Do RNN and LSTM have Long Memory?

Jingyu Zhao [1] **Feiqing Huang** [1] **Jia Lv** [2] **Yanjie Duan** [2] **Zhen Qin** [2] **Guodong Li** [1] **Guangjian Tian** [2]

Abstract

The LSTM network was proposed to overcome the difficulty in learning long-term dependence, and has made significant advancements in applications. With its success and drawbacks in mind, this paper raises the question - do RNN and LSTM have long memory? We answer it partially by proving that RNN and LSTM do not have long memory from a statistical perspective. A new definition for long memory networks is further introduced, and it requires the model weights to decay at a polynomial rate. To verify our theory, we convert RNN and LSTM into long memory networks by making a minimal modification, and their superiority is illustrated in modeling long-term dependence of various datasets.

1. Introduction

Sequential data modeling is an important topic in machine learning, and it has been well addressed by a variety of recurrent networks. In the meanwhile, modeling long-range dependence remains a key challenge. Bengio et al. (1994) concluded that learning long-term dependencies by a system $y_t = M(y_{t-1}) + \varepsilon_t$ with gradient descent is difficult. Later, much effort was spent on proposing new networks to fight against vanishing gradients, such as LSTM (Hochreiter & Schmidhuber, 1997) and GRU networks (Cho et al., 2014).

LSTMs have shown their success on both synthetic classification tasks, such as the "parity" problem and the "latching" problem (Hochreiter & Schmidhuber, 1997), and countless applications, including speech recognition and machine translation. Despite its success, questions concerning the ability of LSTM in handling long-term dependence arise. For example, Cheng et al. (2016) pointed out that the update equation of LSTM is Markovian, which is consistent with the system assumed in Bengio et al. (1994). Greaves-Tunnell & Harchaoui (2019) utilized statistical tests and concluded that LSTM cannot fully represent the long memory effect in the input, nor can it generate long memory sequences from white noise inputs.

Does LSTM have long memory? It is difficult to judge if long memory is assessed heuristically by the performance of LSTM on certain tasks. However, long memory happens to be a well-defined and long-existing terminology in statistics. From a statistical perspective, our investigation yields an affirmative reply. Our first contribution is to prove that a recurrent network process with Markovian update dynamics does not have long memory under mild conditions.

Nevertheless, the statistical definition of long memory is neither applicable nor easily transferable to general neural networks. The main obstacle is the existence of non $i.i.d.$ exogenous input to the network, violating a key assumption in the study of stochastic processes. The long or short memory effect in the input confounds the memory property of the output sequence. Thus, we propose a new definition for the long memory network process, which aims to characterize the ability of a network process to extract long-term dependence from the input.

We further explore the possible implications of our theory in practice. Our analysis implies that we cannot assume that information can be stably stored in the hidden states of a recurrent network with Markovian updates. Subsequently, providing the hidden states with more past information is a natural strategy to avoid the vanishing gradient problem.

In terms of network design, we propose a new memory filter component, which is controlled by a learnable memory parameter d. The filter uses d to deduce dynamic weightings on historical observations of arbitrary lengths and feeds past information to the hidden units. We incorporate this memory filter structure into RNN and LSTM and introduce Memory-augmented RNN (MRNN) and Memory-augmented LSTM (MLSTM). By adding this memory filter, the modified models achieve the ability to approximate the fractional differencing effect, which is a type of long memory effect, and retain the flexibility and expressiveness of neural networks.

The main contributions of this work are below:

- We provide theoretical conditions for recurrent net-

[1]Department of Statistics and Actuarial Science, The University of Hong Kong, Hong Kong, China [2]Huawei Noah's Ark Lab, Hong Kong, China. Correspondence to: Guodong Li <gdli@hku.hk>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

works, such as RNN and LSTM, to have short memory.

- We propose a new definition for long memory network processes under a statistical motivation, and make it applicable to neural networks.
- We propose a long memory filter component for network design. Modifying RNN and LSTM with this filter allows the networks to better model sequences with long memory effects.
- We perform numerical studies on several datasets to illustrate the advantages of our proposed models.

1.1. Related Work

Besides the aforementioned papers, the following works are also related to our paper in different aspects.

Long memory phenomenon and statistical models Long memory effect can be observed in many datasets, including language and music (Greaves-Tunnell & Harchaoui, 2019), financial data (Ding et al., 1993; Bouri et al., 2019), dendrochronology and hydrology (Beran et al., 2016). There are many statistical results and applications on modeling datasets with long memory (Grange & Joyeux, 1980; Hosking, 1981; Ding et al., 1993; Torre et al., 2007; Bouri et al., 2019; Musunuru et al., 2019; Sabzikar et al., 2019). Admittedly, these models enjoy rigorous statistical properties. However, they appear to be quite rigid and inflexible for real-world applications in comparison to neural networks.

