lorenz, Mackey-Glass and Rossler problems than the previous problems. This is the first instance where the CNN has outperformed LSTM for Mackey-Glass and Rossler time series.

We note that there have been distinct trends in prediction for the different types of problems. In the simulated time series problems, if we exclude Henon, we find similar trend for Mackey-Glass, Lorenz and Rossler time series. The trend indicates that simple neural networks face major difficulties and

ED-LSTM and BD-LSTM networks provides the best performance, which also applies to Henon time series, except that it has close performance for simple neural networks when compared to deep learning models for 7-10 prediction horizons (Figure 9 b). This difference reflects in the nature of the time series which is highly chaotic in nature (Figure 15). We further note that the simple neural networks, in the Henon case (Figure 9) does not deteriorate in performance as the prediction horizon increases when compared to Mackley-Glass, Lorenz and Rossler problems, although they give poor performance.

The performance of simple neural networks in Lazer problem shows a similar trend in Lazer time series, where the predictions are poor from the beginning and its striking that LSTM networks actually improve the performance as the prediction horizon increases (Figure 8 b). This trend is a clear outline when compared to rest of real-world and simulated problems, as all of them have results where the deep learning models deteriorate as the prediction horizon increase.

4.3. Comparison with the literature

Table 3 and 4 show a comparison with related methods from the literature for simulated and real-world time series, respectively. We note that the comparison is not truly fair as other methods may have employed different models with different data processing and also in reporting of results with different measures of error as some papers report best experimental run and do not show mean and standard deviation. We highlight in bold the best performance for respective prediction horizon. Table 3, we compare the Mackey-Glass and Lorenz time series performance for two-step-ahead prediction by real-time recurrent learning (RTRL) and echo state networks (ESN) [12]. Note that the * in the results implies that the comparison was not very fair due to different embedding dimension in state-space reconstruction and it is not clear if the mean or best run has been reported. We show further comparison for Mackey-Glass for 5th prediction horizon using Extended Kalman Filtering (EKF), the Unscented Kalman Filtering (UKF) and the Gaussian Particle Filtering (GPF), along with their generalized versions G-EKF, G-UKF and G-GPF, respectively [111]. Considering MultiTL-KELM [55] for Mackey-Glass and Henon time series, it is observed that it is performing very well for the Mackey-Glass time series and beats all our proposed methods but fails to perform better for the Henon time series. In general, we find that our proposed deep learning methods (LSTM, BD-LSTM, ED-LSTM) have beaten most of the methods from the literature for the simulated time series, except for the Mackey-Glass time series.

In Table 4, we compare the performance of Sunspot time series with support vector regression(SVR), iterated (SVR-I), direct (SVR-D), and multiple models (M-SVR) methods [14]. In the respective problems, we also compare with coevolutionary multi-task learning (CMTL) [54]. We observe that our proposed deep learning methods have given the best performance for the respective problems for most of the prediction horizon. Moreover, we find the FNN-Adam overtakes CMTL in all time-series problems except in 8-step ahead prediction in Mackey-Glass and 2-step ahead prediction in Lazer time series. It should also be noted that except for the Mackey-Glass and ACI-Finance time series, the deep learning methods are the best which motivates further applications for these methods for challenging forecasting problems.

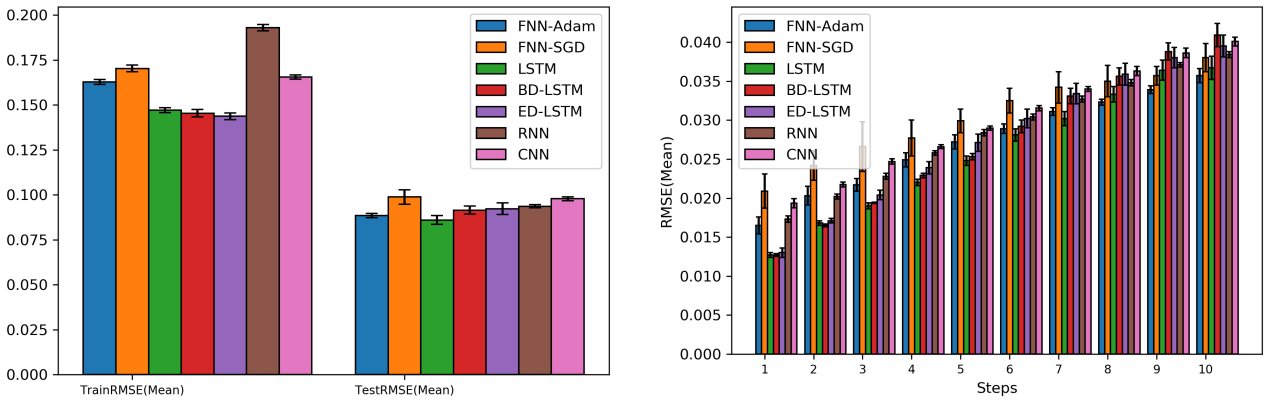
5. Discussion

We provide a ranking of the methods in terms of performance accuracy over the test dataset across the prediction horizons in Table 2. We observe that FNN-SGD gives the worst performance for all time-series problems followed by FNN-Adam in most cases, which is further followed by RNN and CNN. We observe that the BD-LSTM and ED-LSTM models provide one of the best performance across different problems with differed properties. We also note that in across all the problems, the confidence interval of RNN is the lowest followed by CNN which indicates that they provide more robust training performance given different initialisation in weight space.

We note that its natural for the results to deteriorate as the prediction horizons increases in multi-step ahead problems since the prediction is based on current values and the gap in the missing information increases as the horizon decreases since the next predictions are not used as inputs due to our problem formulated as direct strategy of multi-step ahead prediction, as opposed to iterated prediction strategies. ACI-finance problem is unique in since there is not major difference with simple neural networks and deep learning models (Figure 7 b) from 7 - 10 prediction horizon.

Long term dependency problems arise in the analysis of time series where the rate of decay of statistical dependence of two points with increasing time interval between the points. Canonical RNNs had difficulties in training with long-term dependencies [29], hence LSTM networks were proposed to address the learning imitates with memory cells for addressing vanishing error problem in learning long term dependencies [22]. We note that the time series problems in our experiments are not long-term dependency problems, yet LSTM give better performance when compared to simple RNNs. It seems that the memory gates in LSTM networks help better capture information in temporal sequences, even though they do not have long-term dependencies. We note that the memory gates in LSTM networks were originally designed to cater for the vanishing gradient problem. It seems the memory gates of LSTM networks are helpful in capturing salient features in temporal series that help in predicting future tends much better than simple RNNs. We note that simple RNNs provided better results than simple neural networks (FNN-SGD and FNN-Adam) since they are more suited for temporal series. Moreover, we find striking results given that CNNs which are suited for image processing task performances better than simple RNNs in general. This could be due to the convolutional layers in CNNs that help in better capturing salient features for the temporal sequences.

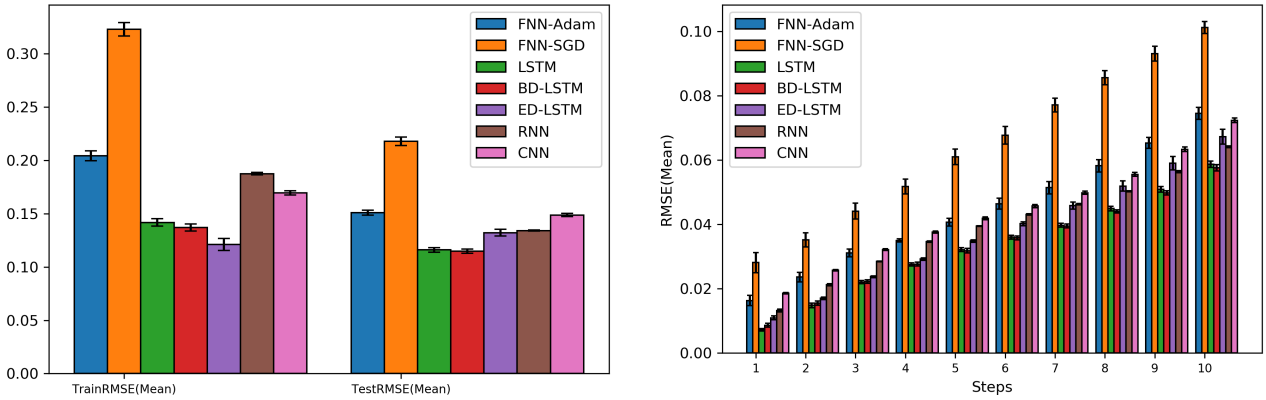
Moving on, it is important to understand why the novel LSTM network model (ED-LSTM and BD-LSTM) have given much better results. ED-LSTMS were designed for language



(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

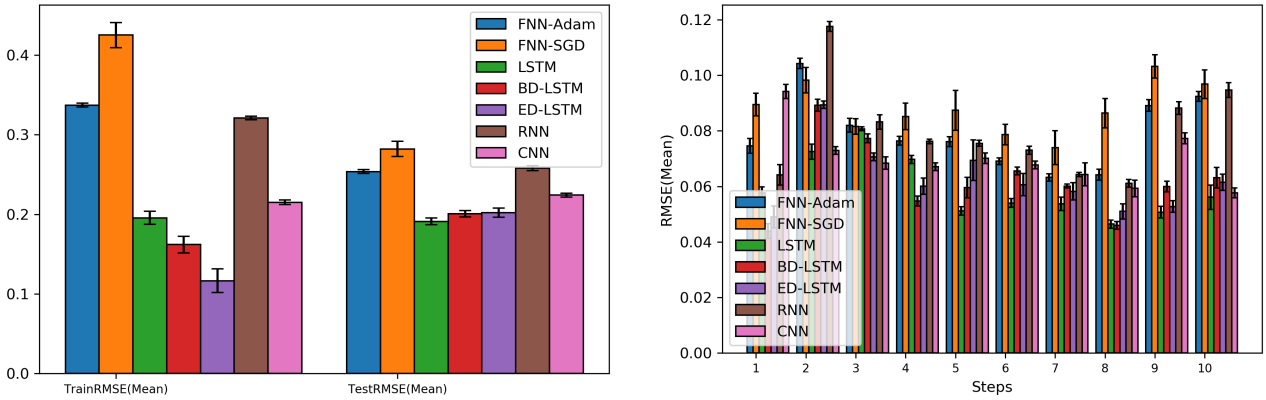
Figure 6: ACI-finance time series: performance evaluation of respective methods



