On the Memory Mechanism of Tensor-Power Recurrent Models

Hejia Qiu[⋆]
University of Nottingham Ningbo China
University of Nottingham UK

Chao Li[⋆,§]
RIKEN-AIP
Tokyo, Japan

Ying Weng[§]
University of Nottingham Ningbo China
University of Nottingham UK

Zhun Sun
BIGO PTE. LTD.
Singapore

Xingyu He
University of Nottingham Ningbo China
University of Nottingham UK

Qibin Zhao[§]
RIKEN-AIP
Tokyo, Japan

Abstract

Tensor-power (TP) recurrent model is a family of non-linear dynamical systems, of which the recurrence relation consists of a p-fold (*a.k.a.*, degree-p) tensor product. Despite such the model frequently appears in the advanced recurrent neural networks (RNNs), to this date there is limited study on its memory property, a critical characteristic in sequence tasks. In this work, we conduct a thorough investigation of the memory mechanism of TP recurrent models. Theoretically, we prove that a large degree p is an essential condition to achieve the long memory effect, yet it would lead to unstable dynamical behaviors. Empirically, we tackle this issue by extending the degree p from discrete to a differentiable domain, such that it is efficiently learnable from a variety of datasets. Taken together, the new model is expected to benefit from the long memory effect in a stable manner. We experimentally show that the proposed model achieves competitive performance compared to various advanced RNNs in both the single-cell and seq2seq architectures.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

1 Introduction

Recurrent neural networks (RNNs) have been popularly used for tasks arising in domains such as time series analysis and natural language processing. They are powerful because the recurrent dynamics of the hidden states allow the models to remember the past information. In the standard RNNs, the transition function of the hidden states obeys a linear transform[1] following an element-wise activation function, and the vanilla form of RNN has an inherent difficulty to learn the long-range dependence of the data (Bengio et al., 1994). As alternatives, a family of variants of RNNs are proposed, where the linear form of the transition function is extended to higher-degree polynomials (Sutskever et al., 2011; Wu et al., 2016; Schlag and Schmidhuber, 2018). Rich results by numerical experiments demonstrate the RNNs equipped with polynomial transitions are more expressive than their vanilla counterparts (Yu et al., 2017a; Su et al., 2020).

However, those polynomial-induced variants fail to become mainstream models in sequence tasks. The cause is mainly twofold: theoretically, it remains unclear why the higher-degree models outperform their linear counterparts; empirically, an "over-large" degree would lead to the model being unstable in both the training and inference phases. Moreover, the model size would explode when the degree increases. Al-

[⋆]Equal Contribution.
[§]Correspondence to: Chao Li<chao.li@riken.jp>; Ying Weng<yingweng@gmail.com>; Qibin Zhao<qibin.zhao@riken.jp>
[1]It is an affine transform more precisely.

though for the latter it is alleviated by tensor decomposition (TD) (Ye et al., 2018; Pan et al., 2019), the selection of the degree parameters is still achieved with the inefficient exhausitive search (Yu et al., 2017b; Hou et al., 2019). As a consequence, the state of affairs raises important unresolved questions. *How does the model degree determine the memory property and dynamical behaviors? Is it possible to efficiently select the optimal degree in practice?*

The goal of this work is to shed light on both two questions through a theoretical and empirical investigation. We focus on a unified expression of the aforementioned variants of RNNs named *tensor-power (TP) recurrent models*, in which the degree-p polynomial is reformulated as a p-fold tensor product of the hidden states multiplied by a weight tensor.

Theoretically, we prove the long memory property of the model requires a large value of the degree p (see *Thm. 2*), while in this case the unstable behaviors of the model are inherently inevitable (see *Thm. 3*). Empirically, we propose a degree-learnable method to seek a "saddle-point" of balancing the long memory effect and stability of the model. In particular, we extend the feasible range of the model degree from discrete to a continuous domain using tensor decomposition (TD). The differentiable essence of the degree allows us to efficiently learn it from datasets by stochastic gradient descent (SGD) and its variants. Endowing with the additional freedom on the degree parameter, we expect that the new model can benefit from the long memory effect yet keep the stability reachable. Extensive numerical results demonstrate the superior performance of the proposed model in time series forecasting tasks with both shallow and relatively deep architectures.

1.1 Related works

Polynomial recurrent neural networks (RNNs). The earliest study of recurrent neural networks (RNNs) with polynomial transition dates back to literature (Lee et al., 1986; Giles et al., 1990; Pollack, 1991). Later, similar architectures are used in various machine learning tasks (Sutskever et al., 2011; Wu et al., 2016). More recently, the connection between higher-degree polynomial RNNs and the weighted automata (WA) is uncovered through spectral learning (Rabusseau et al., 2019), and on the practical side the polynomials are also applied to constructing the correlation among time-steps in multimodal tasks (Hou et al., 2019; Li et al., 2020a). Compared to those works, we are the first to theoretically investigate the memory property of the model, and the proposed method has the capability of learning the optimal degree parameters from datasets while the parameters are assumed to be fixed in previous works.

Tensor methods in RNNs. There are three lines of studying the tensor methods in RNNs. One line of work is the extension of the polynomial RNNs to higher-degree. In the existing works, the most closed one to ours is HOTRNN (Yu et al., 2017a,b), and the similar idea is extended in recent works on convolutional LSTM (Huang et al., 2020a; Su et al., 2020). Compared to those methods, we mainly focus on the issue how the "tensor order" (*i.e.*, the "degree" in this work) influences the performance of RNNs and how to obtain the optimal tensor order in a more efficient manner rather than by the exhaustive search. The second line is to apply tensor decomposition (TD) to compressing RNNs (Tjandra et al., 2017; Yang et al., 2017; Ye et al., 2018; Mehta et al., 2019; Pan et al., 2019; Wang et al., 2020). Unlike ours, they still model the transition function as a "linear" transform, yet reshaping the weights into high-order tensors. The third line of work is to use the tensor network modeling to analyze the expressive power of RNNs (Khrulkov et al., 2017, 2019). In contrast, our work pays more effort on the memory property and focuses on the impact by the "tensor-order" rather than different decomposition models.