Analysis of LSTM networks There is limited theoretical analysis of LSTM networks in the literature. Miller & Hardt (2018) studied RNN and LSTM as dynamic systems and proved that r-step LSTM is a stable procedure, which implies that LSTM cannot model long-range dependence well. There also exists some work that tried to analyze LSTM numerically. For example, Levy et al. (2018) studied which component of LSTM contributes the most to its success by ablation experiments, and concluded that the memory cell is largely responsible for the performance of LSTM. This is consistent with our view that the core of LSTM is the update equation of the cell state, which takes the form of a varying coefficient vector AR(1) model.

Recurrent networks for long-term dependencies Examples include the Hierarchical RNN (El Hihi & Bengio, 1996), where several layers of state variables are constructed at different time scales, and direct linkages are added between distant historical and current observations. Zhang et al. (2016) propose the skip connections among multiple timestamps to account for long-range temporal dependencies. The idea is further extended by Chang et al. (2017) with the Dilated RNNs, where the number of parameters is reduced to enhance computation efficiency. The NARX RNN (Lin et al., 1996) revised the hidden states of an RNN to follow a Nonlinear AutoRegressive model with eXogenous variables (NARX). Based on NARX RNN, MIxed hiSTory RNNs (MIST RNNs) (DiPietro et al., 2017) was proposed to reduce the number of trainable weights and link to hidden units in the distant past. The recurrent weighted average (RWA) model (Ostmeyer & Cowell, 2019) is also considered very akin to our proposed models. Thus, we compare our models with MIST RNN and RWA in the experiments.

Memory networks and attention Another popular trend of modeling long-range dependence lies in creating an external memory unit. The main controller interacts with the stand-alone memory unit through reading and writing heads. Neural Turing Machine (Graves et al., 2014) is perhaps the most famous example of such construction. Based on this idea, the attention mechanism (Vaswani et al., 2017) is also proposed. These models have gone beyond a recurrent nature and thus are not compared with our proposed models.

Finite impulse response (FIR) filters and signal processing Our proposed memory filter can be viewed as a special FIR, and we propose to add it to RNN at the input and LSTM at the cell states. Literatures on using filters on inputs include TFLSTM (Li et al.), SRU (Oliva et al., 2017) and Convolutional RNN (Keren & Schuller, 2016), while those for filters on hidden states can be found in NARX RNN (Lin et al., 1996) and Low-pass RNN (Stepleton et al., 2018). Our filter is directly motivated by the ARFIMA model in statistics and shares the same root with the fractional-order filters (Radwan et al., 2008; 2009a;b) in signal processing. The fractional-order filters enjoy similar properties with ours in terms of modeling long-range dependencies.

2. Memory Property of Recurrent Networks

2.1. Background

For a stationary univariate time series, there exists a clear definition of long memory (or long-range dependency) in statistics (Beran et al., 2016), and we state it below. It is noticeable that Greaves-Tunnell & Harchaoui (2019) utilized the following definition to claim the long memory in language and music data.

Definition 1. *Let $\{X_t, t \in \mathbb{Z}\}$ be a second-order stationary univariate process with autocovariance function $\gamma_X(k)$ for all $k \in \mathbb{Z}$ and spectral density $f_X(\lambda) = (2\pi)^{-1} \sum_{k=-\infty}^{\infty} \gamma_X(k) \exp(-ik\lambda)$ for $\lambda \in [-\pi, \pi]$. Then $\{X_t\}$ has (a) long memory, or (b) short memory if*

$$(a) \sum_{k=-\infty}^{\infty} \gamma_X(k) = \infty, \text{ or } (b)\ 0 < \sum_{k=-\infty}^{\infty} \gamma_X(k) < \infty. \tag{1}$$

In terms of spectral densities, the conditions are, as $|\lambda| \to 0$,

$$(a) f_X(\lambda) \to \infty, \text{ or } (b)\ f_X(\lambda) \to c_f \in (0, \infty). \tag{2}$$

There are many types of long memory processes, and the fractionally integrated process is the most commonly used one in real applications; see Beran et al. (2016). Let B be the backshift operator, defined as $B^j X_t = X_{t-j}$ for a $j \geq 0$, applicable to all random variables in a time series $\{X_t\}$. The fractionally integrated process (Hosking, 1981) is defined as

$$(1-B)^d Y_t = X_t,$$

where

$$(1-B)^d = \sum_{j=0}^{\infty} \frac{\Gamma(d+j)}{j!\Gamma(d)} B^j =: \sum_{j=0}^{\infty} w_j(d) B^j, \qquad (3)$$

and $\Gamma(\cdot)$ is the gamma function. The sequence $\{X_t\}$ is usually assumed to follow an ARMA model to capture the short-range dependency, and it then leads to a fractional ARIMA process of $\{Y_t\}$ (Grange & Joyeux, 1980). It can be verified that

$$w_j(d) \sim j^{-d-1}$$

as $j \to \infty$, and $\gamma_X(k) \sim |k|^{2d-1}$ as $k \to \infty$. As a result, time series $\{Y_t\}$ exhibit long memory when $d \in (0, 0.5)$, and larger d indicates longer memory. In other words, the long-range dependence of fractionally integrated processes can be characterized by the parameter d, which hence is called the memory parameter.