(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

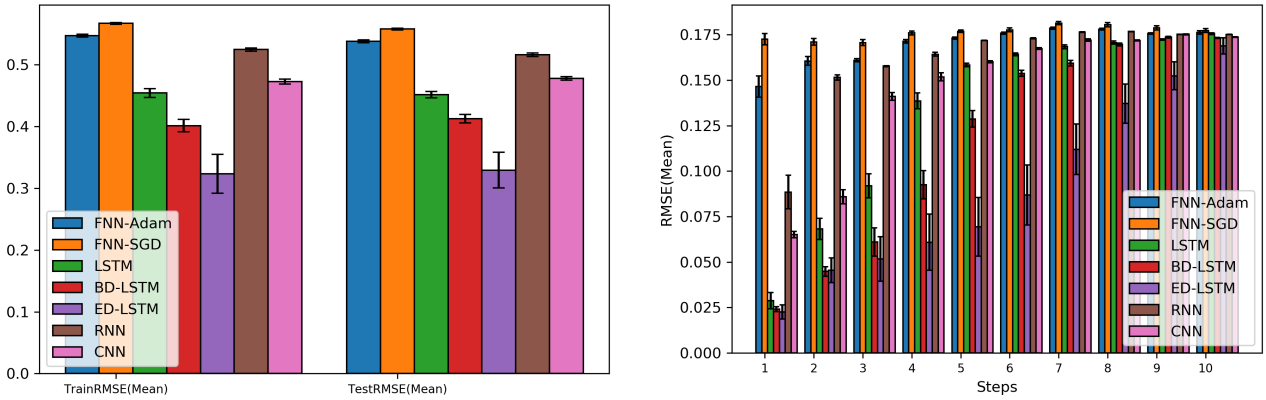
Figure 7: Sunspot time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)



(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

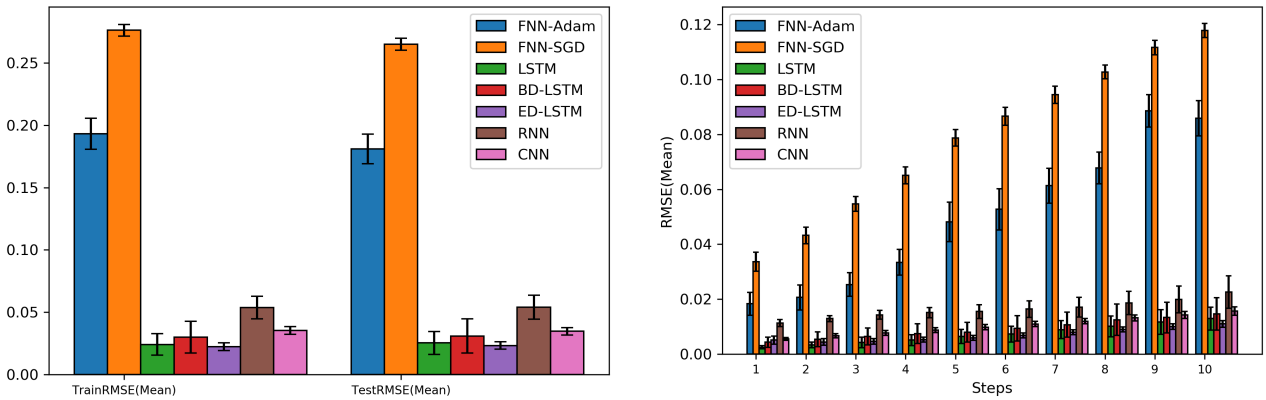
Figure 8: Lazer time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)



(a) RMSE-Mean

(b) 10 step-ahead prediction

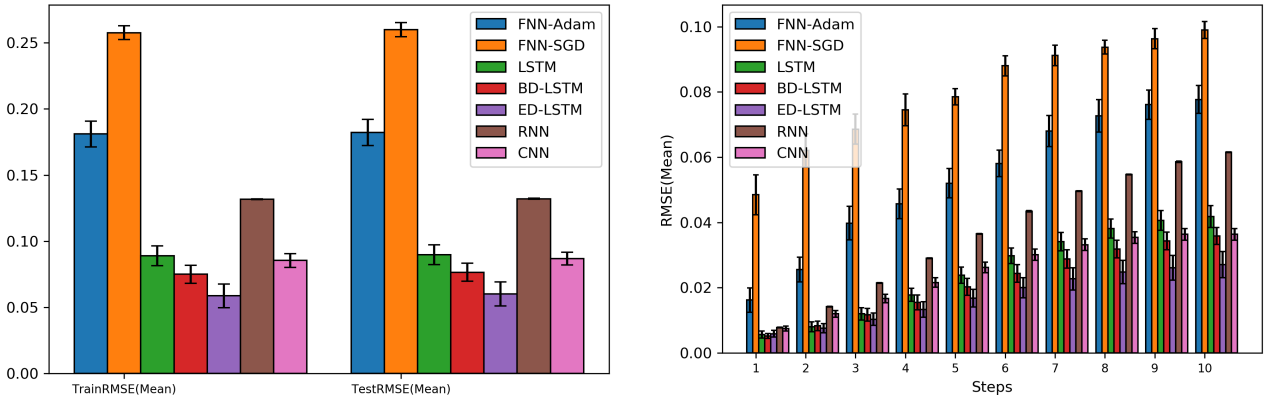
Figure 9: Henon time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)



(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

Figure 10: Lorenz time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)

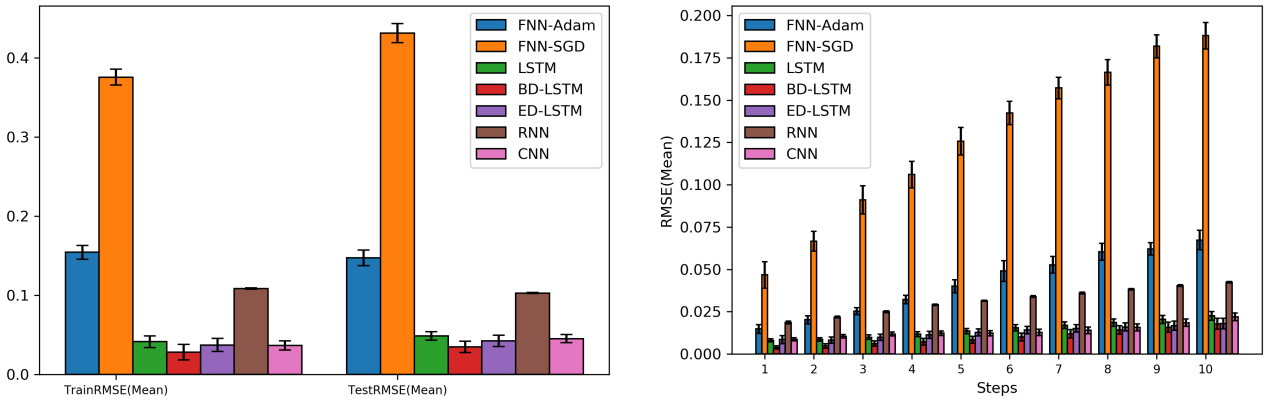


(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

Figure 11: Mackey-Glass time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)

modeling tasks, primarily sequence to sequence model for language translation where encoder LSTM maps a source sequence to a fixed-length vector, and the decoder LSTM maps the vector representation back to a variable-length target se-



(a) RMSE across 10 prediction horizons

(b) 10 step-ahead prediction

Figure 12: Rossler time series: performance evaluation of respective methods (RMSE mean and 95% confidence interval as error bar)

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
ACI-finance	2	7	1	3	4	5	6
Sunspot	6	7	2	1	3	4	5
Lazer	6	7	1	2	3	5	4
Henon	6	7	3	2	1	5	4
Lorenz	6	7	2	3	1	5	4
Mackey-Glass	6	7	4	2	1	5	1
Rossler	6	7	4	1	2	5	3
Mean-Rank	5.42	7.00	2.42	2.00	2.14	4.85	3.85

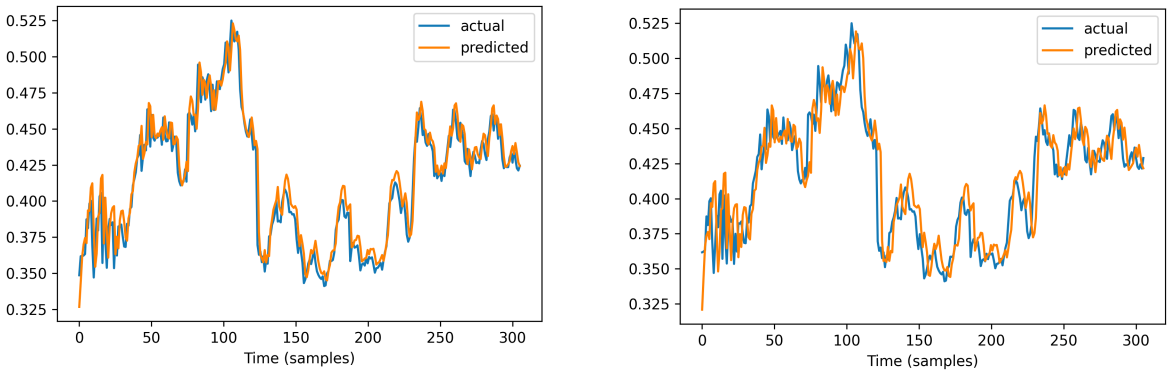
Table 2: Performance (rank) of different models for respective time-series problems. Note lower rank denotes better performance.