RNNs for long memory. Many works have been proposed on the long memory property of RNNs, for instance, the works by Le et al. (2019); Trinh et al. (2018); Voelker et al. (2019); Wang and Niepert (2019); Lechner and Hasani (2020) to name a few. In this work, the theoretical aspect is partially inspired by recent studies (Greaves-Tunnell and Harchaoui, 2019; Zhao et al., 2020) from the stochastic perspective and the works by Pascanu et al. (2013); Miller and Hardt (2018) from the stability perspective. Compared to those works, we focus on more specific non-linear dynamics of the transition and analysing how the model degree influences the memory property.

The study on multivariate factional polynomial fitting originates by Royston and Altman (1994) in statistics, and is also discussed in the deep learning community (Gulcehre et al., 2014; Sun et al., 2018). Our work on the degree-learnable approach is connected to those works, but focuses on the different issue. Most recently, we also extend the similar idea into a more general form named fractional tensor network (Li et al., 2020b), which focuses on different tensor structures and feedforward learning models.

The proofs and additional experimental details can be found in the supplementary material. Our code is available at `https://github.com/minogame/AISTATS_2021`.

2 Preliminaries

In this section, we first present basic notations of tensor algebra. After that, we recall the definition of the long memory in statistics in terms of *recurrent network process (RNP)* (Zhao et al., 2020). Last, we introduce the *tensor-power (TP) recurrent model.*

Notation. We define a *tensor* as a multi-dimensional array of real numbers (Kolda and Bader, 2009). To distinguish from the notion of "order" in time-series analysis, we use the term *degree of a tensor* to denote the number of indices. In the rest of the paper, we use italic letters to denote scalars, *e.g.*, $n, N \in \mathbb{R}$, and use boldface letters to denote vectors and matrices, *e.g.*, $\mathbf{h}, \mathbf{y} \in \mathbb{R}^n$ and $\mathbf{W} \in \mathbb{R}^{n\times n}$. For higher-degree tensors, we denote them by calligraphic letters, *e.g.*, $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_d}$. Sometimes we also use the calligraphic letters to represent vectors or matrices without ambiguity. The *symmetry* of a tensor is defined as the invariance under arbitrarily reshuffling a sub-collection of the tensor's indices, and the *index-shift* operation of a tensor is defined as rotating permuting the indices of a tensor in counterclockwise order. Examples about the "symmetry" and "index-shifting" are given in the supplementary material. For any integer k, we use $[k]$ to denote the set of integers from 1 to k. For any real number x, we use $|x|$ and $sgn(x)$ to denote its absolute value and sign, respectively. If $\mathbf{x} \in \mathbb{R}^{I_x}$ and $\mathbf{y} \in \mathbb{R}^{I_y}$, we use $\langle \mathbf{x}, \mathbf{y}\rangle \in \mathbb{R}$ and $\mathbf{x}\otimes\mathbf{y} \in \mathbb{R}^{I_x\times I_y}$ to respectively denote the inner and tensor product between vectors, and their straightforward extension to matrices and tensors. Moreover, we use $\mathbf{x}^{\otimes p} \in \mathbb{R}^{I_x^p}$ to denote the degree-p *tensor power* (TP), which represents the p-fold tensor product of $\mathbf{x}$ with itself. For a degree-q tensor $\mathcal{G} \in \mathbb{R}^{n^q}$ and a vector $\mathbf{x} \in \mathbb{R}^n$, we use $\mathcal{G} \times_i \mathbf{x}$, $i \in [q]$ to denote *tensor-vector* product along the ith index (Cichocki et al., 2007). The spectral norm of a tensor $\mathcal{G} \in \mathbb{R}^{I_1\times\cdots\times I_q}$ is denoted by $\|\mathcal{G}\|_2$, which is defined as

$$\begin{aligned}\|\mathcal{G}\|_2 = \sup_{\mathbf{u}_i\in\mathbb{R}^{I_i}, i\in[q]} &\|\mathcal{G}\times_1 \mathbf{u}_1 \times_2 \cdots \times_q \mathbf{u}_q\|_2, \\ s.t.\ &\|\mathbf{u}_i\|_2 \le 1, \forall i\end{aligned} \quad , \tag{1}$$

where $\|\mathbf{u}_i\|_2$ denotes the Euclidean norm of $\mathbf{u}_i$.

2.1 Recurrent network process (RNP)

The notion of *recurrent network process (RNP)* is defined to analyze the memory mechanism of recurrent neural networks (RNNs). Specifically, assume a general RNN with input $\{\mathbf{x}^{(t)}\}$, output $\{\mathbf{y}^{(t)}\}$ and the hidden states $\{\mathbf{h}^{(t)}\}$ where $\mathbf{y}^{(t)} \in \mathbb{R}^l$ and $\mathbf{h}^{(t)} \in \mathbb{R}^m$, then its RNP is defined as a homogeneous Markov chain with a transition function M:

$$\begin{pmatrix}\mathbf{y}^{(t)}\\ \mathbf{h}^{(t)}\end{pmatrix} = M\begin{pmatrix}\mathbf{y}^{(t-1)}\\ \mathbf{h}^{(t-1)}\end{pmatrix} + \begin{pmatrix}\varepsilon^{(t)}\\ 0\end{pmatrix}, \tag{2}$$

where $\{\varepsilon^{(t)}\}$ represents a sequence of independent and identically distributed (*i.i.d.*) random vectors. Eq. (2) can be used to describe various RNNs like the vanilla RNN and LSTM (Hochreiter and Schmidhuber, 1997) without exogenous inputs. The term *short or long memory* of the stochastic process generated by Eq. (2) is roughly defined as below. More precise definition is given in works by Beran et al. (2016); Greaves-Tunnell and Harchaoui (2019); Zhao et al. (2020). Roughly speaking,

Definition 1 (memory of RNPs, roughly)**.** *The process* $\left\{\mathbf{s}^{(t)} := \left(\mathbf{y}^{(t),\top}, \mathbf{h}^{(t),\top}\right)^\top \in \mathbb{R}^{l+m}\right\}$ *generated by Eq.* (2) *has short memory if its autocovariance function is summable. Otherwise, the process has long memory.*

The relationship between the memory property and the operator norm of the transition function M is revealed as below:

Assumption 1. (1) *The joint density function of* $\varepsilon(t)$ *is continuous and positive everywhere;* (2) *for some* $\kappa \ge 2$, $\mathbb{E}\|\varepsilon^{(t)}\|_2^\kappa < \infty$.

