Learning Chaotic Dynamics using Tensor Recurrent Neural Networks

Rose Yu [1*] Stephan Zheng [2*] Yan Liu [1]

Abstract

We present Tensor-RNN, a novel RNN architecture for multivariate forecasting in chaotic dynamical systems. Our proposed architecture captures highly nonlinear dynamic behavior by using high-order Markov states and transition functions. Furthermore, we decompose the high-dimensional structure of the model using tensor-train networks to reduce the number of parameters while preserving the model performance. We demonstrate significant learning speed improvements over state-of-the-art RNN architectures in learning speed and predictive accuracy on a range of simulation data of non-linear dynamical systems, as well on real-world climate and traffic data. Moreover, we show that Tensor-RNN shows improved long-term forecasting robustness over unstructured baselines.

1. Introduction

One of the central questions in science is forecasting: given the past, what is the optimal future prediction? In many domains with complex spatiotemporal structure and nonlinear dynamics, such as weather systems and transportation networks, forecasting poses a significant unsolved structured prediction challenge. The key issue is how to accurately capture the non-linear dynamics and higher-order correlations of the data-generating process, which can make systems highly *sensitive to initial conditions* (Baker & Gollub, 1996). Examples abound in science and engineering, from neural activity, turbulence, climate to traffic systems. In chaotic systems, small perturbations in state can grow exponentially in time, which limits the *long-term* forecasting power of state-of-the-art approaches.

A classic example is the climate system: one can *derive* the system dynamics from physical principles (e.g. the Navier-Stokes fluid dynamics). However, simulations are fundamentally limited by finite-precision machines and perfectly measuring the initial conditions is impossible or requires prohibitively large amounts of data.

Common approaches to forecasting involve using spectral analysis, discrete time models, functional interpolation and explicitly learning the system dynamics (system identification) (Aguirre & Letellier, 2009). Other methods have used hierarchical models that learn both short-term and long-term behavior of the dynamics (Zheng et al., 2016) from data. However, it is often infeasible to obtain good analytical dynamics models or stable predictions.

In this work we therefore take a *data-driven approach to stable long-term forecasting in dynamical systems*. Our problem can be stated as follows: given a dataset of raw sequential data that was generated by a nonlinear dynamical system, how can we *efficiently* learn an RNN architecture that can *reliably* forecast over a future horizon of T time-steps? Here, we would like T to be as large as possible.

We propose and validate a novel recurrent neural network architecture, called Tensor-RNN, that provides a black-box model to perform forecasting and learns the system dynamics automatically. Previous RNN models have achieved state-of-the-art performance in sequence generation, machine translation and speech recognition (LeCun et al., 2015). However, these models do not generalize well to the domain of forecasting in chaotic systems.

The key features of Tensor-RNN are: (1) it considers a longer history of previous hidden states; (2) it directly models the high order state interactions with multiplicative memory units; (3) it employs a tensor network method, a structured low-rank tensor decomposition to reduce the number of parameters in the model. This provides an effective approximation which largely preserves the correlation structure of the full-rank model.

Our contributions can be summarized as follows:

- We describe a novel RNN architecture, Tensor-RNN, that encodes higher-order non-Markovian dynamics and present a tensor network decomposition that makes learning tractable and fast.

[*]Equal contribution [1]University of Southern California, Los Angeles, CA 90089 USA [2]California Institute of Technology, Pasadena CA 91125 USA. Correspondence to: Stephan Zheng <stephan@caltech.edu>, Rose Yu <qiyu@usc.edu>.

Proceedings of the ICML 17 Workshop on Deep Structured Prediction, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

- We validate our model on 3 chaotic systems: the Lorenz attractor, real-world climate and traffic data.
- We experimentally show that the proposed model can forecast chaotic dynamics more accurately and faster than common RNN models, both for short-term and long-term forecast horizons.