Table 3: Comparison with Literature for Simulated time series.

Problem	Method	2-step	5-step	8-step	10 steps
Mackey-Glass					
	2SA-RTRL* [12]	0.0035			
	ESN*[12]	0.0052			
	EKF[111]		0.2796		
	G-EKF [111]		0.2202		
	UKF [111]		0.1374		
	G-UKF [111]		0.0509		
	GPF[111]		0.0063		
	G-GPF[111]		0.0022		
	Multi-KELM[55]	0.0027	0.0031	0.0028	0.0029
	MultiTL-KELM[55]	**0.0025**	**0.0029**	**0.0026**	**0.0028**
	CMTL [54]	0.0550	0.0750	0.0105	0.1200
	ANFIS(SL) [112]	0.0051	0.0213	0.0547	
	R-ANFIS(SL) [112]	0.0045	0.0195	0.0408	
	R-ANFIS(GL) [112]	0.0042	0.0127	0.0324	
	FNN-Adam	0.0256± 0.0038	0.0520± 0.0044	0.0727± 0.0050	0.0777± 0.0043
	FNN-SGD	0.0621± 0.0051	0.0785± 0.0025	0.0937± 0.0022	0.0990± 0.0026
	LSTM	0.0080± 0.0014	0.0238± 0.0024	0.0381± 0.0029	0.0418± 0.0033
	BD-LSTM	0.0083± 0.0015	0.0202± 0.0026	0.0318± 0.0027	0.0359± 0.0026
	ED-LSTM	0.0076± 0.0014	0.0168± 0.0027	0.0248± 0.0036	0.0271± 0.0040
	RNN	0.0142± 0.0001	0.0365± 0.0001	0.0547± 0.0001	0.0615± 0.0001
	CNN	0.0120± 0.0010	0.0262± 0.0016	0.0354± 0.0018	0.0364± 0.0017
Lorenz					
	2SA-RTRL*[12]	0.0382			
	ESN*[12]	0.0476			
	CMTL [54]	0.0490	0.0550	0.0710	0.0820
	FNN-Adam	0.0206± 0.0046	0.0481± 0.0072	0.0678± 0.0058	0.0859± 0.0065
	FNN-SGD	0.0432± 0.0030	0.0787± 0.0030	0.1027± 0.0025	0.1178± 0.0026
	LSTM	**0.0033**± 0.0010	0.0064± 0.0026	0.0101± 0.0038	0.0129± 0.0042
	BD-LSTM	0.0054± 0.0026	0.0079± 0.0036	0.0125± 0.0057	0.0146± 0.0059
	ED-LSTM	0.0044± 0.0012	**0.0059± 0.0009**	**0.0090± 0.0009**	**0.0110± 0.0012**
	RNN	0.0129± 0.0012	0.0155± 0.0024	0.0186± 0.0042	0.0226± 0.0058
	CNN	0.0067 ±0.0007	0.0098± 0.0009	0.0132± 0.0011	0.0157± 0.0015
Rossler					
	CMTL [54]	0.0421	0.0510	0.0651	0.0742
	FNN-Adam	0.0202± 0.0024	0.0400± 0.0039	0.0603± 0.0050	0.0673± 0.0056
	FNN-SGD	0.0666± 0.0058	0.1257± 0.0082	0.1664± 0.0075	0.1881± 0.0078
	LSTM	0.0086± 0.0011	0.0135± 0.0015	0.0185± 0.0022	0.0225± 0.0026
	BD-LSTM	**0.0047± 0.0014**	**0.0084± 0.0021**	**0.0142± 0.0027**	**0.0178± 0.0032**
	ED-LSTM	0.0082± 0.0019	0.0128± 0.0021	0.0159± 0.0024	0.0180± 0.0030
	RNN	0.0218± 0.0005	0.0314± 0.0004	0.0382± 0.0004	0.0424± 0.0004
	CNN	0.0105± 0.0011	0.0122± 0.0016	0.0157± 0.0020	0.0220± 0.0022
Henon					
	Multi-KELM[55]	0.0041	0.2320	0.2971	0.2968
	MultiTL-KELM[55]	**0.0031**	0.1763	0.2452	0.2516
	CMTL [54]	0.2103	0.2354	0.2404	0.2415
	FNN-Adam	0.1606 ± 0.0024	0.1731 ± 0.0005	0.1781 ± 0.0005	0.1762 ± 0.0009
	FNN-SGD	0.1711 ± 0.0018	0.1769 ± 0.0007	0.1805 ± 0.0012	0.1773 ± 0.0011
	LSTM	0.0682 ± 0.0058	0.1584 ± 0.0010	0.1707 ± 0.0008	0.1756 ± 0.0005
	BD-LSTM	0.0448 ± 0.0026	0.1287 ± 0.0046	0.1697 ± 0.0008	0.1733 ± 0.0003
	ED-LSTM	0.0454 ± 0.0069	**0.0694 ± 0.0161**	**0.1371 ± 0.0107**	**0.1689 ± 0.0046**
	RNN	0.1515± 0.0016	0.1718 ± 0.0001	0.1768 ± 0.0001	0.1751 ± 0.0002
	CNN	0.0859 ± 0.0038	0.1601± 0.0007	0.1718± 0.0003	0.1737± 0.0002

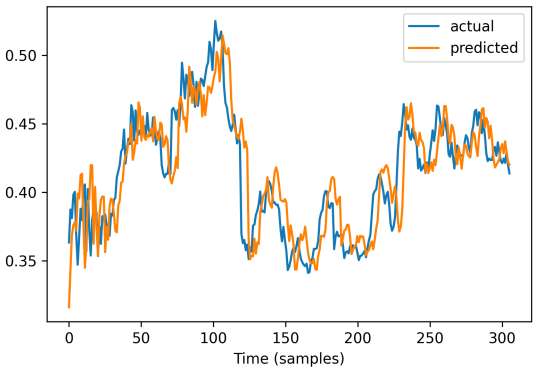
Table 4: Comparison with Literature for Real World time series.

Problem	Method	2-step	5-step	8-step	10-step
Lazer					
	CMTL [54]	0.0762	0.1333	0.1652	0.1885
	FNN-Adam	0.1043 ± 0.0018	0.0761 ± 0.0019	0.0642 ± 0.0020	0.0924± 0.0018
	FNN-SGD	0.0983 ± 0.0046	0.0874 ± 0.0072	0.0864± 0.0053	0.0968± 0.0052
	LSTM	**0.0725 ± 0.0027**	**0.0512 ± 0.0015**	0.0464± 0.0015	**0.0561± 0.0044**
	BD-LSTM	0.0892 ± 0.0022	0.0596 ± 0.0036	**0.0460± 0.0015**	0.0631± 0.0037
	ED-LSTM	0.0894 ± 0.0013	0.0694 ± 0.0073	0.0510± 0.0027	0.0615± 0.0030
	RNN	0.1176 ± 0.0019	0.0755 ± 0.0011	0.0611± 0.0015	0.0947± 0.0027
	CNN	0.0729± 0.0014	0.0701± 0.0020	0.0593± 0.0029	0.0577± 0.0018
Sunspot					
	M-SVR [14]				0.2355 ± 0.0583
	SVR-I [14]				0.2729 ±0.1414
	SVR-D [14]				0.2151 ± 0.0538
	CMTL [54]	0.0473	0.0623	0.0771	0.0974
	FNN-Adam	0.0236± 0.0015	0.0407± 0.0012	0.0582± 0.0019	0.0745± 0.0020
	FNN-SGD	0.0352± 0.0022	0.0610± 0.0024	0.0856± 0.0023	0.1012± 0.0019
	LSTM	**0.0148± 0.0007**	0.0321± 0.0006	0.0449± 0.0007	0.0587± 0.0010
	BD-LSTM	0.0155± 0.0007	**0.0318± 0.0007**	**0.0440± 0.0005**	**0.0576± 0.0010**
	ED-LSTM	0.0170± 0.0004	0.0348± 0.0004	0.0519± 0.0016	0.0673± 0.0022
	RNN	0.0212± 0.0003	0.0395± 0.0002	0.0503± 0.0002	0.0641± 0.0003
	CNN	0.0257± 0.0002	0.0419 ±0.0004	0.0555± 0.0006	0.0723± 0.0008
ACI-Finance					
	CMTL [54]	0.0486	0.0755	0.08783	0.1017
	FNN-Adam	0.0203 ± 0.0012	0.0272 ± 0.0008	**0.0323 ± 0.0004**	**0.0357 ± 0.0008**
	FNN-SGD	0.0242 ± 0.0020	0.0299 ± 0.0015	0.0350 ± 0.0021	0.0380 ± 0.0018
	LSTM	0.0168 ± 0.0003	**0.0248 ± 0.0006**	0.0333 ± 0.0010	0.0367± 0.0015
	BD-LSTM	**0.0165 ± 0.0002**	0.0253 ± 0.0004	0.0356 ± 0.0010	0.0409 ± 0.0015
	ED-LSTM	0.0171 ± 0.0003	0.0271 ± 0.0010	0.0359 ± 0.0014	0.0395 ± 0.0014
	RNN	0.0202 ± 0.0003	0.0284 ± 0.0004	0.0348 ± 0.0004	0.0384 ± 0.0003
	CNN	0.0217 ±0.0004	0.0290 ±0.0002	0.0363 ±0.0006	0.0401± 0.0005

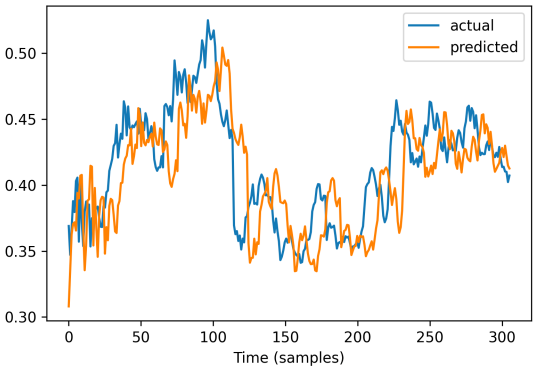


(a) Step 1

(b) Step 3



(c) Step 5



(d) Step 10

Figure 13: ACI-finance actual vs predicted values for Encoder-Decoder LSTM Model

quence [38]. In our case, the encoder maps an input time series to a fixed length vector and then the decoder LSTM maps the vector representation to the different prediction horizons. Although the application is different, the underlying task of mapping inputs to outputs remains the same and hence, ED-LSTM models have been very effective for multi-step ahead prediction.