Theorem 1 (by Zhao et al. (2020))**.** *Under Assumption 1, if there exist real numbers* $0 < a < 1$ *and* b *such that* $\|M(\mathbf{x})\|_2 \le a\|\mathbf{x}\|_2 + b$, *then the RNP* (2) *has short memory.*

Def. 1 and Thm. 1 will be used in Sec. 3.1 to model and analyze the memory property of the TP recurrent model. We use them to bridge the gap from the memory property to the tensor power operation.

2.2 Tensor-power (TP) recurrent model

A *degree-p* tensor-power (TP) recurrent model, with its input $\mathbf{x}^{(t)} \in \mathbb{R}^l$ and hidden state $\mathbf{h}^{(t)} \in \mathbb{R}^m$ at the time-step t, is defined as[2]

$$\begin{aligned}\mathbf{h}^{(t)} &= \mathcal{G}\times_1 \begin{pmatrix}\mathbf{x}^{(t)}\\ \mathbf{h}^{(t-1)}\end{pmatrix} \times_2 \cdots \times_p \begin{pmatrix}\mathbf{x}^{(t)}\\ \mathbf{h}^{(t-1)}\end{pmatrix} \\ &= \mathcal{G}\cdot \begin{pmatrix}\mathbf{x}^{(t)}\\ \mathbf{h}^{(t-1)}\end{pmatrix}^{\otimes p}\end{aligned} \quad , \tag{3}$$

where $\mathcal{G} \in \mathbb{R}^{n^p\times m}$, $n = l + m$ denotes a degree-$(p+1)$ tensor, which is partially symmetric involving the first p indices. The symbol $\cdot$ is used as a simplified representation of the sequential tensor-vector multiplication in Eq. (3). In this case, the current state $\mathbf{h}^{(t)}$ is

[2]Here the bias term is omitted for brevity.

updated by a system of degree-p homogeneous polynomials of the concatenation of the input and the historic values of the state. Its more general extensions that include *inhomogeneous* terms can be trivially obtained by padding additional constants along $\mathbf{x}^{(t)}$. Although activation functions such as "tanh" or "ReLU" are popularly used in the model, in this work we consider the case that the activation functions are ignored in order to simplify the theoretical results. We claim that in practical tasks *the non-trivial tensor power operation has the capability of providing sufficient non-linearity to the model.*

Examples. The model (3) would degenerate into the most common linear form when $p = 1$. If setting $p = 2$, Eq. (3) can fully describe the dynamics of models in the works by Giles et al. (1990); Sutskever et al. (2011); Wu et al. (2016), and has a strong connection to weighted finite automata (WTAs) (Rabusseau et al., 2019). For the higher-degree case, Eq. (3) is strongly related to various tensor recurrent models (Yu et al., 2017a; Huang et al., 2020a; Su et al., 2020). It is worth noting that in those methods the non-Markovian model is also applied by considering longer historic hidden states into the recurrent process. Our empirical results show that additional historic states are not necessary for the long memory property.

3 Memory of TP recurrent model

In this section, we theoretically investigate the memory mechanism of the TP recurrent model (3) from both stochastic and stability perspectives, and focus on proving how the degree determines the memory property of the model. Note that, in the theoretical analysis, we do not apply tensor decomposition (TD) to the weights, which is different from the works by Yu et al. (2017a); Su et al. (2020). TD will be exploited in Sec. 4 to develop a degree-learnable model.

3.1 Perspective from RNPs

Assume that the output of the TP recurrent model is also obtained by the similar degree-p tensor power form to Eq. (3) and there is no exogenous input, *i.e.*, $\mathbf{x}^{(t)} = \mathbf{y}^{(t-1)}$, then Eq. (2) can be specified as

$$\begin{aligned}\mathbf{s}^{(t)} &= \mathcal{M} \times_1 \mathbf{s}^{(t-1)} \times_2 \cdots \times_p \mathbf{s}^{(t-1)} + \mathbf{e}^{(t)} \\ &= \mathcal{M} \cdot \left(\mathbf{s}^{(t-1)}\right)^{\otimes p} + \mathbf{e}^{(t)}, \quad \forall t\end{aligned}, \qquad (4)$$

where $\mathbf{s}^{(t)} := \left(\mathbf{y}^{(t),\top}, \mathbf{h}^{(t),\top}\right)^\top \in \mathbb{R}^{l+m}$, $n = l + m$, $\mathcal{M} \in \mathbb{R}^{n^{(p+1)}}$ denotes a degree-$(p+1)$ tensor with the symmetric structure among the first p indices, and $\mathbf{e}^{(t)} := \left(\varepsilon^{(t),\top}, 0\right)^\top$. Below, we refer to the stochastic process generated by the model (4) as *tensor-power recurrent network process (TP-RNP)*. Below we investigate how the degree p influences the memory of TP-RNP under Def. 1. First, we show that TP-RNP has short memory if the spectral norm of $\mathcal{M}$ is upper-bounded.

Lemma 1. *Under Assumption 1, the tensor-power recurrent network process (TP-RNP)* (4) *has short memory under Def. 1 if the spectral norm of the tensor* $\mathcal{M}$ *obeys* $\|\mathcal{M}\|_2 < 1$.

The claim is a natural corollary from Thm. 1. It is implied from Lemma 1 that TP-RNP has the short memory property if the spectral norm of the tensor $\mathcal{M}$ is sufficiently small. As pointed out by Zhao et al. (2020), the condition is often satisfied in practice when $p = 1$ (*i.e.*, the linear RNN), because the entries of $\mathcal{M}$ are practically bounded away from one. However, *can we expect that the short memory of TP-RNP* (4) *would be kept if increasing the degree parameter* $p > 1$*?* To study this point, we specify that the entries of $\mathcal{M}$ obey the sub-Gaussian distribution, *i.e.*,

Assumption 2 (sub-Gaussian and decoupling). *The tensor* $\mathcal{M}$ *is obtained by the average over all* p *first-p-indices-shifted variants of a tensor* $\mathcal{A} \in \mathbb{R}^{n^{(p+1)}}$*, of which each entries* $\mathcal{A}_{i_1,i_2,\ldots,i_{p+1}}$ *is independent, zero-mean and satisfied* $\mathbb{E}\left(e^{t\mathcal{A}_{i_1,i_2,\ldots,i_{p+1}}}\right) \leq e^{\sigma^2 t^2/2}$.