For multivariate time series, its definition of long memory is not unique, and a usual way is to check the long-range dependency of each component (Greaves-Tunnell & Harchaoui, 2019). Accordingly the multivariate fractionally integrated process $\{Y_t\}$ can be defined as $(I-\mathcal{B})^d Y_t = X_t$, where $Y_t \in \mathbb{R}^q$, $X_t \in \mathbb{R}^q$, $d = (d_1, \ldots, d_q)'$,

$$(I-\mathcal{B})^d Y_t = \begin{pmatrix} (1-B)^{d_1} & & 0 \\ & \ddots & \\ 0 & & (1-B)^{d_q} \end{pmatrix} Y_t, \quad (4)$$

and $\{X_t\}$ can be a multivariate ARMA process.

Many time series models can be rewritten into a discrete-time Markov chain. Their stationarity can be checked by verifying the geometric ergodicity, defined below.

Definition 2. *Let $\{X_t, t \in \mathbb{Z}\}$ be a temporal homogeneous Markov chain with state space $(S, \mathcal{F})$, where the σ-field $\mathcal{F}$ of subsets of S is assumed to be countably generated. Let $P^n(x, A) = P(X_{t+n} \in A | X_t = x)$ be the n-step transition function. $\{X_t\}$ is geometrically ergodic if there exists a $0 < \rho < 1$ such that for every $x \in S$ and some probability measure π on $\mathcal{F}$*

$$\rho^{-n} \| P^n(x, \cdot) - \pi \| \to 0 \text{ as } n \to \infty, \qquad (5)$$

where $\| \cdot \|$ denotes the total variation of signed measures on $\mathcal{F}$.

For a geometrically ergodic process $\{X_t\}$, from Harris Theorem (Meyn & Tweedie, 2012), we have that $\gamma_X(k) \sim \rho^k$ as $k \to \infty$, and it hence has short memory. Heuristically, geometric ergodicity implies that the Markov chain converges to its stationary distribution exponentially fast. This means that information in the past is forgotten exponentially fast, since the influence of observation $X_t = x$ on the distribution of X_{t+n} vanishes exponentially.

2.2. Recurrent Network Process

Consider a general recurrent network with input $\{x^{(t)}\}$, output $\{z^{(t)}\}$ and target sequence $\{y^{(t)}\}$, where $z^{(t)} \in \mathbb{R}^p$ and $y^{(t)} \in \mathbb{R}^p$. We first assume that the data generating process (DGP) is

$$y^{(t)} = z^{(t)} + \varepsilon^{(t)}, \quad \text{for } t \in \mathbb{Z}, \qquad (6)$$

where $\{\varepsilon^{(t)}\}$ is a sequence of independent and identically distributed ($i.i.d.$) random vectors. This additive error assumption corresponds to many frequently used loss functions, which measure the distance between $y^{(t)}$ and $z^{(t)}$. Examples include l_1, l_2, Huber and quantile loss.

We use the term *network process* to describe the generated target sequence by (6), and the term *network* to describe the implemented networks, which might include some compromises and simplifications.

We further assume that there is no exogenous input, i.e., $x^{(t)} = y^{(t-1)}$, since the long-range dependence in an exogenous input will confound the memory properties of $\{y^{(t)}\}$. We aim to check whether a network can model long memory sequences, and it is then equivalent to verifying whether the corresponding network process has long memory under Definition 1.

General hidden states $s^{(t)} \in \mathbb{R}^q$ are also introduced so that the recurrent network process (6) can be written into a homogeneous Markov Chain with transition function $\mathcal{M}$:

$$\begin{pmatrix} y^{(t)} \\ s^{(t)} \end{pmatrix} = \mathcal{M}\left(y^{(t-1)}, s^{(t-1)}\right) + \begin{pmatrix} \varepsilon^{(t)} \\ 0 \end{pmatrix}. \qquad (7)$$

When transition function $\mathcal{M}$ is linear in $y^{(t-1)}$ and $s^{(t-1)}$, the above equation also has the form of

$$\begin{pmatrix} y^{(t)} \\ s^{(t)} \end{pmatrix} = W \begin{pmatrix} y^{(t-1)} \\ s^{(t-1)} \end{pmatrix} + \begin{pmatrix} \varepsilon^{(t)} \\ 0 \end{pmatrix}, \qquad (8)$$

where W is a $(p+q)$-by-$(p+q)$ transition matrix.

The first example is the vanilla RNN. Consider a many-to-many RNN structure for time series prediction problems, and use square loss as an example:

$$\begin{cases} l^{(t)} = \left\| y^{(t)} - z^{(t)} \right\|^2 \\ z^{(t)} = g(W_{zh} h^{(t)} + b_z) \\ h^{(t)} = \sigma(W_{hh} h^{(t-1)} + W_{hy} y^{(t-1)} + b_h) \end{cases}, \qquad (9)$$

for $t \in \{1, ..., T\}$, where $y^{(0)} = 0$, $h^{(0)} = 0$, $y^{(t)}, z^{(t)} \in \mathbb{R}^p$, $h^{(t)} \in \mathbb{R}^q$, g is an elementwise output function, σ is an elementwise activation function. Accordingly the RNN process can be defined as