Our work is related to classic work in time series forecasting, which has studied auto-regressive models, such as the ARMA or ARIMA model (Box et al., 2015). These model a process $x(t)$ linearly, which does not capture nonlinear, chaotic systems. Using neural networks to capture the chaotic dynamics has a long history (Aihara et al., 1990; Jaeger & Haas, 2004) and have been applied to weather forecasting, traffic prediction and other domains (Schmidhuber, 2015). From a modeling perspective, (Giles et al., 1989) considers a *high-order RNN* to simulate a deterministic finite state machine and recognize regular grammars. (Sutskever et al., 2011) propose *multiplicative RNN* that allow each hidden state to specify a different hidden-to-hidden weight matrix. However, these models only use the most recent state and are limited to additive memory units.

2. Tensor RNN

Our goal is to develop an efficient model for sequential multivariate *forecasting* problems: given a sequence of d-dimensional vectors $(x(t))_{t>0}$, predict $y(t) = x(t+1)$. An RNN recursively computes $y(t)$ from a hidden state $h(t)$:

$$h(t) = f(x(t), h(t-1); \theta), y(t) = g(h(t)) \qquad (1)$$

where f is a non-linear activation function and θ denote the layer parameters. An RNN therefore learns a model for the Markov process $(h(t))_{t>0}$, of order 1 (only the previous time-step is considered). A common choice is to model the recurrent layer as an activation function applied to a linear combination of $x(t)$ and $h(t)$:

$$\begin{aligned} h(t) &= f(W^{hx}x(t) + W^{hh}h(t-1) + b^h), \\ x(t+1) &= \tilde{f}(W^{xh}h(t) + b^x) \end{aligned} \qquad (2)$$

where $f, \tilde{f}$ are activation functions (e.g. sigmoid, $\tanh$) for the state transition function, W^{hx}, W^{xh} and W^{hh} are transition matrices and b^h, b^x are biases. Although existing RNNs are very expressive, in the majority of RNNs the state $h(t)$ is limited to only a linear combination of the previous state $h(t-1)$ and input $x(t)$.

However, many systems evolve under complex *continuous-time dynamics* that are given by *nonlinear* equations:

$$g\left(x(0:t), \frac{dx}{dt}(0:t), \frac{d^2x}{dt^2}(0:t), \ldots; \phi\right) = 0, \qquad (3)$$

where g can be an arbitrary (smooth) function in the (historical) states $x(0:t)$ and their derivatives. Due to nonlinearities in g, many systems exhibit *chaotic* behavior: two initial states $x_1(0), x_2(0)$ that are arbitrarily close can drift exponentially far apart under evolution of g. Learning an accurate discrete-time forecasting model $y(t)$ thus implies learning an optimal approximation to g.

2.1. Tensor-RNN

We propose a high-order model, called Tensor-RNN, that is suited for learning nonlinear dynamical systems and can be viewed as a higher-order generalization of RNNs. Our model has two goals: explicitly modeling k-order Markov processes with k steps of temporal memory and polynomial interactions between the hidden states $h(\cdot)$ and $x(t)$.

First, we consider longer "history": we keep K order historic states: $h_1, \cdots, h_K$, making the Tensor-RNN high-order Markov. In its most general form, the high-order Markov design leads to the following architecture:

$$h(t) = f(x(t), h(t-1), \cdots, h(t-K)) \qquad (4)$$

where f is an activation function. In principle, previous work has shown that with a large enough hidden state size, such RNN structures are capable of approximating any temporal function.

Second, to learn nonlinear dynamics g efficiently, we also add higher-order interactionspasade to approximate the state transition function. We can define the K-order hidden state as:

$$s(t-1)^T = \begin{bmatrix} 1 & h(t-1) & \ldots & h(t-K) \end{bmatrix}$$

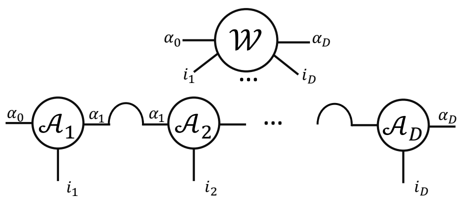
The bias units 1 allow us to model all possible polynomial interactions up to order D in a compact form. We can now construct a higher-order transition *tensor* by modeling a degree D polynomial interaction between the hidden states using a $D+1$ dimensional tensor $\mathcal{W}$:

$$h(t+1)_\alpha = f\left(\sum_{i_1,\cdots,i_D} \mathcal{W}_{\alpha i_1 \cdots i_D} \underbrace{s_{i_1}(t) \otimes \cdots \otimes s_{i_D}}_{D}(t)\right) \qquad (5)$$