It is easily known that the averaged tensor $\mathcal{M}$ is symmetric among the first p indices. Moreover, because the sub-Gaussian distribution generates samples concentrating around zero with few outliers, the assumption is empirically reasonable as aforementioned. Next, we theoretically reveal how the degree impacts the model's memory property. Assuming that our TP-RNP satisfies Assumption 1 and 2, we show below that the long memory requires a high model degree. Specifically,

Theorem 2 (Long memory requires a high model degree.). *Under Assumptions 1 and 2, with high probability, if TP-RNP* (4) *has the long memory under Def. 1, then the following inequality obeys:*

$$p \geq \frac{p_0}{2}\left(1 + \sqrt{1 + \frac{C_1}{n\sigma^2} - \frac{C_2}{n}}\right) - 1, \qquad (5)$$

where $p_0 = \log(3/2)$*, and* C_1*,* C_2 *denote two positive constants.*

The inequality (5) is obtained using Lemma 2 and the results on the non-asymptotic bound of tensor spectral norm given by Tomioka and Suzuki (2014). We can see that the degree p is controlled by both the dimension of the model and the distribution of $\mathcal{M}$. Fixed the dimension n, a smaller value of σ^2 leads to a larger value of the degree p for the long memory.

Thm. 2 implies that the degree p should be sufficiently large, otherwise the model only has short memory (see Fig. 1 in the *supplementary material* for the visualization of the bound in (5)). Intuitively, the model with a higher-degree would result in an unbounded non-linear transition function. In this case, the transition plays a role like an *"amplifier"*, such that at each time-step both the hidden states and input would be amplified by the degree p. As the result, it would become uneasy for the network to "forget" the information a long time ago, *i.e.*, the long memory effect.

3.2 Perspective from stability analysis

The long memory effect does not guarantee the recurrent model is trainable. It is therefore of importance to state clearly whether an unstable behavior would happen in the TP recurrent model. In particular, *how does the model degree affect the stability of the model?*

Recall the TP recurrent model (3). To answer the question, we define its stability following the definition given by Miller and Hardt (2018) for an arbitrary recurrent model. Specifically,

Definition 2 (stability of TP recurrent model). *The model* (3) *is stable if there exist some* $\lambda < 1$ *such that, for any states* $\mathbf{h}$, $\mathbf{h}'$, *and input* $\mathbf{x}$,

$$\left\| \mathcal{G} \cdot \left(\begin{pmatrix} \mathbf{x} \\ \mathbf{h} \end{pmatrix}^{\otimes p} - \begin{pmatrix} \mathbf{x} \\ \mathbf{h}' \end{pmatrix}^{\otimes p} \right) \right\|_2 \leq \lambda \|\mathbf{h} - \mathbf{h}'\|_2. \quad (6)$$

As shown in Def. 2, the TP recurrent model is stable if Eq. (3) is λ-Lipschitz with $\lambda < 1$. Hence, we next prove that *the TP recurrent model is not L_p-Lipschitz if* $p > 1$. Before the main claim, we first give a general form of the Jacobian of Eq. (3):

Lemma 2 (Jacobian of the model). *For any tensor* $\mathcal{G} \in \mathbb{R}^{n^p \times m}$ *of degree-*$(p+1)$, $p > 0$, *the Jacobian matrix* $\frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}}$ *with respect to Eq.* (3) *is equal to*

$$J\left(\mathbf{h}^{(i-1)}; \mathbf{x}^{(i)}\right) = \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}} = \sum_{k=1}^{p} \left(\mathcal{G} \cdot \begin{pmatrix} \mathbf{x}^{(i)} \\ \mathbf{h}^{(i-1)} \end{pmatrix}^{\otimes p / \{k\}} \right) \times_{p+1} \begin{pmatrix} \mathbf{0}_l \\ \mathbf{I}_m \end{pmatrix}, \quad (7)$$

where $\mathbf{0}_l \in \mathbb{R}^{l \times m}$ *denotes the matrix filled by zeros,* $\mathbf{I}_m \in \mathbb{R}^{m \times m}$ *is a identity matrix, and the operator* $(\cdot)^{\otimes p/\{k\}}$ *denotes the sequential "tensor-vector" product along the indices in the ordered set* $[p]/\{k\}$. *If* $\mathcal{G}$ *is symmetric among the first p indices, then Eq.* (7) *can be simplified as*

$$J_s(\mathbf{h}^{(i-1)}; \mathbf{x}^{(i)}) = p \left(\mathcal{G} \cdot \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix}^{\otimes (p-1)} \right) \times_{p+1} \begin{pmatrix} \mathbf{0}_l \\ \mathbf{I}_m \end{pmatrix}. \quad (8)$$

As shown in Lemma 2, the Jacobian is a constant matrix when $p = 1$. It implies that a linear recurrent model is stable if the spectral norm of $\mathcal{G}$ is sufficiently small. In this case, the model has the short memory as Lemma 1. Next, we give the main claim by proving the Jacobian (7) is unbounded, which implies the high-degree TP recurrent model would be in an unstable regime.

Theorem 3 (High degree models lead to unstable behaviors.). *Given a tensor* $\mathcal{G}$ *with the symmetric structure among the first p indices, assume* $\mathcal{G}$ *has non-zero sub-blocks with respect to* $\mathbf{U} = \left(\mathbf{0}_l^\top \ \mathbf{I}_m\right)^\top$, *i.e.* $\|\mathcal{G} \cdot \mathbf{U}^{\otimes p}\|_2 \neq 0$. *For any positive number* $K > 0$ *and* $p > 1$, *there always exist a pair of vectors* $\mathbf{h} \in \mathbb{R}^m$ *and* $\mathbf{x} \in \mathbb{R}^l$, *such that* $\|J_s(\mathbf{h}; \mathbf{x})\|_2 > K$, *i.e., the model* (3) *is unstable under Def. 2.*

Thm. 3 implies that the TP recurrent model (3) is non-Lipschtiz-continuous when $p > 1$. In this case, the model would stay in the unstable regime according to Def. 2. It is known that the gradients of the loss function explode in unstable models (Hardt et al., 2018). Therefore, Thm. 3 partially reveal the insight why most of (high-degree) tensor models are practically difficult to train, even in the deep feedforward neural networks (Su et al., 2020; Huang et al., 2020b).