$$\begin{pmatrix} y^{(t)} \\ h^{(t)} \end{pmatrix} = \mathcal{M}_{\text{RNN}} \left(y^{(t-1)}, h^{(t-1)} \right) + \begin{pmatrix} \varepsilon^{(t)} \\ 0 \end{pmatrix} \tag{10}$$

where $h^{(t)}$ is the hidden state, and $\mathcal{M}_{\text{RNN}}(\cdot, \cdot) : \mathbb{R}^{p+q} \to \mathbb{R}^{p+q}$ is a function given by

$$\mathcal{M}_{\text{RNN}}(u, v) = \begin{pmatrix} g(W_{zh}\sigma(W_{hh}v + W_{hy}u + b_h) + b_z) \\ \sigma(W_{hh}v + W_{hy}u + b_h) \end{pmatrix}, \tag{11}$$

with $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$.

The second example is an LSTM network

$$\begin{cases} l^{(t)} = \|y^{(t)} - z^{(t)}\|^2 \\ z^{(t)} = g(W_{zh}h^{(t)} + b_z) \end{cases}, \text{ for } t \in \{1, ..., T\}, \tag{12}$$

where the hidden unit calculations involve several gating operations as follows,

$$\begin{cases} f^{(t)} = \sigma(W_{fh}h^{(t-1)} + W_{fy}y^{(t-1)} + b_f) \\ i^{(t)} = \sigma(W_{ih}h^{(t-1)} + W_{iy}y^{(t-1)} + b_i) \\ o^{(t)} = \sigma(W_{oh}h^{(t-1)} + W_{oy}y^{(t-1)} + b_o) \\ \tilde{c}^{(t)} = \tanh(W_{ch}h^{(t-1)} + W_{cy}y^{(t-1)} + b_c) \\ c^{(t)} = i^{(t)} \odot \tilde{c}^{(t)} + f^{(t)} \odot c^{(t-1)} \\ h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \end{cases}, \tag{13}$$

for $t \in \{1, ..., T\}$, where $y^{(0)} = 0$, $h^{(0)} = 0$, $y^{(t)}, z^{(t)} \in \mathbb{R}^p$, $h^{(t)}, c^{(t)}, \tilde{c}^{(t)}, f^{(t)}, i^{(t)}, o^{(t)} \in \mathbb{R}^q$, g is an elementwise output function, σ is the elementwise sigmoid function, $\tanh$ is the elementwise hyperbolic tangent function, $\odot$ is the elementwise product. Similarly the LSTM process can be defined as

$$\begin{pmatrix} y^{(t)} \\ h^{(t)} \\ c^{(t)} \end{pmatrix} = \mathcal{M}_{\text{LSTM}} \left(y^{(t-1)}, h^{(t-1)}, c^{(t-1)} \right) + \begin{pmatrix} \varepsilon^{(t)} \\ 0 \\ 0 \end{pmatrix} \tag{14}$$

where $h^{(t)}$ and $c^{(t)}$ are hidden states defined in (7), the transition function $\mathcal{M}_{\text{LSTM}}(\cdot, \cdot, \cdot) : \mathbb{R}^{p+q+q} \to \mathbb{R}^{p+q+q}$ has a complicated form, and hence omitted here.

2.3. Memory Property of Recurrent Network Processes

We first introduce a technical assumption, and then state two general theorems for recurrent network processes.

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

Theorem 1. *Under Assumptions 1, if there exist real numbers $0 < a < 1$ and b such that $\|\mathcal{M}(x)\| \leq a\|x\| + b$, then recurrent network process (7) is geometrically ergodic, and hence has short memory.*

Table 1. Restrictions on weights such that the RNN process is geometrically ergodic.

Output function g	Activation function σ	
	identity or ReLU	sigmoid or tanh
identity	$\|w_{zh}w_{hh}\| \leq a$, $\|w_{zh}w_{hy}\| \leq a$, $\|w_{hh}\| \leq a, \|w_{hy}\| \leq a$	No
sigmoid	$\|w_{hh}\| \leq a, \|w_{hy}\| \leq a$	No
softmax	$\|w_{hh}\| \leq a, \|w_{hy}\| \leq a$	No

Theorem 2. *Under Assumption 1, linear recurrent network process (8) is geometrically ergodic if and only if spectral radius $\rho(W) < 1$. Model (8) hence has short memory.*

Technical proofs of the above two theorems are deferred to the supplementary material. Theorem 2 gives a sufficient and necessary condition, while Theorem 1 provides only a sufficient condition since it is usually difficult to achieve a sufficient and necessary condition for a nonlinear model (Zhu et al., 2018).

It is implied by Theorems 1 and 2 that both RNN and LSTM have no capability of handling a stable time series with long-range dependence.

Specifically we consider a RNN process at (10) with $p = q = 1$. Assume that the norm $\|\cdot\|$ in Theorem 1 takes l_1 norm, the output function g is linear, sigmoid or softmax, and the activation function σ is ReLU, sigmoid or hyperbolic tangent. Table 1 gives the ranges of weights so that the RNN process is geometrically ergodic with short memory.