We note that conventional recurrent networks make use of only the previous context states for determining future states. BD-LSTMS on the other hand processes information using two LSTM models to feature forward and backward information about the sequence at every time step [94]. Although these have been useful for language modelling tasks, our results show that they are applicable for mapping current and future states for time series modelling since information from past and future states are preserved that seems to be the key feature in achieving better performance for multi-step prediction problems when compared to conventional LSTM models.

6. Conclusion and Future Work

In this paper, we provide a comprehensive evaluation of emerging deep learning models for multi-step-ahead time series problems. Our results indicate that encoder-decoder and bidirectional LSTM networks provide best performance for both simulated and real-world time series problems. The results have significantly improved over other related time series prediction methods given in the literature.

In future work, it would be worthwhile to provide similar evaluation for multivariate time series prediction problems. Moreover, it is worthwhile to investigate the performance of given deep learning models for spatial-temporal time series, such as the prediction of behavior of storms and cyclone and also further applications in other real-world problems such as air pollution and energy forecasting.

Software and Data

We provide open source implementation in Python along with data for the respective methods for further research [2].

Appendix

References

[1] A. Tealab, "Time series forecasting using artificial neural networks methodologies: A systematic review," *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, 2018.

[2] `https://github.com/sydney-machine-learning/deeplearning_timeseries`

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.1628 ± 0.0012	0.1703 ± 0.0018	0.1471 ± 0.0014	0.1454 ± 0.0021	0.1437 ± 0.0019	0.1930 ± 0.0018	0.1655 ±0.0013
Test	0.0885 ± 0.0011	0.0988 ± 0.0040	0.0860 ± 0.0025	0.0915 ± 0.0023	0.0923 ± 0.0032	0.0936 ± 0.0009	0.0978± 0.0011
Step-1	0.0165 ± 0.0011	0.0209 ± 0.0022	0.0127 ± 0.0003	0.0127 ± 0.0002	0.0130 ± 0.0005	0.0173 ± 0.0004	0.0193± 0.0006
Step-2	0.0203 ± 0.0012	0.0242 ± 0.0020	0.0168 ± 0.0003	0.0165 ± 0.0002	0.0171 ± 0.0003	0.0202 ± 0.0003	0.0217 ±0.0004
Step-3	0.0217 ± 0.0008	0.0266 ± 0.0032	0.0190 ± 0.0004	0.0194 ± 0.0002	0.0204 ± 0.0006	0.0228 ± 0.0003	0.0247± 0.0003
Step-4	0.0249 ± 0.0009	0.0277 ± 0.0023	0.0220 ± 0.0004	0.0229 ± 0.0003	0.0239 ± 0.0008	0.0258 ± 0.0004	0.0266± 0.0002
Step-5	0.0272 ± 0.0008	0.0299 ± 0.0015	0.0248 ± 0.0006	0.0253 ± 0.0004	0.0271 ± 0.0010	0.0284 ± 0.0004	0.0290 ±0.0002
Step-6	0.0289 ± 0.0006	0.0325 ± 0.0016	0.0281 ± 0.0008	0.0292 ± 0.0007	0.0302 ± 0.0012	0.0304 ± 0.0004	0.0315 ±0.0004
Step-7	0.0311 ± 0.0005	0.0342 ± 0.0020	0.0302 ± 0.0008	0.0331 ± 0.0010	0.0334 ± 0.0014	0.0327 ± 0.0004	0.0340± 0.0003
Step 8	0.0323 ± 0.0004	0.0350 ± 0.0021	0.0333 ± 0.0010	0.0356 ± 0.0010	0.0359 ± 0.0014	0.0348 ± 0.0004	0.0363 ±0.0006
Step 9	0.0339 ± 0.0005	0.0357 ± 0.0012	0.0364 ± 0.0013	0.0388 ± 0.0011	0.0380 ± 0.0014	0.0371 ± 0.0003	0.0386± 0.0006
Step 10	0.0357 ± 0.0008	0.0380 ± 0.0018	0.0367± 0.0015	0.0409 ± 0.0015	0.0395 ± 0.0014	0.0384 ± 0.0003	0.0401± 0.0005

Table 5: ACI-finance reporting RMSE mean and 95 % confidence interval (±).

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.2043± 0.0047	0.3230± 0.0064	0.1418± 0.0034	0.1369± 0.0033	0.1210± 0.0057	0.1875± 0.0011	0.1695± 0.0019
Test	0.1510± 0.0024	0.2179± 0.0041	0.1160± 0.0021	0.1147± 0.0020	0.1322± 0.0032	0.1342± 0.0004	0.1487± 0.0015
Step-1	0.0163± 0.0016	0.0281± 0.0032	0.0072± 0.0005	0.0086± 0.0006	0.0109± 0.0006	0.0132± 0.0004	0.0186± 0.0002
Step-2	0.0236± 0.0015	0.0352± 0.0022	0.0148± 0.0007	0.0155± 0.0007	0.0170± 0.0004	0.0212± 0.0003	0.0257± 0.0002
Step-3	0.0311± 0.0013	0.0441± 0.0025	0.0220± 0.0005	0.0222± 0.0006	0.0237± 0.0003	0.0285± 0.0002	0.0321± 0.0003
Step-4	0.0350± 0.0006	0.0518± 0.0022	0.0275± 0.0005	0.0276± 0.0006	0.0292± 0.0003	0.0346± 0.0002	0.0376± 0.0003
Step-5	0.0407± 0.0012	0.0610± 0.0024	0.0321± 0.0006	0.0318± 0.0007	0.0348± 0.0004	0.0395± 0.0002	0.0419 ±0.0004
Step-6	0.0464± 0.0016	0.0677± 0.0027	0.0360± 0.0006	0.0358± 0.0006	0.0402± 0.0006	0.0431± 0.0002	0.0457± 0.0004
Step-7	0.0514± 0.0019	0.0771± 0.0020	0.0397± 0.0006	0.0395± 0.0006	0.0458± 0.0011	0.0463± 0.0002	0.0498± 0.0005
Step 8	0.0582± 0.0019	0.0856± 0.0023	0.0449± 0.0007	0.0440± 0.0005	0.0519± 0.0016	0.0503± 0.0002	0.0555± 0.0006
Step 9	0.0653± 0.0016	0.0931± 0.0023	0.0509± 0.0009	0.0498± 0.0007	0.0590± 0.0020	0.0564± 0.0002	0.0633± 0.0007
Step 10	0.0745± 0.0020	0.1012± 0.0019	0.0587± 0.0010	0.0576± 0.0010	0.0673± 0.0022	0.0641± 0.0003	0.0723± 0.0008

Table 6: Sunspot reporting RMSE mean and 95 % confidence interval (±).

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.3371 ± 0.0026	0.4251 ± 0.0157	0.1954 ± 0.0082	0.1619 ± 0.0106	0.1166 ± 0.0147	0.3210 ± 0.0020	0.2151± 0.0029
Test	0.2537 ± 0.0024	0.2821 ± 0.0098	0.1910 ± 0.0042	0.2007 ± 0.0042	0.2020 ± 0.0057	0.2580 ± 0.0031	0.2240± 0.0025
Step-1	0.0746 ± 0.0027	0.0895 ± 0.0041	0.0577 ± 0.0021	0.0439 ± 0.0027	0.0490 ± 0.0039	0.0641 ± 0.0037	0.0942± 0.0025
Step-2	0.1043 ± 0.0018	0.0983 ± 0.0046	0.0725 ± 0.0027	0.0892 ± 0.0022	0.0894 ± 0.0013	0.1176 ± 0.0019	0.0729± 0.0014
Step-3	0.0820 ± 0.0026	0.0816 ± 0.0028	0.0807 ± 0.0007	0.0773 ± 0.0016	0.0707 ± 0.0014	0.0832 ± 0.0026	0.0684± 0.0022
Step-4	0.0764 ± 0.0017	0.0852 ± 0.0048	0.0697 ± 0.0015	0.0547 ± 0.0018	0.0601 ± 0.0029	0.0762 ± 0.0008	0.0671± 0.0013
Step-5	0.0761 ± 0.0019	0.0874 ± 0.0072	0.0512 ± 0.0015	0.0596 ± 0.0036	0.0694 ± 0.0073	0.0755 ± 0.0011	0.0701± 0.0020
Step-6	0.0691 ± 0.0013	0.0787 ± 0.0037	0.0540 ± 0.0016	0.0655 ± 0.0014	0.0606 ± 0.0041	0.0730 ± 0.0015	0.0677 ±0.0014
Step-7	0.0632 ± 0.0013	0.0740 ± 0.0061	0.0537 ± 0.0024	0.0601 ± 0.0007	0.0582 ± 0.0031	0.0643 ± 0.0009	0.0643 ± 0.0041
Step 8	0.0642 ± 0.0020	0.0864± 0.0053	0.0464± 0.0015	0.0460± 0.0015	0.0510± 0.0027	0.0611± 0.0015	0.0593± 0.0029
Step 9	0.0891± 0.0021	0.1032± 0.0042	0.0507± 0.0021	0.0599± 0.0019	0.0527± 0.0021	0.0882± 0.0023	0.0773± 0.0019
Step 10	0.0924± 0.0018	0.0968± 0.0052	0.0561± 0.0044	0.0631± 0.0037	0.0615± 0.0030	0.0947± 0.0027	0.0577± 0.0018

Table 7: Lazer reporting RMSE mean and 95 % confidence interval (±).