3.3 Discussion

The aforementioned results show a desperate picture: it seems difficult for the model to obtain the long memory effect meanwhile operating in a stable regime. To infer its causes, we conjecture *the culprit is the discrete essence of the degree parameter p*. For instance, $p = 1$ (*i.e.*, the linear form) provides the model superior stability with a generally short memory; however, the $p = 2, 3, \ldots$ cases will result in unstable states and a potential long memory. It inspires us to seek the "edge" of such phase transition from the middle of integers (even less than one). As the consequence, we could benefit from the both stability and long-term memory in practice. However, it is non-trivial to achieve the goal because the degree-p tensor power is defined as the multiple folds of tensor products. To tackle the issue, in the next section we reformulate the weight tensor $\mathcal{G}$ in Eq. (3) using the well-known tensor decomposition (TD), which allows us to extend the feasible region of p from the intrinsically discrete to a continuous domain. The extension gives the model the capability of spontaneously learning the optimal degree parameter with respect to the loss function.

4 Degree-learnable approach

Below, we show how to extend the discrete degree parameter to a continuous domain, and empirically introduce a degree-learnable approach for the corresponding RNNs.

4.1 Model description

Recall the TP recurrent model (3) yet adding the bias term for the empirical purpose:

$$\mathbf{h}^{(t)} = \mathcal{G} \cdot \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix}^{\otimes p} + \mathbf{b}, \tag{9}$$

where $\mathbf{b} \in \mathbb{R}^m$. Thus the jth entry of the hidden state $\mathbf{h}^{(t)}$, *i.e.* $\mathbf{h}^{(t)}[j]$, can be rewritten as

$$\mathbf{h}^{(t)}[j] = \left\langle \mathcal{G}_j, \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix}^{\otimes p} \right\rangle + \mathbf{b}[j], \tag{10}$$

where $\mathcal{G}_j \in \mathbb{R}^{n^p}$ denotes the sub-block of $\mathcal{G}$ by fixing the last index to equal j. Then we know the tensor $\mathcal{G}_j$ is super-symmetric (Cichocki et al., 2007). In this form, the parameter p is only defined in the range of non-negative integers, *i.e.*, $p \in \mathbb{Z}^+$. Next, we apply symmetric tensor decomposition (Comon et al., 2008; Brachat et al., 2010) to decomposing the tensor $\mathcal{G}_j$ into factors, *a.k.a.*, latent components (Cichocki et al., 2007). As a result, Eq. (10) can be represented as

$$\mathbf{h}^{(t)}[j] = \sum_{r=1}^{R} \left\langle \mathbf{w}_{j,r}, \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix} \right\rangle^p + \mathbf{b}[j], \tag{11}$$

where $\mathbf{w}_{j,r} \in \mathbb{R}^n$ denotes the r-th factor of $\mathcal{G}_j$ and $R > 0$ corresponds to the symmetric tensor rank. Comon et al. (2008) shows that the decomposition always exists for any symmetric tensor $\mathcal{G}_l$ if the rank R is sufficiently large, which implies the equivalence between Eq. (10) and (11). Apart from the equivalence, we can also see that *the parameter p is converted into a vanilla exponent, which has explicit definition on not $\mathbb{Z}^+$ but the real field* $\mathbb{R}$. It allows us to naturally extend the TP recurrent model to the "real" degree. However, given a non-integer p, note that the exponential term in Eq. (11) is defined only when the base (*i.e.*, the inner product term) is positive. Therefore, we heuristically extract the sign of the base from exponential function, *i.e.*,

$$\mathbf{h}^{(t)}[j] = \sum_{r=1}^{R} a_{j,r} \left| \left\langle \mathbf{w}_{j,r}, \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix} \right\rangle \right|^p + \mathbf{b}[j], \tag{12}$$

where $a_{j,r} := sgn\left(\left\langle \mathbf{w}_{j,r}, \begin{pmatrix} \mathbf{x}^{(t)} \\ \mathbf{h}^{(t-1)} \end{pmatrix} \right\rangle \right)$. Given any $p \in \mathbb{R}$, define an element-wise non-linear function $\phi_p(\cdot) : \mathbb{R}^m \to \mathbb{R}^m$, of which each element $\phi_p(s)$ is given by

$$\phi_p(s) = sgn(s) \cdot |s|^p, \tag{13}$$

then we finally have the extended TP recurrent model by concatenating Eq. (12) over all possible $j \in [m]$, *i.e.*,

$$\mathbf{h}^{(t)} = \sum_{r=1}^{R} \phi_p \left(\mathbf{W}_{hh,r}\mathbf{h}^{(t-1)} + \mathbf{W}_{hx,r}\mathbf{x}^{(t)} \right) + \mathbf{b}, \tag{14}$$

where $\mathbf{W}_{hh,r} \in \mathbb{R}^{m\times m}$ and $\mathbf{W}_{hx,r} \in \mathbb{R}^{m\times l}$ are the weights, where the j-th row of their concatenation corresponds to the vector $\mathbf{w}_{j,r}$ in Eq. (12).

It can be seen that the transition of the hidden states in Eq. (14) results in a *multi-branch* neural network following the activation function ϕ_p, and the number of branches is determined by the maximum of the symmetric tensor rank of $\mathcal{G}_j$, $\forall j \in [l]$.

Remark. We can see that the degree-induced function ϕ_p is able to provide sufficient non-linearity (when $p \neq 1$) to the learning model even if the standard "activations" are omitted. Moreover, Zhao et al. (2020) shows that any saturated continuous activation functions like "tanh" and "sigmoid" lead to short memory of the model. Unlike them, the function ϕ_p is unbounded if $p \neq 0$, and its curvature is controllable in terms of the parameter p. Therefore, the model (14) can be also considered as an activation-learnable recurrent model, which is expected to have the long-memory effect.

4.2 Application in RNNs

The model (14) can be directly employed to replace the original recurrent model in both vanilla RNN and LSTM. For the latter, we suggest a similar way by Yu et al. (2017a), i.e.,

$$\begin{aligned} &[\mathbf{i}^{(t)}, \mathbf{g}^{(t)}, \mathbf{f}^{(t)}, \mathbf{o}^{(t)}] = \\ &\sum_{r=1}^{R} \phi_p \left(\mathbf{W}_{hh,r}\mathbf{h}^{(t-1)} + \mathbf{W}_{hx,r}\mathbf{x}^{(t)} \right) + \mathbf{b}, \\ &\mathbf{c}^{(t)} = \mathbf{c}^{(t-1)} \circ \mathbf{f}^{(t)}, \quad \mathbf{h}^{(t)} = \mathbf{c}^{(t)} \circ \mathbf{o}^{(t)} \end{aligned} \tag{15}$$

where $\circ$ denotes the Hadamard product. In addition, the trick by considering more historic states as (Soltani and Jiang, 2016; Yu et al., 2017a) can be trivially applied to the model (14).