For linear or ReLU activation, RNN has short memory when the magnitude of the weights is bounded away from one. This condition is often satisfied for numerically stable RNNs. Thus, RNN with linear or ReLU activation often has short memory. Moreover, in practice, the activation function in RNN is commonly chosen as $\tanh$. According to Table 1, RNN with $\tanh$ or sigmoid activation always has short memory. In fact, this holds for general RNN processes with any bounded and continuous output and activation function; see Corollary 1 below.

Corollary 1. *Suppose that the output and activation functions, $g(\cdot)$ and $\sigma(\cdot)$, at (10) are continuous and bounded. If Assumption 1 holds, then the RNN process is geometrically ergodic and has short memory.*

We next consider the LSTM process at (14), and a detailed analysis of one-dimensional LSTM, similar to the analysis presented in Table 1, can be referred to in the supplementary material. The restrictions on weights such that an LSTM process is geometrically ergodic is given below.

Corollary 2. *The input series features $\{y^{(t-1)}\}$ are scaled to the range of $[-1, 1]$. Suppose that $M :=$*

$\sup_{x\in B_\infty^q} \|g(W_{zh}x + b_z)\|_{l_1} < \infty$ *and* $\sigma(\|W_{fh}\|_{l_\infty} + \|W_{fy}\|_{l_\infty} + \|b_f\|_{l_\infty}) \leq a$ *for some* $a < 1$*, where* B_∞^q *is the* q*-dimensional* l_∞*-ball and* $\|W\|_{l_\infty} = \max_{1\leq i\leq m}\sum_{j=1}^n |w_{ij}|$ *is the matrix* l_∞*-norm. If Assumption 1 holds, then the LSTM process at* (14) *is geometrically ergodic and has short memory.*

It is worthy to point out that the condition of $M < \infty$ is satisfied by many commonly used output functions, including linear function, ReLU, sigmoid or $\tanh$. From the conditions in Corollary 2, we may conclude that the forget gate mainly affects the memory property of an LSTM.

2.4. Long Memory Network Process

In the previous subsection, we verify that the two commonly used recurrent networks, RNN and LSTM, are both short memory when there is no exogenous input. However, it is very common of $\{x^{(t)}\}$ to include exogenous inputs, and the long-range dependence in $\{x^{(t)}\}$ will confound the memory property of $\{y^{(t)}\}$. It is then misleading to check the long-range dependence of a network by verifying that of the corresponding network process.

In the literature of conditional heteroscedastic time series models, there also exists vast interest in long-range dependence; see, e.g., the fractionally integrated GARCH (FIGARCH) models (Davidson, 2004), hyperbolic GARCH (HYGARCH) models (Li et al., 2015), etc. However, for a stationary conditional heteroscedastic process, its autocovariance function is always summable, i.e. it always has short memory in terms of Definition 1.

Consider a stationary conditional heteroscedastic model, $y_t = \eta_t\sqrt{h_t}$ and $h_t = \omega + \sum_{k=1}^\infty \theta_k y_{t-k}^2$, where $\{\eta_t\}$ is a sequence of $i.i.d.$ random variables. Davidson (2004) defined that it has long memory if the coefficients $\{\theta_k\}$ have hyperbolic decay. This definition has been widely used in the literature. Moreover, the commonly used fractional ARIMA process also has the form of $y^{(t)} = \sum_{k=1}^\infty a_k y^{(t-k)} + \varepsilon^{(t)}$, where $a_k \sim k^{-d-1}$ as $k \to \infty$. This motivates us to propose a new definition of long memory for a network process,

$$y^{(t)} = \sum_{k=0}^{\infty} A_k x^{(t-k)} + \varepsilon^{(t)}, \tag{15}$$

where $y^{(t)} \in \mathbb{R}^p$ is the generated target, $x^{(t)} \in \mathbb{R}^q$ is the input, $q \times p$ coefficient matrices $A_k = \{(A_k)_{ij}\}$ with $1 \leq i \leq p$ and $1 \leq j \leq q$ and $\{\varepsilon^{(t)}\}$ are $i.i.d.$ random vectors.

Definition 3. *The network process at* (15) *has long memory if there exist* $1 \leq i \leq p$ *and* $1 \leq j \leq q$ *such that*

$$(A_k)_{ij} \sim k^{-d-1}$$

as $k \to \infty$*, where* $d \in (0, 0.5)$ *is the memory parameter.*

The coefficients $\{A_k\}_{k\in\mathbb{N}}$ reflect the dependence of $y^{(t)}$ on $\{x^{(t)}\}$. If $(A_k)_{ij}$ decays slowly at a polynomial rate, we can conclude that $y_i^{(t)}$ is long-term dependent on $x_j^{(t)}$ according to Definition 1. This definition is consistent with the visual method used by Lin et al. (1996) since the Jacobian $J(t,k) = \partial y^{(t)}/\partial x^{(t-k)}$ equals A_k and exhibits slow decay under Definition 3.