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.5470 ± 0.0023	0.5670 ± 0.0015	0.4542 ± 0.0071	0.4014 ± 0.0100	0.3235 ± 0.0316	0.5247 ± 0.0027	0.4728± 0.0038
Test	0.5378 ± 0.0022	0.5578 ± 0.0016	0.4516 ± 0.0052	0.4127 ± 0.0066	0.3294 ± 0.0290	0.5162 ± 0.0027	0.4779± 0.0027
Step-1	0.1465 ± 0.0058	0.1725 ± 0.0031	0.0287 ± 0.0045	0.0241 ± 0.0014	0.0226 ± 0.0039	0.0885 ± 0.0093	0.0650± 0.0018
Step-2	0.1606 ± 0.0024	0.1711 ± 0.0018	0.0682 ± 0.0058	0.0448 ± 0.0026	0.0454 ± 0.0069	0.1515± 0.0016	0.0859 ± 0.0038
Step-3	0.1610 ± 0.0008	0.1707 ± 0.0017	0.0920 ± 0.0066	0.0610 ± 0.0077	0.0517 ± 0.0122	0.1577 ± 0.0003	0.1411± 0.0021
Step-4	0.1714 ± 0.0009	0.1760 ± 0.0011	0.1386 ± 0.0044	0.0925 ± 0.0077	0.0609 ± 0.0154	0.1643 ± 0.0011	0.1519± 0.0022
Step-5	0.1731 ± 0.0005	0.1769 ± 0.0007	0.1584 ± 0.0010	0.1287 ± 0.0046	0.0694 ± 0.0161	0.1718 ± 0.0001	0.1601± 0.0007
Step-6	0.1758 ± 0.0006	0.1777 ± 0.0010	0.1642 ± 0.0007	0.1538 ± 0.0017	0.0868 ± 0.0165	0.1730 ± 0.0003	0.1674± 0.0004
Step-7	0.1786 ± 0.0006	0.1814 ± 0.0009	0.1684± 0.0011	0.1593 ± 0.0016	0.1120 ± 0.0139	0.1764 ± 0.0003	0.1721± 0.0006
Step 8	0.1781 ± 0.0005	0.1805 ± 0.0012	0.1707 ± 0.0008	0.1697 ± 0.0008	0.1371 ± 0.0107	0.1768 ± 0.0001	0.1718± 0.0003
Step 9	0.1757 ± 0.0004	0.1788 ± 0.0011	0.1723 ± 0.0005	0.1737 ± 0.0004	0.1524 ± 0.0077	0.1752 ± 0.0001	0.1752± 0.0003
Step 10	0.1762 ± 0.0009	0.1773 ± 0.0011	0.1756 ± 0.0005	0.1733 ± 0.0003	0.1689 ± 0.0046	0.1751 ± 0.0002	0.1737± 0.0002

Table 8: Henon reporting RMSE mean and 95 % confidence interval (±).

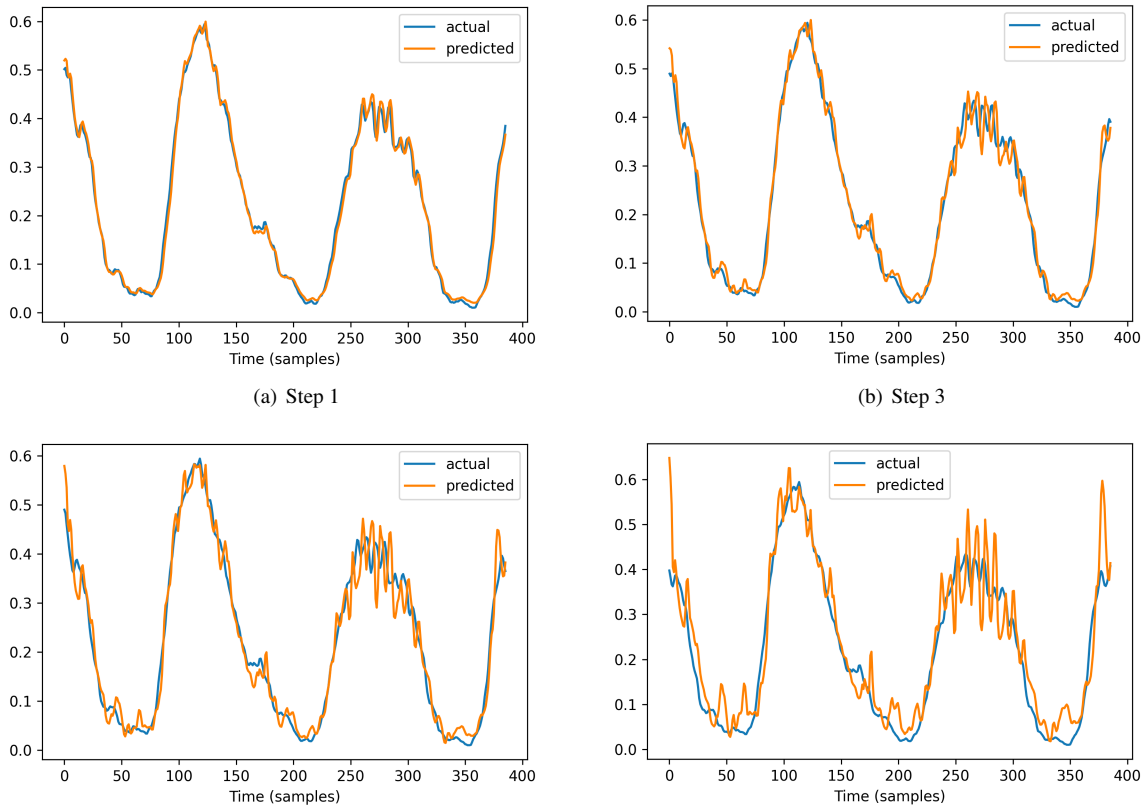
	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.1932± 0.0124	0.2761± 0.0048	0.0242± 0.0086	0.0300± 0.0127	0.0225± 0.0031	0.0538± 0.0091	0.0354± 0.0032
Test	0.1809± 0.0119	0.2649± 0.0048	0.0254± 0.0093	0.0310± 0.0137	0.0234± 0.0031	0.0542± 0.0097	0.0347± 0.0029
Step-1	0.0183± 0.0042	0.0336± 0.0035	0.0025± 0.0006	0.0043± 0.0019	0.0051± 0.0015	0.0113± 0.0013	0.0055± 0.0006
Step-2	0.0206± 0.0046	0.0432± 0.0030	0.0033± 0.0010	0.0054± 0.0026	0.0044± 0.0012	0.0129± 0.0012	0.0067 ±0.0007
Step-3	0.0253± 0.0043	0.0547± 0.0028	0.0042± 0.0019	0.0064± 0.0031	0.0046± 0.0010	0.0143± 0.0016	0.0077± 0.0009
Step-4	0.0334± 0.0048	0.0651± 0.0032	0.0051± 0.0020	0.0074± 0.0035	0.0052± 0.0009	0.0151± 0.0018	0.0087± 0.0009
Step-5	0.0481± 0.0072	0.0787± 0.0030	0.0064± 0.0026	0.0079± 0.0036	0.0059± 0.0009	0.0155± 0.0024	0.0098± 0.0009
Step-6	0.0527± 0.0076	0.0866± 0.0033	0.0073± 0.0029	0.0094± 0.0046	0.0068± 0.0008	0.0164± 0.0029	0.0109 ±0.0010
Step-7	0.0613± 0.0064	0.0944± 0.0031	0.0089± 0.0033	0.0107± 0.0047	0.0079± 0.0009	0.0171± 0.0036	0.0120 ±0.0010
Step 8	0.0678± 0.0058	0.1027± 0.0025	0.0101± 0.0038	0.0125± 0.0057	0.0090± 0.0009	0.0186± 0.0042	0.0132± 0.0011
Step 9	0.0885± 0.0060	0.1116± 0.0027	0.0117± 0.0045	0.0133± 0.0056	0.0100± 0.0010	0.0199± 0.0049	0.0142± 0.0013
Step 10	0.0859± 0.0065	0.1178± 0.0026	0.0129± 0.0042	0.0146± 0.0059	0.0110± 0.0012	0.0226± 0.0058	0.0157± 0.0015

Table 9: Lorenz reporting RMSE mean and 95 % confidence interval (±).