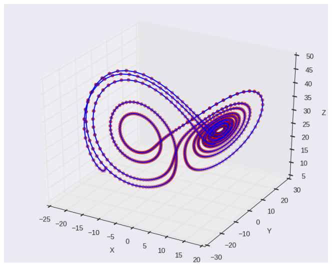
where the indices $i.$ index the memory states and D is the degree of the polynomial.

2.2. Tensor Network Method

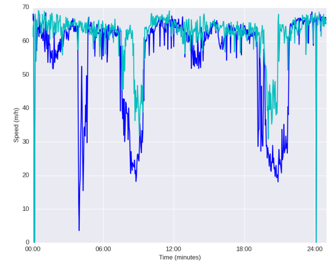
Unfortunately, the number of parameters in $\mathcal{W}$ grows exponentially as $O(K^D)$, which makes the general $K-D$-order model prohibitively large to train. To overcome this difficulty, we approximate $\mathcal{W}$ with *tensor networks*. Such networks encode a structural decomposition of tensors into



(a) Tensor-train decomposition: nodes represent tensor variables and straight edges represent indices. Arcs represent tensor contractions.



(b) Lorenz Attractor



(c) Traffic: 2 sensors

Figure 1. (a) Tensor-train decomposition (left). 2 examples of chaotic systems: (b) Lorenz attractor (middle) and (c) traffic data (right).

low-dimensional components and have been shown to provide the most general approximation to smooth tensors (Orús, 2014). The most commonly used tensor networks are *linear tensor networks* (LTN), also known as *tensor-trains* in numerical analysis or *matrix-product states* in quantum physics (Oseledets, 2011).

A tensor train model decomposes a D-dimensional tensor $\mathcal{W}$ into a network of sparsely connected low-dimensional tensors $\{\mathcal{A}^d \in \mathbb{R}^{r_{d-1}\times n_d \times r_d}\}$ as:

$$\mathcal{W}_{\alpha_0 i_1 \cdots i_D \alpha_D} = \sum_{\alpha_1 \cdots \alpha_{D-1}} \mathcal{A}^1_{\alpha_0 i_1 \alpha_1} \mathcal{A}^2_{\alpha_1 i_2 \alpha_2} \cdots \mathcal{A}^D_{\alpha_{D-1} i_D \alpha_D}$$

as depicted in Figure (1a). When $r_0 = r_{D+1} = 1$ the $\{r_d\}$ is called the tensor train rank. With tensor networks, we can reduce the number of parameters from $(HK+1)^D$ to $(HK+1)R^2D$, with H as the hidden layer size and $R = \max_d r_d$ as the upper bound on the tensor network rank. Thus, a major benefit of LTNs is that *they do not suffer from the curse of dimensionality.* This is in sharp contrast with many classical tensor decompositions, such as the Tucker decomposition ($O(\text{rank}^D)$).

3. Experiments

We validated the accuracy, efficiency and robustness of Tensor-RNNs on 1 synthetic and 2 real-world datasets.

Lorenz Attractor. The Lorenz system describes a two-dimensional flow of fluids (see Figure 1b):

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y-x) \quad \frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y \quad \frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z$$

This system has chaotic solutions (for certain parameter values) that revolve around the so-called Lorenz attractor. We simulated $100,000$ time-steps using $\sigma = 10$ $\rho = 28$, $\beta = 2.667$, with 10 random initial points (seeds).

USHCN-CA. The U.S. Historical Climatology Network (USHCN) daily (`http://cdiac.ornl.gov/ftp/ushcn_daily/`) contains daily measurements for 5 climate variables for more than 100 years. The five climate variables correspond to maximal daily temperature, minimal daily temperature, precipitation, snow fall and snow depth. The records were collected across more than 1 200 locations and span over 45 384 time stamps.