This definition admits extensions to nonlinear networks. Firstly, we can linearize a network by letting all output and activation functions be the identity function, and we call the resulting network process as its *linear network process*. A network can handle data with long-range dependence if its linear network process has long memory in terms of Definition 3. Note that, for some networks such as LSTM, their linear network processes are still nonlinear. Moreover, nonlinear networks can often be well approximated by linear networks via polynomial approximations and introducing the powers of $x^{(t)}$ into the input; see Yu et al. (2017) as an example. Lastly, we may resort to a first-order approximation $y^{(t)} \approx \sum_{k=0}^\infty J(t,k)\, x^{(t-k)} + \varepsilon^{(t)}$ as in Lin et al. (1996) and check the decay pattern of the Jacobians.

3. Long Memory Recurrent Networks

Motivated by the fractionally integrated process at (4) and the new definition of long memory in Section 2.4, we propose a long memory filter structure that can be added to neural networks to enable modeling long-term dependence. This long memory filter can be viewed as a special attention mechanism with only a few memory parameters and a guaranteed memory elongation effect when active. In Sections 3.1 and 3.2, we modify the RNN and the LSTM network by the proposed long memory filter structure, respectively.

3.1. Memory-augmented RNN (MRNN)

Long memory pattern is introduced via a memory filter,

$$F(x^{(t)}; d) = ((I - \mathcal{B})^d - I)x^{(t)} \tag{16}$$

for $x^{(t)} \in \mathbb{R}^{p_x}$ with p_x being the dimension of inputs, where $d = (d_1, \ldots, d_{p_x})'$; see (4) for details. The memory filter can be viewed as soft attention to a reasonably sized memory with only a few memory parameters of d_is. Note that, from (3), the ith element of the memory filter is $F(x^{(t)}; d)_i = ((1 - B)^{d_i} - 1)x_i^{(t)} = \sum_{j=1}^\infty w_j(d_i)x_i^{(t-j+1)}$, where $w_j(d) = \Gamma(d + j)/[j!\Gamma(d)] = \prod_{i=0}^{j-1}(i - d)/(i + 1)$. To implement this model, we truncate this infinite summation at lag K. In our experiments, we choose $K = 100$ by default, since, taking $d = 0.4$ as an example, $w_{100}(0.4) \approx -4.27 \times 10^{-4}$ is already small enough.

In MRNN, we introduce a long memory hidden unit

$$m^{(t)} = \tanh(W_{mm}m^{(t-1)} + W_{mf}F(x^{(t)}; d) + b^{(m)}),$$

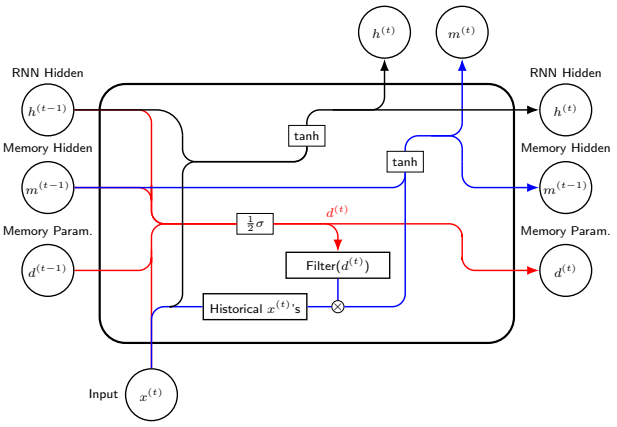

Figure 1. The MRNN cell structure.

which shares a similar structure with $h^{(t)}$, while the input term is replaced with the filter $F(x^{(t)}; d)$. The unit $m^{(t)}$ works in parallel with the traditional hidden unit $h^{(t)}$ and takes responsibility for modeling the long memory pattern in time series. The underlying network can then be designed,

$$z^{(t)} = g(W_{zh}h^{(t)} + W_{zm}m^{(t)} + b_z) + \varepsilon^{(t)}, \quad \text{for } t \in \mathbb{Z}$$

where $z^{(t)} \in \mathbb{R}^{p_z}$ with p_z being the dimension of outputs, and $h^{(t)} = (h_1^{(t)}, \cdots, h_q^{(t)})' \in \mathbb{R}^q$ is the same as the hidden states in RNN.

For the memory parameter $d = (d_1, \ldots, d_{p_x})'$, we restrict $0 < d_i < 0.5$ such that the fractional integration can provide long memory. Moreover, to make the design more general, we also let the memory parameter d depend on other variables, and hence the notation $d^{(t)}$. As a result, the procedure of MRNN is given below.

$$\begin{cases} l^{(t)} = \|y^{(t)} - z^{(t)}\|^2 \\ z^{(t)} = g(W_{zh}h^{(t)} + W_{zm}m^{(t)} + b_z) \\ h^{(t)} = \tanh(W_{hh}h^{(t-1)} + W_{hx}x^{(t)} + b_h) \\ F(x^{(t)}; d^{(t)})_i = \sum_{j=1}^{K} w_j(d_i^{(t)})x_i^{(t-j+1)} \\ d^{(t)} = \frac{1}{2}\sigma(W_d[d^{(t-1)}, h^{(t-1)}, m^{(t-1)}, x^{(t)}] + b_d) \\ m^{(t)} = \tanh(W_m[m^{(t-1)}, F(x^{(t)}; d^{(t)})] + b_m) \end{cases} \tag{17}$$

for $i \in \{1, ..., p_x\}, t \in \{1, ..., T\}$, where $h^{(t)}, m^{(t)} \in \mathbb{R}^q$, $d^{(t)}, x^{(t)} \in \mathbb{R}^{p_x}$, and σ is the sigmoid activation function.