Besides the hidden state, we suggest two methods to learning the degree parameter p in the training phase. The most vanilla way is just to consider p as a trainable variable. Since its feasible range becomes the whole real field $\mathbb{R}$, p can be efficiently trained by stochastic gradient descent (SGD) and its variants. The second way is to learn the parameter p by sub-networks, *i.e.*,

$$p^{(t)} = MLP\left(p^{(t-1)}, \mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}\right), \tag{16}$$

where $MLP(\cdot)$ denotes a multilayer perceptron (MLP), and $p^{(t)}$ is the degree parameter at time-step t. Unlike the first method, the sub-network gives the model the variety of the degree parameters for different time-steps. From the empirical perspective, it is reasonable because the model would have an "attention-like" mechanism to selectively remember more important information from the data.

5 Experiments

In this section, we numerically show the model (14) can achieve the superior performance on various time-series forecasting tasks.

5.1 Single-cell architecture

We first evaluate the performance of the model on *four long memory datasets*. To eliminate the irrelevant influence by the network architecture, we compare the models by employing them in a single-cell RNN framework.

Dataset. We consider one synthetic dataset and three real-world datasets in this experiment, including "ARFIMA" (Zhao et al., 2020), "Dow Jones Industrial Average (DJI)", "Traffic" (UCI, 2019) and "tree-ring (Tree)" (TSDL, 2019). The length of the training, validation, and test sets for each dataset is given in the supplementary material. Zhao et al. (2020) shows that the four datasets have strong long-range dependence, *i.e.*, the long memory property. In the experiment, we perform one-step rolling forecasts on the test set.

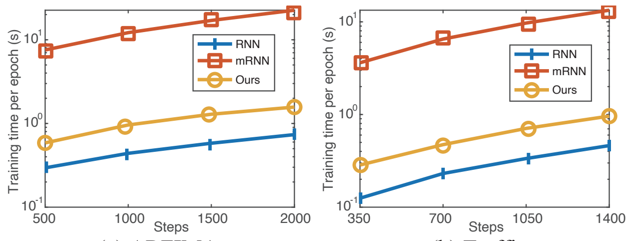
Setup. In our model, we use a two-layers MLP of the hidden dimension equaling 3 to calculate the degree parameter p, and fix the tensor rank R in Eq. (14) to equal 1. Moreover, we also apply the trick mentioned in Soltani and Jiang (2016); Yu et al. (2017a) by taking more historic states (one step more or none) into account, and the model is selected by choosing the one with the best performance on the validation set. In the training phase, we apply the mean square error (MSE) as the loss function, and employ the Adam algorithm with learning rate equaling 0.01 for optimization, which stops after 1000 epochs.

For comparison, we also report the performance of various RNN models given by Zhao et al. (2020), which includes the vanilla RNN (RNN), two-lane RNN with the past 100 values as input (RNN2), recurrent weighted average network (RWA), MIxed hiSTory RNNs (MIST), memory-augmented RNN (MRNN), vanilla LSTM (LSTM), and memory-augmented LSTM (MLST). All the experiments run independently by 50 times, where both the mean value and standard deviation (std.) of the root-mean-square error (RMSE) on the test sets are illustrated.

Results. The RMSE performance by various models is shown in Table 1. We can see that our model (14) outperforms other methods on "ARFIMA", "Traffic" and "Tree", and remains competitive on "DJI". Apart from the mean value, we can also see that our model gives smaller standard deviation than its counterpart like MRNN and MLSTM. Although RWA gives the smallest std. in the experiment, yet its average performance is not comparable to ours. In addition, Figure 1 shows the average training time per epoch under various training size. As shown in Figure 1, our model is twice slower than the vanilla RNN yet 10× faster than MRNN, which has the most closed performance to ours.

Table 1: Performance comparison in terms of RMSE, where the average and standard deviation (in brackets) are reported, and the best results are hightlighted in bold.

	ARFIMA	DJI(×100)	Traffic	Tree
RNN	1.1620 (0.1980)	0.2605 (0.0171)	336.44 (10.401)	0.2871 (0.0086)
RNN2	1.1630 (0.1820)	0.2521 (0.0112)	336.32 (10.182)	0.2855 (0.0077)
RWA	1.6840 (0.0050)	0.2689 (0.0095)	346.62 (1.410)	0.3048 (0.0001)
MIST	1.1390 (0.1832)	0.2604 (0.0154)	358.09 (16.270)	0.2883 (0.0091)
MRNN	1.0880 (0.1140)	**0.2487** (0.0105)	333.72 (10.157)	0.2818 (0.0053)
LSTM	1.1340 (0.1200)	0.2492 (0.0128)	337.60 (8.146)	0.2833 (0.0070)
MLSTM	1.1490 (0.1660)	0.2531 (0.0130)	337.83 (9.440)	0.2859 (0.0083)
Ours	**1.0691** (0.0245)	0.2672 (0.0526)	**329.22** (3.3713)	**0.2799** (0.0023)



(a) ARFIMA (b) Traffic

Figure 1: Training time per epoch with various length of the training sets.

Larger ranks make the network less non-convex. Table 2 shows the results of our model on "ARFIMA" with various tensor rank R. We can see a counter-intuitive phenomenon that a larger tensor rank would neither improve (even degrade) the prediction accuracy nor decrease the training loss. In fact, the similar results have been also reported in several tensor literature (Liu et al., 2018; Li et al., 2020a; Huang et al., 2020b). It is worth noting that, although the averages of loss are almost unchanged when increasing the rank, yet the kurtosis (a concentration measure of the distribution) significantly increases. We infer that the model with large ranks generally has a more flat landscape in the training phase, *i.e.*, the loss is less non-convex, such that the well-trained loss values are less influenced by gradient-based optimizers with different initialization. This result is empirically consistent with a recent study on the landscape of multi-branch feedforward neural networks (Zhang et al., 2019).

Are more historic states really necessary? Table 3 shows the test RMSE of our model on "ARFIMA" and "Tree" armed with various numbers of historic states as Soltani and Jiang (2016); Yu et al. (2017a), where $D_h = 1$ denotes only the current hidden state is used. It can be seen that additional states would improve the prediction accuracy. We infer that more historic information of the hidden states would enhance the short-term prediction capability of the model. On the other hand, the performance would be worse if too much "history" is used. We conjecture that in this case the short-term contribution would *dominate* the prediction performance, such that the model cannot well exploit the long memory effect.