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.1810± 0.0097	0.2576± 0.0053	0.0890± 0.0076	0.0750± 0.0067	0.0587± 0.0090	0.1317± 0.0003	0.0854± 0.0052
Test	0.1822± 0.0098	0.2599± 0.0054	0.0897± 0.0075	0.0765± 0.0068	0.0602± 0.0090	0.1321± 0.0003	0.0868± 0.0048
Step-1	0.0162± 0.0037	0.0485± 0.0061	0.0056± 0.0011	0.0052± 0.0009	0.0059± 0.0010	0.0078± 0.0001	0.0075± 0.0007
Step-2	0.0256± 0.0038	0.0621± 0.0051	0.0080± 0.0014	0.0083± 0.0015	0.0076± 0.0014	0.0142± 0.0001	0.0120± 0.0010
Step-3	0.0398± 0.0051	0.0686± 0.0046	0.0120± 0.0018	0.0117± 0.0019	0.0103± 0.0020	0.0214± 0.0001	0.0167± 0.0013
Step-4	0.0457± 0.0045	0.0745± 0.0049	0.0178± 0.0020	0.0155± 0.0023	0.0133± 0.0024	0.0290± 0.0001	0.0216± 0.0015
Step-5	0.0520± 0.0044	0.0785± 0.0025	0.0238± 0.0024	0.0202± 0.0026	0.0168± 0.0027	0.0365± 0.0001	0.0262± 0.0016
Step-6	0.0581± 0.0042	0.0880± 0.0031	0.0298± 0.0025	0.0244± 0.0027	0.0200± 0.0031	0.0434± 0.0001	0.0301± 0.0017
Step-7	0.0680± 0.0047	0.0912± 0.0031	0.0341± 0.0028	0.0288± 0.0028	0.0227± 0.0033	0.0496± 0.0001	0.0332± 0.0018
Step 8	0.0727± 0.0050	0.0937± 0.0022	0.0381± 0.0029	0.0318± 0.0027	0.0248± 0.0036	0.0547± 0.0001	0.0354± 0.0018
Step 9	0.0761± 0.0046	0.0963± 0.0031	0.0406± 0.0030	0.0343± 0.0027	0.0261± 0.0038	0.0586± 0.0001	0.0364± 0.0017
Step 10	0.0777± 0.0043	0.0990± 0.0026	0.0418± 0.0033	0.0359± 0.0026	0.0271± 0.0040	0.0615± 0.0001	0.0364± 0.0017

±

Table 10: Mackey-Glass reporting RMSE mean and 95 % confidence interval (±).



(a) Step 1 (b) Step 3

(c) Step 5 (d) Step 10

Figure 14: Sunspot actual vs predicted values for Encoder-Decoder LSTM Model

[2] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam, "Time series forecasting for nonlinear and non-stationary processes: A review and comparative study," *Iie Transactions*, vol. 47, no. 10, pp. 1053–1071, 2015.

[3] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition," *Expert systems with applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[4] N. K. Ahmed, A. F. Atiya, N. E. Gayar, and H. El-Shishiny, "An empirical comparison of machine learning models for time series forecasting," *Econometric Reviews*, vol. 29, no. 5-6, pp. 594–621, 2010.

[5] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

[6] G. Li, H. Song, and S. F. Witt, "Recent developments in econometric modeling and forecasting," *Journal of Travel Research*, vol. 44, no. 1, pp. 82–99, 2005.

[7] D. F. Hendry and J.-F. Richard, "The econometric analysis of economic time series," *International Statistical Review/Revue Internationale de Statistique*, pp. 111–148, 1983.

[8] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning for dynamic time series prediction," *Applied Soft Computing*, vol. 70, pp. 576–589, 2018.

[9] H. Sandya, P. Hemanth Kumar, and S. B. Patil, "Feature extraction, classification and forecasting of time series signal using fuzzy and garch techniques," in *Research & Technology in the Coming Decades (CRT 2013), National Conference on Challenges in*. IET, 2013, pp. 1–7.

[10] R. Chandra and M. Zhang, "Cooperative coevolution of elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 86, pp. 116–123, 2012.

[11] S. B. Taieb and A. F. Atiya, "A bias and variance analysis for multistep-ahead time series forecasting," 2015.

[12] L.-C. Chang, P.-A. Chen, and F.-J. Chang, "Reinforced two-step-ahead weight adjustment technique for online training of recurrent neural networks," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1269–1278, 2012.

[13] R. Boné and M. Crucianu, "Multi-step-ahead prediction with neural networks: a review," *9emes rencontres internationales: Approches Connexionnistes en Sciences*, vol. 2, pp. 97–106, 2002.

[14] L. Zhang, W.-D. Zhou, P.-C. Chang, J.-W. Yang, and F.-Z. Li, "Iterated time series prediction with multiple support vector regression models," *Neurocomputing*, vol. 99, pp. 411–422, 2013.

[15] S. Ben Taieb and R. Hyndman, "Recursive and direct multi-step forecasting: the best of both worlds," Monash University, Department of Econometrics and Business Statistics, Tech. Rep., 2012.

[16] "Methodology for long-term prediction of time series," *Neurocomputing*, vol. 70, no. 16-18, pp. 2861–2869, 2007.

[17] S. B. Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, vol. 73, no. 10–12, pp. 1950 – 1957, 2010.

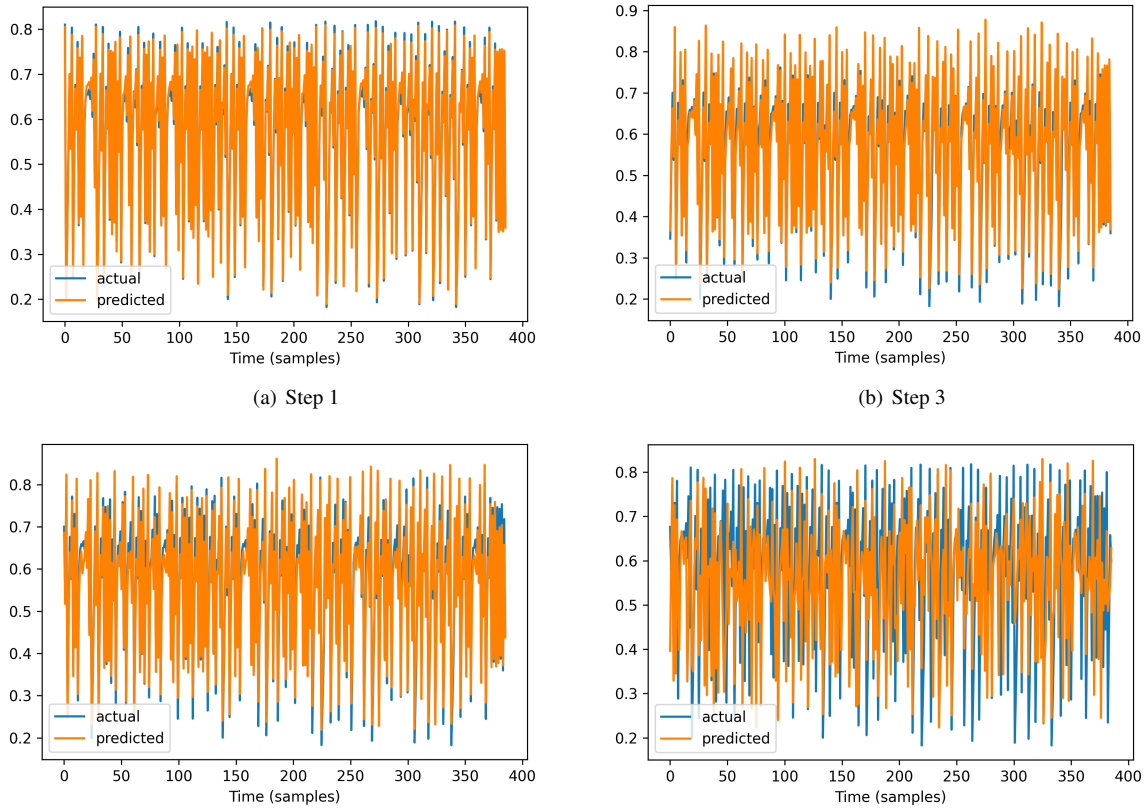
[18] X. Wu, Y. Wang, J. Mao, Z. Du, and C. Li, "Multi-step prediction of time series with random missing data," *Applied Mathematical Modelling*, vol. 38, no. 14, pp. 3512 – 3522, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0307904X13007658

[19] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of intelligent and robotic systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[20] J. L. Elman and D. Zipser, "Learning the hidden structure of speech," *The Journal of the Acoustical Society of America*, vol. 83, no. 4, pp. 1615–1626, 1988. [Online]. Available: http://link.aip.org/link/?JAS/83/1615/1

[21] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural*



(a) Step 1

(b) Step 3

(c) Step 5

(d) Step 10

Figure 15: Henon actual vs predicted values for Encoder-Decoder LSTM Model

computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[23] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[24] C. W. Omlin, K. K. Thornber, and C. L. Giles, "Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks," *IEEE Transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 76–89, 1998.

[25] C. W. Omlin and C. L. Giles, "Training second-order recurrent neural networks using hints," in *Proceedings of the Ninth International Conference on Machine Learning*. Morgan Kaufmann, 1992, pp. 363–368.

[26] C. L. Giles, C. W. Omlin, and K. K. Thornber, "Equivalence in knowledge representation: automata, recurrent neural networks, and dynamical fuzzy systems," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1623–1640, 1999.

[27] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 240–254, 1994.

[28] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[29] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[30] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[31] 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.

[32] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[33] C. Downey, A. Hefny, B. Boots, G. J. Gordon, and B. Li, "Predictive state recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6053–6064.

[34] S. Singh, M. R. James, and M. R. Rudary, "Predictive state representations: A new theory for modeling dynamical systems," in *Proceedings of the 20th conference on Uncertainty in artificial intelligence*. AUAI Press, 2004, pp. 512–519.

[35] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[36] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[37] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 357–370, 2016.