Figure 1 gives a graphical representation of the MRNN cell structure, imitating the style of Olah (2015). The memory filter *Filter*$(d^{(t)})$ refers to (16), and *Historical* $x^{(t)}$*'s* are $(x^{(t)}, x^{(t-1)}, ..., x^{(t-K+1)})$, where we treat $x^{(s)} = 0$ for $s \leq 0$.

For the network with memory parameters constant over time points t, we refer to it as the MRNNF model, and it can be implemented by fixing $W_d = 0$ in the update equation (17).

Theorem 3. *In terms of Definition 3, the MRNNF has the capability of handling long-range dependence data, while the RNN cannot.*

Technical proof is provided in the supplementary material.

3.2. Memory-augmented LSTM (MLSTM)

Corollary 2 indicates that the forget gates determine the memory property of the LSTM. The update equation of cell states has a varying coefficient vector AR(1) form,

$$c^{(t)} - f^{(t)} \odot c^{(t-1)} = i^{(t)} \odot \tilde{c}^{(t)},$$

which has short memory when the coefficients $f^{(t)}$s are smaller than one. Thus, we propose to revise the cell states of LSTM by adding the long memory filter to it

$$(I - \mathcal{B})^d c^{(t)} = i^{(t)} \odot \tilde{c}^{(t)},$$

where the memory parameter d can depend on other variables as for the MRNN in the previous subsection. The MLSTM cell structure is shown in Figure 2. The revised cell states can be viewed as paying soft attention to past cell states controlled by only a few memory parameters.

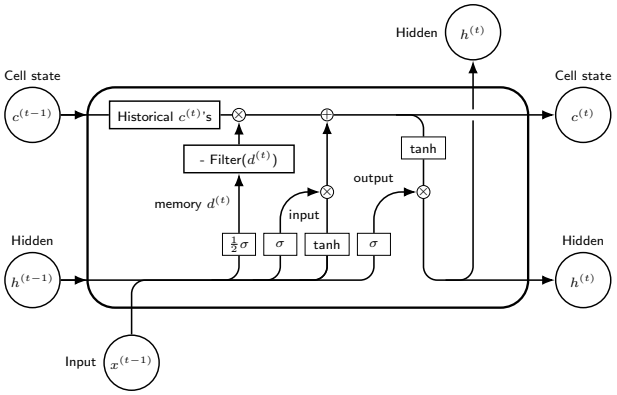

Figure 2. The MLSTM cell structure.

The underlying network can then be designed as

$$z^{(t)} = g(W_{zh}h^{(t)} + b_z) + \varepsilon^{(t)}, \quad \text{for } t \in \mathbb{Z}, \tag{18}$$

where the hidden unit $h^{(t)}$ is produced by the MLSTM cell. We rename the forget gate as the memory gate and modify the update equations as

$$\begin{cases} d^{(t)} = \frac{1}{2}\sigma(W_d[d^{(t-1)}, h^{(t-1)}, x^{(t)}] + b_d) \\ i^{(t)} = \sigma(W_{ih}h^{(t-1)} + W_{ix}x^{(t)} + b_i) \\ o^{(t)} = \sigma(W_{oh}h^{(t-1)} + W_{ox}x^{(t)} + b_o) \\ \tilde{c}^{(t)} = \tanh(W_{ch}h^{(t-1)} + W_{cx}x^{(t)} + b_c) \\ (I - \mathcal{B})^d c^{(t)} = i^{(t)} \odot \tilde{c}^{(t)} \ \text{ with } \ d = d^{(t)} \\ h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \end{cases}, \tag{19}$$

for $t \in \{1, ..., T\}$, where $(I - \mathcal{B})^d$ is defined as in (4).

As for the MLSTM, to implement this model, we truncate the infinite summation $(I-\mathcal{B})^d c^{(t)}$ at lag K. The procedure related to the MLSTM cell states is implemented as

$$c_i^{(t)} = -\sum_{j=1}^{K} w_j(d_i^{(t)}) c_i^{(t-j)} + i^{(t)}\tilde{c}^{(t)}, \qquad (20)$$

where $c_i^{(t)}$ is the i-th element of $c^{(t)}$, $w_j(d) = \Gamma(d+j)/[j!\Gamma(d)] = \prod_{i=0}^{j-1}(i-d)/(i+1)$. We remark that the negative sign before the summation in equation (20) is introduced by the definition of $(I - \mathcal{B})^d$. In MRNN, we let the negative sign be absorbed by the weight matrix W_{mf} for elegance.