Table 2: Performance of the proposed model on "ARFIMA" under various ranks.

	RMSE		Training loss		
	mean	std.	mean	std.	kurtosis
$R = 5$	1.0851	0.0270	0.0078	0.0003	2.0620
$R = 10$	1.0963	0.0261	0.0079	0.0003	2.6152
$R = 30$	1.1057	0.0315	0.0081	0.0004	3.2933
$R = 50$	1.1103	0.0316	0.0081	0.0004	3.6301

5.2 Seq2seq architecture

Next, we evaluate the effectiveness of our model in a deeper architecture, *seq2seq*, on the forecasting task.

Datasets. We consider one synthetic dataset and two real-word datasets in the experiment, including "Genz" (Yu et al., 2017a), "Traffic of Los Angeles County (TrafficLA)"[3] and "Solar"[4]. The preprocessing details and the data format are introduced in the supplementary material. Unlike the one-step rolling forecast in the single-cell experiment, here we use partial observation to predict the rest of the time series, *i.e.*, relatively long-term prediction.

Table 3: Test RMSE of the proposed model on datasets "ARFIMA" and "Tree". Each row of the table represents the model arms with different numbers of the historic states, where $D_h = 1$ denotes only the current hidden sate is used.

	ARFIMA		Tree	
	mean	std.	mean	std.
$D_h = 1$	1.0828	0.0353	**0.2799**	0.0023
$D_h = 2$	**1.0691**	0.0245	0.2803	0.0021
$D_h = 3$	1.0741	0.0388	0.2805	0.0022
$D_h = 5$	1.0743	0.0348	0.2803	0.0021
$D_h = 10$	1.0835	0.0357	0.2814	0.0018

Setup. We follow a similar setup to the experiments given by Yu et al. (2017a). In our model, we modify the LSTM cells in the seq2seq network as Eqs. (15), and learn the degree parameters by the both trainable variables (Vanilla) and sub-networks (Sub-net) as mentioned in Sec. 4.2. To train the models, we use the sum of square errors as the loss function and RMSprop as the optimizer. For comparison, we employ the seq2seq models with both RNN and LSTM in the experiment. We also compare the performance with the high-order tensor LSTM (HOTLSTM) (Yu et al., 2017a), which is the most related tensor method to ours. More details such as the hyper-parameter search are given in the supplementary material.

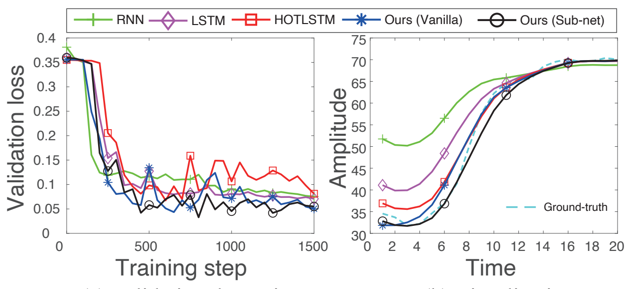
Table 4: The best RMSE performance on the test sets. In our models, "vanilla" indicates directly learning the degree parameters as trainable variables, and "Sub-nets" means they are learned by sub-networks. The column "size" shows the number of trainable parameters for each model (rounding to 100).

	Genz	TrafficLA	Solar	Size
RNN	0.0172	0.0874	0.1273	–
LSTM	0.0085	0.0900	0.0947	3300
HOTLSTM	0.0074	0.0851	0.0933	2600
Ours (Vanilla)	**0.0061**	0.0853	0.0933	**2400**
Ours (Sub-net)	0.0068	**0.0828**	**0.0926**	2900

Results. Table 4 shows the best RMSE performance of various models on the three datasets. We can see that our models outperform not only the conventional

[3]http://pems.dot.ca.gov

[4]https://www.nrel.gov/grid/solar-power-data.html



(a) Validation dynamic (b) Visualization

Figure 2: Validation dynamics on "Genz" and the visualization of the prediction on "TrafficLA".

RNN and LSTM but also the more advanced tensor-based model HOTLSTM, and the model size of our method is comparable. We also show the dynamics of validation loss on "Genz" and the prediction visualization on "TrafficLA" in Figure 2. We can see that our model (Sub-nets, the black line) consistently obtains superior validation loss with growing the training steps, and prediction results visually demonstrate the effectiveness of our model.

6 Discussion and concluding remarks

Our experiments show the recurrent neural networks (RNNs) can benefit from the *learnable* tensor-power (TP) operations, and our theoretical results show the degree parameter plays a critical role in the both memory mechanism and dynamic behaviors of the TP recurrent model.

An interesting finding in the experiments is that growing the tensor rank R in Eq. (14) does not improve the approximation error of the model. It implies that the rank would not be a key factor to determine the approximation capacity of the model[5]. The result is partially counter-intuitive. Therefore, the study of the model's approximation theorem remains important for future work.

Acknowledgement

We thank the anonymous (meta-)reviewers of AISTATS 2021 for helpful comments and discussion. The work is partially supported by JSPS KAKENHI (Grant No. 20K19875, 20H04249, 20H04208), the National Natural Science Foundation of China (Grant No. 62006045), Ningbo Project (Grant No. NBCP 2019C50052), and Nottingham Project (Grant No. NCHI I01200100023).

[5]Note that it is a different issue from the discussion in the work by Khrulkov et al. (2017).

References

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Beran, J., Feng, Y., Ghosh, S., and Kulik, R. (2016). *Long-Memory Processes*. Springer.

Brachat, J., Comon, P., Mourrain, B., and Tsigaridas, E. (2010). Symmetric tensor decomposition. *Linear Algebra and its Applications*, 433(11-12):1851–1872.

Cichocki, A., Zdunek, R., and Amari, S.-i. (2007). Nonnegative matrix and tensor factorization [lecture notes]. *IEEE Signal Processing Magazine*, 25(1):142–145.

Comon, P., Golub, G., Lim, L.-H., and Mourrain, B. (2008). Symmetric tensors and symmetric tensor rank. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1254–1279.

Giles, C. L., Sun, G.-Z., Chen, H.-H., Lee, Y.-C., and Chen, D. (1990). Higher order recurrent networks and grammatical inference. In *Advances in Neural Information Processing Systems*, pages 380–387.