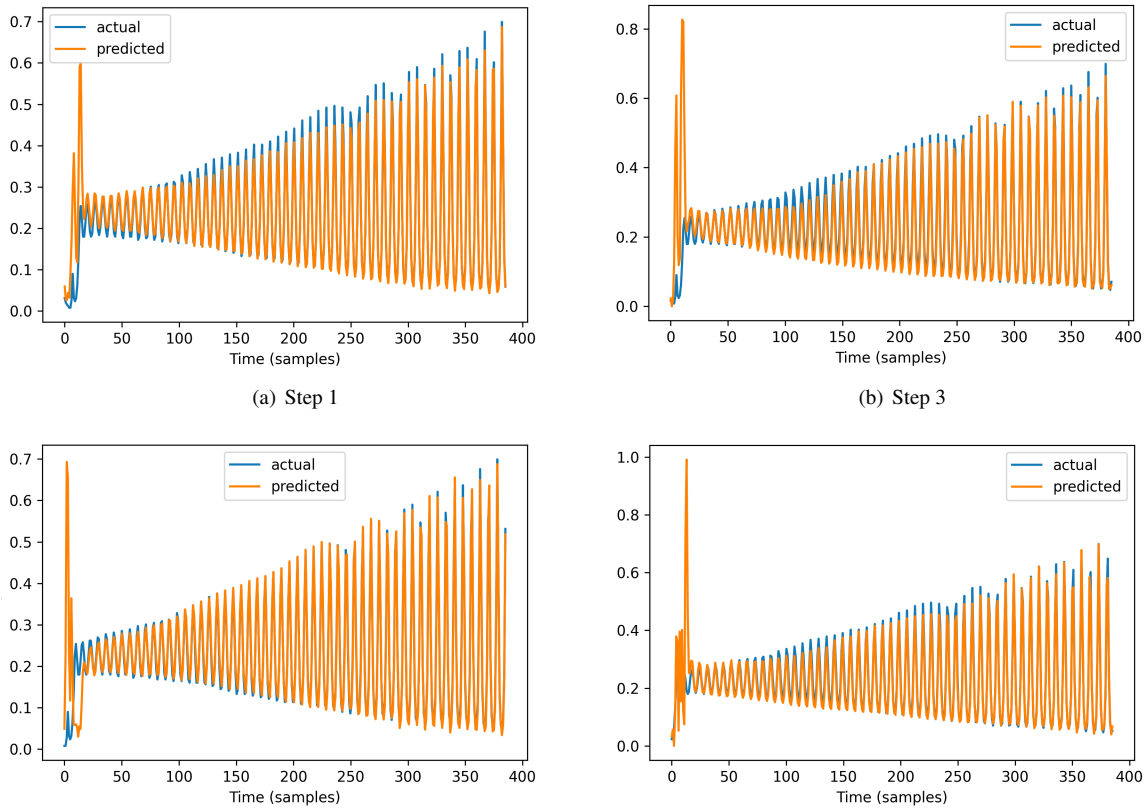
[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[40] J. M. Adams, N. M. Gasparini, D. E. J. Hobley, G. E. Tucker, E. W. H. Hutton, S. S. Nudurupati, and E. Istanbulluoglu, "The landlab v1.0 overlandflow component: a python tool for computing shallow-water flow across watersheds," *Geoscientific Model Development*, vol. 10, no. 4, pp. 1645–1663, 2017.

[41] R. Chandra and S. Cripps, "Coevolutionary multi-task learning for feature-based modular pattern classification," *Neurocomputing*, vol. 319, pp. 164 – 175, 2018.

[42] X. Cai, N. Zhang, G. K. Venayagamoorthy, and D. C. Wunsch II, "Time



(a) Step 1

(b) Step 3

(c) Step 5

(d) Step 10

Figure 16: Lazer actual vs predicted values for Encoder-Decoder LSTM Model

series prediction with recurrent neural networks trained by a hybrid pso–ea algorithm," *Neurocomputing*, vol. 70, no. 13-15, pp. 2342–2353, 2007.

[43] R. Chandra and M. Zhang, "Cooperative coevolution of Elman recurrent neural networks for chaotic time series prediction," *Neurocomputing*, vol. 186, pp. 116 – 123, 2012.

[44] T. Koskela, M. Lehtokangas, J. Saarinen, and K. Kaski, "Time series prediction with multilayer perceptron, fir and elman neural networks," in *Proceedings of the World Congress on Neural Networks*. Citeseer, 1996, pp. 491–496.

[45] R. Chandra and S. Chand, "Evaluation of co-evolutionary neural network architectures for time series prediction with mobile application in finance," *Applied Soft Computing*, vol. 49, pp. 462–473, 2016.

[46] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[47] C. N. Ng and P. C. Young, "Recursive estimation and forecasting of non-stationary time series," *Journal of Forecasting*, vol. 9, no. 2, pp. 173–204, 1990.

[48] H. T. Su, T. J. McAvoy, and P. Werbos, "Long-term predictions of chemical processes using recurrent neural networks: a parallel training approach," *Industrial & Engineering Chemistry Research*, vol. 31, no. 5, pp. 1338–1352, 1992.

[49] A. Parlos, O. Rais, and A. Atiya, "Multi-step-ahead prediction using dynamic recurrent neural networks," *Neural Networks*, vol. 13, no. 7, pp. 765 – 786, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608000000484

[50] A. Girard, C. E. Rasmussen, J. Q. nonero Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. MIT Press, 2003, pp. 545–552. [Online]. Available: http://papers.nips.cc/paper/2313-gaussian-process-priors-with-uncertain-inputs-application-to-multiple-step-a pdf

[51] G. Niu and B.-S. Yang, "Dempster–shafer regression for multi-step-ahead time-series prediction towards data-driven machinery prognosis," *Mechanical Systems and Signal Processing*, vol. 23, no. 3, pp. 740 – 751, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888327008002094

[52] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482 – 493, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121300917X

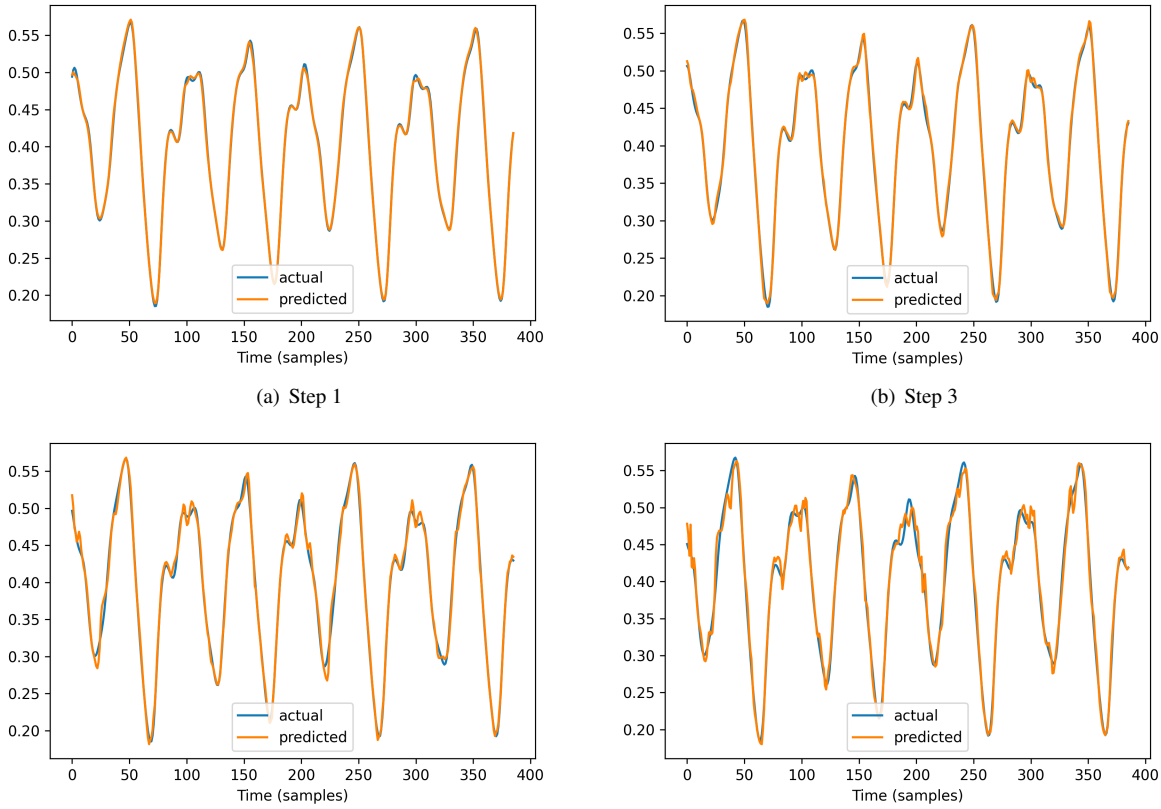
[53] A. Grigorievskiy, Y. Miche, A.-M. Ventelä, E. Séverin, and A. Lendasse, "Long-term time series prediction using op-elm," *Neural Networks*, vol. 51, pp. 50 – 56, 2014.

[54] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction," *Neurocomputing*, vol. 243, pp. 21–34, 2017.

[55] R. Ye and Q. Dai, "Multitl-kelm: A multi-task learning algorithm for multi-step-ahead time series prediction," *Applied Soft Computing*, vol. 79, pp. 227 – 253, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619301656

[56] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the {NN5} forecasting competition," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067 – 7083, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417412000528

[57] M. Marcellino, J. H. Stock, and M. W. Watson, "A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series," *Journal of econometrics*, vol. 135, no. 1, pp. 499–526, 2006.



(a) Step 1

(b) Step 3

(c) Step 5

(d) Step 10

Figure 17: Mackey-Glass actual vs predicted values for Encoder-Decoder LSTM Model

[58] T. Proietti, "Direct and iterated multistep {AR} methods for difference stationary processes," *International Journal of Forecasting*, vol. 27, no. 2, pp. 266 – 280, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207010001068

[59] G. Chevillon, "Multistep forecasting in the presence of location shifts," *International Journal of Forecasting*, vol. 32, no. 1, pp. 121 – 137, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169207015000801

[60] H. ElMoaqet, D. M. Tilbury, and S. K. Ramachandran, "Multi-step ahead predictions for critical levels in physiological time series," *IEEE Transactions on Cybernetics*, vol. 46, no. 7, pp. 1704–1714, July 2016.