LA Highway Traffic. We also used traffic data collected from the highways and arterial streets of Los Angeles County (covering 5400 miles cumulatively) from May 19, 2012 to June 30, 2012. Due to many missing values in the raw data, we filtered out sensors with more than 20% missing values (see Figure 1c). In total, after processing, the dataset covers 36 000 time-steps.

Experimental Setup. To justify that Tensor-RNN effectively learns chaotic dynamics, we evaluated it on:

- Short-term forecasting: $\forall t$: given x_t, predict x_{t+1}.
- Long-term forecasting: given a *burn-in* sequence $x_0, \ldots, x_{t_0}$, predict $x_{t_0+1}, \ldots, x_{t+n}$. For each t, the model uses as input its previous prediction $\hat{x}_t$.

Our experiments use $t_0 = 5, n = 20$. During training, we randomly select subsequences and do back-propagation-through-time over 20 time-steps. During testing, for long-term forecasting, we move through the test-set in order, and reset hidden states for each prediction window.

Baselines. We compared Tensor-RNN against 3 baseline recurrent neural network models: RNN, LSTM and Higher-order RNNs (HORNN) (Soltani & Jiang, 2016).

Training. We trained all models using gradient descent on the length-T sequence regression loss

$$L(y,\hat{y}) = \sum_{t=1}^{T} ||\hat{y}_t - y_t||_2^2, \tag{6}$$

where $y_t, \hat{y}_t$ are the ground truth and model prediction respectively. We trained all models using the ADAM optimizer and performed a hyper-parameter search over the learning-rate $(10^{-1} \ldots 10^{-5})$, hidden state size $(32, 64, 128, 256)$ and tensor-train rank $(1 \ldots 128)$. For

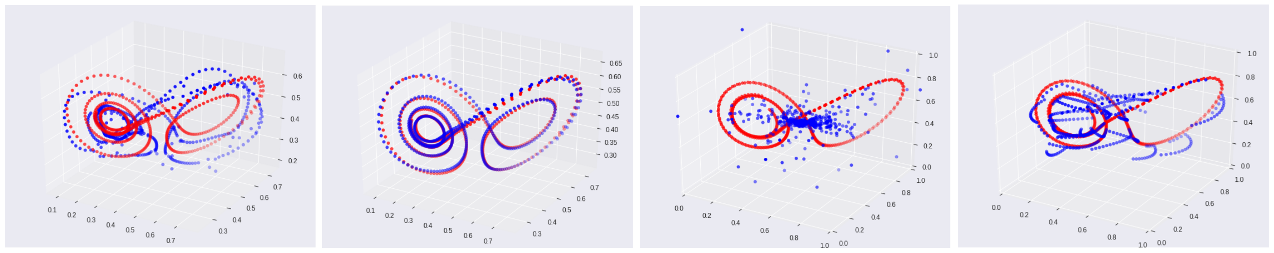


Figure 2. Short-term (left 2) and long-term (right 2) predictions for LSTM (left, blue) and Tensor-RNN (right, blue) versus the ground truth (red). In short-term forecasting, Tensor-RNN is closer than the LSTM to the true dynamics. In long-term forecasting, Tensor-RNN shows more consistent, but imperfect, predictions, whereas the LSTM is highly unstable and gives noisy predictions.

Dataset	Lorenz	Climate	Traffic
RNN	0.145	0.258	0.113
LSTM	0.029	0.220	**0.086**
HORNN	0.100	0.239	0.090
TRNN	**0.009**	**0.177**	0.089

Table 1. Short-term forecasting test-set accuracy (RMSE)

Dataset	Lorenz	Climate	Traffic
RNN	0.161	0.281	0.187
LSTM	0.228	0.301	0.141
HORNN	0.159	0.353	0.224
TRNN	**0.083**	**0.230**	**0.139**

Table 2. Long-term forecasting ($n = 20$) test accuracy (RMSE)

all datasets, we used a 80%-10%-10% train-validation-test split.