Greaves-Tunnell, A. and Harchaoui, Z. (2019). A statistical investigation of long memory in language and music. In *International Conference on Machine Learning*, pages 2394–2403.

Gulcehre, C., Cho, K., Pascanu, R., and Bengio, Y. (2014). Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 530–546. Springer.

Hardt, M., Ma, T., and Recht, B. (2018). Gradient descent learns linear dynamical systems. *The Journal of Machine Learning Research*, 19(1):1025–1068.

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

Hou, M., Tang, J., Zhang, J., Kong, W., and Zhao, Q. (2019). Deep multimodal multilinear fusion with high-order polynomial pooling. In *Advances in Neural Information Processing Systems*, pages 12136–12145.

Huang, Y., Tang, Y., Zhuang, H., VanZwieten, J., and Cherubin, L. (2020a). Physics-informed tensor-train convlstm for volumetric velocity forecasting. *arXiv preprint arXiv:2008.01798*.

Huang, Z., Li, C., Duan, F., and Zhao, Q. (2020b). H-owan: Multi-distorted image restoration with tensor 1x1 convolution. *arXiv preprint arXiv:2001.10853*.

Khrulkov, V., Hrinchuk, O., and Oseledets, I. (2019). Generalized tensor models for recurrent neural networks. *arXiv preprint arXiv:1901.10801*.

Khrulkov, V., Novikov, A., and Oseledets, I. (2017). Expressive power of recurrent neural networks. *arXiv preprint arXiv:1711.00811*.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Le, H., Tran, T., and Venkatesh, S. (2019). Learning to remember more with less memorization. In *International Conference on Learning Representations*.

Lechner, M. and Hasani, R. (2020). Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*.

Lee, Y., Doolen, G., Chen, H., Sun, G., Maxwell, T., and Lee, H. (1986). Machine learning using a higher order correlation network. Technical report, Los Alamos National Lab., NM (USA); Maryland Univ., College Park (USA).

Li, B., Li, C., Duan, F., Zheng, N., and Zhao, Q. (2020a). Tpfn: Applying outer product along time to multimodal sentiment analysis fusion on incomplete data. In *Proceedings of the European Conference on Computer Vision*.

Li, C., Sun, Z., and Zhao, Q. (2020b). High-order learning model via fractional tensor network decomposition. In *First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*.

Liu, Z., Shen, Y., Lakshminarasimhan, V. B., Liang, P. P., Zadeh, A. B., and Morency, L.-P. (2018). Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256.

Mehta, R., Chakraborty, R., Xiong, Y., and Singh, V. (2019). Scaling recurrent models via orthogonal approximations in tensor trains. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10571–10579.

Miller, J. and Hardt, M. (2018). Stable recurrent models. In *International Conference on Learning Representations*.

Pan, Y., Xu, J., Wang, M., Ye, J., Wang, F., Bai, K., and Xu, Z. (2019). Compressing recurrent neural networks with tensor ring for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4683–4690.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Pollack, J. B. (1991). The induction of dynamical recognizers. In *Connectionist Approaches to Language Learning*, pages 123–148. Springer.

Rabusseau, G., Li, T., and Precup, D. (2019). Connecting weighted automata and recurrent neural networks through spectral learning. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1630–1639.

Royston, P. and Altman, D. G. (1994). Regression using fractional polynomials of continuous covariates: parsimonious parametric modelling. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 43(3):429–453.

Schlag, I. and Schmidhuber, J. (2018). Learning to reason with third order tensor products. In *Advances in Neural Information Processing systems*, pages 9981–9993.

Soltani, R. and Jiang, H. (2016). Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*.

Su, J., Byeon, W., Huang, F., Kautz, J., and Anandkumar, A. (2020). Convolutional tensor-train lstm for spatio-temporal learning. *arXiv preprint arXiv:2002.09131*.

Sun, Z., Ozay, M., Zhang, Y., Liu, X., and Okatani, T. (2018). Feature quantization for defending against distortion of images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7957–7966.

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *International Conference on Mahcine Learning*, pages 1017–1024.

Tjandra, A., Sakti, S., and Nakamura, S. (2017). Compressing recurrent neural network with tensor train. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 4451–4458. IEEE.

Tomioka, R. and Suzuki, T. (2014). Spectral norm of random tensors. *arXiv preprint arXiv:1407.1870*.

Trinh, T., Dai, A., Luong, T., and Le, Q. (2018). Learning longer-term dependencies in rnns with auxiliary losses. In *International Conference on Machine Learning*, pages 4965–4974.

TSDL (2018 (accessed Jun. 14, 2019)). *tsdl: Time Series Data Library.* https://pkg.yangzhuoranyang.com/tsdl/.

UCI (2019 (accessed Dec. 28, 2019)). *Machine Learning Repository metro interstate traffic volumne data set.* https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume.

Voelker, A., Kajić, I., and Eliasmith, C. (2019). Legendre memory units: Continuous-time representation in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 15570–15579.

Wang, C. and Niepert, M. (2019). State-regularized recurrent neural networks. *arXiv preprint arXiv:1901.08817*.

Wang, D., Wu, B., Zhao, G., Chen, H., Deng, L., Yan, T., and Li, G. (2020). Kronecker cp decomposition with fast multiplication for compressing rnns. *arXiv preprint arXiv:2008.09342*.

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., and Salakhutdinov, R. R. (2016). On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 2856–2864.

Yang, Y., Krompass, D., and Tresp, V. (2017). Tensor-train recurrent neural networks for video classification. *arXiv preprint arXiv:1707.01786*.

Ye, J., Wang, L., Li, G., Chen, D., Zhe, S., Chu, X., and Xu, Z. (2018). Learning compact recurrent neural networks with block-term tensor decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9378–9387.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. (2017a). Long-term forecasting using higher order tensor rnns. *arXiv preprint arXiv:1711.00073*.

Yu, R., Zheng, S., and Liu, Y. (2017b). Learning chaotic dynamics using tensor recurrent neural networks. In *ICML Workshop on Deep Structured Prediction*, volume 17.

Zhang, H., Shao, J., and Salakhutdinov, R. (2019). Deep neural networks with multi-branch architectures are intrinsically less non-convex. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1099–1109.

Zhao, J., Huang, F., Lv, J., Duan, Y., Qin, Z., Li, G., and Tian, G. (2020). Do RNN and LSTM have long memory? *arXiv preprint arXiv:2006.03860*.