[61] P.-A. Chen, L.-C. Chang, and F.-J. Chang, "Reinforced recurrent neural networks for multi-step-ahead flood forecasts," *Journal of Hydrology*, vol. 497, pp. 71 – 79, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022169413004150

[62] F.-J. Chang, P.-A. Chen, Y.-R. Lu, E. Huang, and K.-Y. Chang, "Real-time multi-step-ahead water level forecasting by recurrent neural networks for urban flood control," *Journal of Hydrology*, vol. 517, pp. 836 – 846, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022169414004739

[63] J. Smrekar, P. Potočnik, and A. Senegačnik, "Multi-step-ahead prediction of {NOx} emissions for a coal-based boiler," *Applied Energy*, vol. 106, pp. 89 – 99, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261912007751

[64] M. D. Giorgi, M. Malvoni, and P. Congedo, "Comparison of strategies for multi-step ahead photovoltaic power forecasting models based on hybrid group method of data handling networks and least square support vector machine," *Energy*, vol. 107, pp. 360 – 373, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0360544216304261

[65] D. Yang and K. Yang, "Multi-step prediction of strong earthquake ground motions and seismic responses of {SDOF} systems based on emd-elm method," *Soil Dynamics and Earthquake Engineering*, vol. 85, pp. 117 – 129, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0267726116000695

[66] D. Yang, J. Cao, J. Fu, J. Wang, and J. Guo, "A pattern fusion model for multi-step-ahead cpu load prediction," *Journal of Systems and Software*, vol. 86, no. 5, pp. 1257 – 1266, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212003354

[67] "Prediction of short-term wind and wave conditions for marine operations using a multi-step-ahead decomposition-ANFIS model and quantification of its uncertainty," *Ocean Engineering*, vol. 188, p. 106300, 2019.

[68] J. Wang and Y. Li, "Multi-step ahead wind speed prediction based on optimal feature extraction, long short term memory neural network and error correction strategy," *Applied Energy*, vol. 230, pp. 429 – 443, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261918312844

[69] ——, "An innovative hybrid approach for multi-step ahead wind speed prediction," *Applied Soft Computing*, vol. 78, pp. 296 – 309, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619300985

[70] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[73] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, p. 1, 2015.

	FNN-Adam	FNN-SGD	LSTM	BD-LSTM	ED-LSTM	RNN	CNN
Train	0.1546± 0.0087	0.3757± 0.0100	0.0416± 0.0074	0.0281± 0.0098	0.0374± 0.0082	0.1088± 0.0009	0.0367± 0.0059
Test	0.1473± 0.0098	0.4314± 0.0122	0.0488± 0.0054	0.0349± 0.0070	0.0427± 0.0072	0.1030± 0.0008	0.0454± 0.0052
Step-1	0.0148± 0.0026	0.0467± 0.0077	0.0080± 0.0009	0.0038± 0.0008	0.0085± 0.0025	0.0186± 0.0009	0.0086± 0.0010
Step-2	0.0202± 0.0024	0.0666± 0.0058	0.0086± 0.0011	0.0047± 0.0014	0.0082± 0.0019	0.0218± 0.0005	0.0105± 0.0011
Step-3	0.0252± 0.0022	0.0910± 0.0083	0.0099± 0.0013	0.0061± 0.0017	0.0098± 0.0018	0.0250± 0.0005	0.0118± 0.0011
Step-4	0.0322± 0.0024	0.1060± 0.0078	0.0117± 0.0014	0.0072± 0.0020	0.0112± 0.0021	0.0290± 0.0004	0.0122± 0.0014
Step-5	0.0400± 0.0039	0.1257± 0.0082	0.0135± 0.0015	0.0084± 0.0021	0.0128± 0.0021	0.0314± 0.0004	0.0122± 0.0016
Step-6	0.0490± 0.0063	0.1424± 0.0069	0.0155± 0.0019	0.0100± 0.0023	0.0141± 0.0021	0.0339± 0.0004	0.0128± 0.0019
Step-7	0.0527± 0.0049	0.1572± 0.0063	0.0170± 0.0020	0.0120± 0.0024	0.0151± 0.0022	0.0360± 0.0004	0.0139± 0.0020
Step 8	0.0603± 0.0050	0.1664± 0.0075	0.0185± 0.0022	0.0142± 0.0027	0.0159± 0.0024	0.0382± 0.0004	0.0157± 0.0020
Step 9	0.0621± 0.0036	0.1818± 0.0067	0.0204± 0.0024	0.0157± 0.0030	0.0167± 0.0026	0.0404± 0.0005	0.0185± 0.0021
Step 10	0.0673± 0.0056	0.1881± 0.0078	0.0225± 0.0026	0.0178± 0.0032	0.0180± 0.0030	0.0424± 0.0004	0.0220± 0.0022

Table 11: Rossler reporting RMSE mean and 95 % confidence interval (±).

[74] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.

[75] R. Chandra, "Competition and collaboration in cooperative coevolution of elman recurrent neural networks for time-series prediction," *IEEE Trans. Neural Netw. Learning Syst.*, vol. 26, no. 12, pp. 3123–3136, 2015. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2015.2404823

[76] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, "Deepar: Probabilistic forecasting with autoregressive recurrent networks," *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181 – 1191, 2020.

[77] D. T. Mirikitani and N. Nikolaev, "Recursive bayesian recurrent neural networks for time-series modeling," *IEEE Transactions on Neural Networks*, vol. 21, no. 2, pp. 262–274, 2010.

[78] H.-z. Wang, G.-q. Li, G.-b. Wang, J.-c. Peng, H. Jiang, and Y.-t. Liu, "Deep learning based ensemble approach for probabilistic wind power forecasting," *Applied energy*, vol. 188, pp. 56–70, 2017.

[79] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[80] K. Amarasinghe, D. L. Marino, and M. Manic, "Deep neural networks for energy load forecasting," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1483–1488.

[81] C.-J. Huang and P.-H. Kuo, "A deep cnn-lstm model for particulate matter (pm2.5) forecasting in smart cities," *Sensors (Basel, Switzerland)*, vol. 18, no. 7, 2018.

[82] Y. Sudriani, I. Ridwansyah, and H. A. Rustini, "Long short term memory (lstm) recurrent neural network (rnn) for discharge level prediction and forecast in cimandiri river, indonesia," in *IOP Conference Series: Earth and Environmental Science*, vol. 299, no. 1. IOP Publishing, 2019, p. 012037.

[83] X. Ding, Y. Zhang, T. Liu, and J. Duan, "Deep learning for event-driven stock prediction," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, p. 2327–2333.

[84] D. M. Nelson, A. C. Pereira, and R. A. de Oliveira, "Stock market's price movement prediction with lstm neural networks," in *2017 International joint conference on neural networks (IJCNN)*. IEEE, 2017, pp. 1419–1426.

[85] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in canada using lstm networks," *Chaos, Solitons & Fractals*, vol. 135, p. 109864, 2020.

[86] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, ser. Lecture Notes in Mathematics, 1981, pp. 366–381.

[87] C. Frazier and K. Kockelman, "Chaos theory and transportation systems: Instructive example," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 20, pp. 9–17, 2004.

[88] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[89] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[90] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 6, no. 2, pp. 107–116, 1998.

[91] A. Rawal and R. Miikkulainen, "Evolving deep lstm-based memory networks using an information maximization objective," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 2016, pp. 501–508.

[92] B. Bakker, "Reinforcement learning with long short-term memory," in *Advances in neural information processing systems*, 2002, pp. 1475–1482.

[93] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, pp. 2673 – 2681, 12 1997.

[94] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural networks : the official journal of the International Neural Network Society*, vol. 18, pp. 602–10, 07 2005.

[95] Y. Fan, Y. Qian, F.-L. Xie, and F. K. Soong, "Tts synthesis with bidirectional lstm based recurrent neural networks," in *INTERSPEECH*, 2014.

[96] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 2013, pp. 273–278.

[97] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[98] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, 1990, pp. 396–404.

[99] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[100] Hecht-Nielsen, "Theory of the backpropagation neural network," in *International 1989 Joint Conference on Neural Networks*, vol. 1, 1989, pp. 593–605.

[101] M. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[102] E. Lorenz, "Deterministic non-periodic flows," *Journal of Atmospheric Science*, vol. 20, pp. 267 – 285, 1963.

[103] M. Hénon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics*, vol. 50, no. 1, pp. 69–77, 1976.

[104] O. Rössler, "An equation for continuous chaos," *Physics Letters A*, vol. 57, no. 5, pp. 397 – 398, 1976.

[105] S. S., "Solar cycle forecasting: A nonlinear dynamics approach," *Astronomy and Astrophysics*, vol. 377, pp. 312–320, 2001.

[106] A. Weigend and N. Gershenfeld, "Time series prediction: Forecasting the future and understanding the past," in *Proceedings of a NATO Advanced Research Workshop on Comparative Time Series Analysis*, Santa Fe, New Mexico, 1994.

[107] "NASDAQ Exchange Daily: 1970-2010 Open, Close, High, Low and Volume," accessed: 02-02-2015. [Online]. Available: http://www.nasdaq.com/symbol/aciw/stock-chart

[108] R. Chandra, K. Jain, R. V. Deo, and S. Cripps, "Langevin-gradient parallel tempering for bayesian neural learning," *Neurocomputing*, 2019.

[109] R. Chandra, "Competition and collaboration in cooperative coevolution of Elman recurrent neural networks for time-series prediction," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 26, pp. 3123–3136, 2015.

[110] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction," *Neurocomputing*, vol. 243, pp. 21–34, 2017.

[111] X. Wu and Z. Song, "Multi-step prediction of chaotic time-series with intermittent failures based on the generalized nonlinear filtering methods," *Applied Mathematics and Computation*, vol. 219, no. 16, pp. 8584 – 8594, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0096300313002099

[112] Y. Zhou, S. Guo, and F.-J. Chang, "Explore an evolutionary recurrent anfis for modelling multi-step-ahead flood forecasts," *Journal of Hydrology*, vol. 570, pp. 343 – 355, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022169419300137