Results: short-term forecasting. Table 1 reports short-term forecasting accuracy. The Tensor-RNN achieves significant improvements over baselines for the Lorenz and climate data, while it is competitive for traffic data. The latter result might be due to the high number of missing values, which might limit the performance for all models.

Results: long-term forecasting. We now investigate how robust Tensor-RNNs are to error propagation in long-term forecasting in chaotic systems. We report the prediction error (in RMSE) on the test-set for long-term forecasting with 5 burn-in steps and a large ($n = 20$ steps) rollout, as shown in table 2. Our main result is that Tensor-RNN notably outperforms all baselines on all datasets in this setting, including the matrix HORNN model. In particular, we note that RNNs sometimes outperform LSTMs and HORNNs on long-term forecasting. This indicates that more complicated models are not automatically more robust to error propagation in chaotic systems.

Speed-up Performance Trade-off. We now investigate potential trade-offs between accuracy and computation.

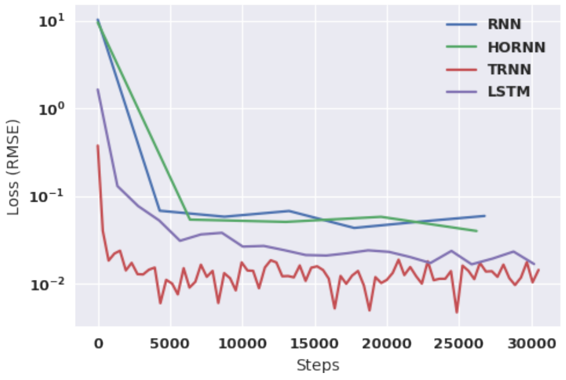


Figure 3. Training speed performance as the log-loss (log-RMSE) versus training step for the models with the best long-term forecasting accuracy from Table 2. The hidden state is 256-dimensional (TRNN) and 32-dimensional (others). We observe that TRNN converges faster than baselines.

Figure 3 displays the training loss with respect to the number of steps, for the best performing mdoels on long-term forecasting. We see that Tensor-RNN converges significantly faster than other models, both in terms of number of training steps and achieved training-loss. This suggests that Tensor-RNN has a more efficient representation of the nonlinear dynamics, and can learn much faster as a result.

Model Inspection. To get intuition for the learned models, we visualize the best performing TRNN and LSTM in Figure 2. In short-term forecasting, we see that the TRNN corresponds much better with ground truth. In long-term forecasting, TRNN is more stable, whereas the LSTM produces highly noisy predictions. However, TRNN does not follow the ground truth perfectly and still produces deviating tracks, showing there is still room for improvements.

References

Aguirre, Luis A and Letellier, Christophe. Modeling nonlinear dynamics and chaos: a review. *Mathematical Problems in Engineering*, 2009, 2009.

Aihara, Kazuyuki, Takabe, T, and Toyoda, M. Chaotic neural networks. *Physics letters A*, 144(6-7):333–340, 1990.

Baker, Gregory L and Gollub, Jerry P. *Chaotic dynamics: an introduction*. Cambridge University Press, 1996.

Box, George EP, Jenkins, Gwilym M, Reinsel, Gregory C, and Ljung, Greta M. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Giles, C Lee, Sun, Guo-Zheng, Chen, Hsing-Hen, Lee, Yee-Chun, and Chen, Dong. Higher order recurrent networks and grammatical inference. In *NIPS*, pp. 380–387, 1989.

Jaeger, Herbert and Haas, Harald. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304(5667):78–80, 2004.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Orús, Román. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

Oseledets, Ivan V. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Soltani, Rohollah and Jiang, Hui. Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*, 2016.

Sutskever, Ilya, Martens, James, and Hinton, Geoffrey E. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 1017–1024, 2011.

Zheng, Stephan, Yue, Yisong, and Hobbs, Jennifer. Generating long-term trajectories using deep hierarchical networks. In *Advances in Neural Information Processing Systems*, pp. 1543–1551, 2016.