Note that the cell state $\{c^{(t)}\}$ has long memory in terms of Definition 3. In the meanwhile, due to the gating mechanism, neither LSTM nor MLSTM can be reasonably simplified to a linear network, i.e. their linear network processes are both nonlinear. However, if we assume that the gates are learnable constants independent of the hidden unit and the inputs, we can prove that the constant-gates-LSTM does not have long memory, while the constant-gates-MLSTM is a long memory network according to Definition 3. We defer the formal statement of this auxiliary result and its proof to the supplementary material.

Similar to MRNNF, we refer MLSTMF to the case with constant memory parameter over time t, and it can be implemented via fixing $W_d = 0$ in the update equation (19).

4. Experiments

This section reports several numerical experiments. We first compare the models using time series forecasting tasks on four long memory datasets and one short memory dataset. Then, we investigate the effect of the model parameter K on the forecasting performance. Lastly, we apply the proposed models to two sentiment analysis tasks.

All the networks are implemented in `PyTorch`. [1] We use the Adam algorithm with learning rate 0.01 for optimization. The optimization is stopped when the loss function drops by less than 10^{-5} or has been increasing for 100 steps or has reached 1000 steps in total. The learned model is chosen to be the one with the smallest loss on the validation set.

Considering the non-convexity of the optimization, we initialize with 100 different random seeds and arrive at 100 different trained models for each model. We refer to the distribution of these 100 results as the *overall performance*, and best of them as the *best performance*. The overall performance reflects what we can expect from a locally optimal model, and the best performance is closer to the outcome of a globally optimal model.

[1] Our implementation in `PyTorch` is available at https://github.com/huawei-noah/noah-research/tree/master/mRNN-mLSTM.

4.1. Long Memory Datasets

We compare our models with the baselines on one synthetic dataset and three real datasets. We split the datasets into training, validation and test sets, and report their lengths below using notation $(n_{train} + n_{val} + n_{test})$. MSE is the target loss function for training. We perform one-step rolling forecasts and calculate test RMSE, MAE, and MAPE.

ARFIMA series We generated a series of length 4001 $(2000+1200+800)$ using the model $(1-0.7B+0.4B^2)(1-B)^{0.4}Y_t = (1-0.2B)\varepsilon_t$ with obvious long memory effect.

Dow Jones Industrial Average (DJI) The raw dataset contains DJI daily closing prices from 2000 to 2019 obtained from Yahoo Finance. We convert it to absolute log return for 5030 $(2500 + 1500 + 1029)$ days in order to model the long memory effect in volatility.

Metro interstate traffic volume The raw dataset contains hourly Interstate 94 Westbound traffic volume for MN DoT ATR station 301, roughly midway between Minneapolis and St Paul, MN, obtained from MN Department of Transportation (UCI). We convert it to de-seasoned daily data with length 1860 $(1400 + 200 + 259)$.

Tree ring Dataset contains 4351 $(2500 + 1000 + 850)$ tree ring measures of a pine from Indian Garden, Nevada Gt Basin obtained from R package `tsdl` (tsdl).

We visualize the long memory in the datasets via autocorrelation plots (Figure 3), where $r_0 = 1$ is cropped-off. Full ACF plots are provided in the supplementary material.

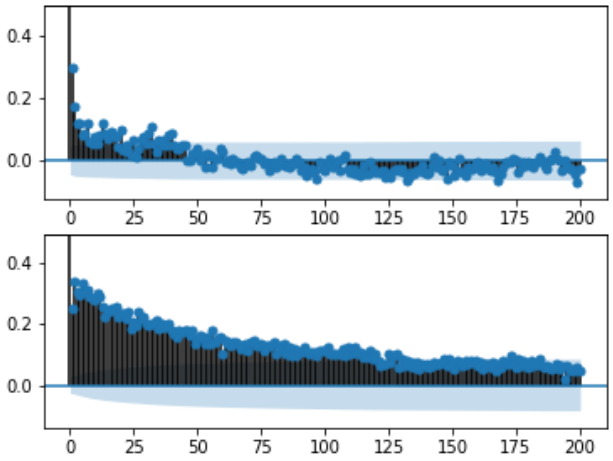

Figure 3. Autocorrelation plot of traffic dataset (top) and DJI dataset (bottom).

We compare the following models: 0. ARFIMA; 1. vanilla

Table 2. Overall performance in terms of RMSE. Average RMSE and the standard deviation (in brackets) are reported. The best result is highlighted in **bold**.

	ARFIMA	DJI (x100)	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)
MRNNF	1.1010 (0.1000)	**0.2472** (0.0109)	**333.36** (8.453)	0.2822 (0.0048)
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)
MLSTMF	1.1580 (0.1660)	0.2540 (0.0139)	337.78 (9.020)	0.2859 (0.0082)
MLSTM	1.1490 (0.1660)	0.2531 (0.0130)	337.83 (9.440)	0.2859 (0.0083)

RNN (RNN); 2. two-lane RNN with past K values as input (RNN2); 3. Recurrent weighted average network (RWA); 4. MIxed hiSTory RNNs (MIST); 5. MRNN with homogeneous memory parameter d (MRNNF); 6. MRNN with dynamic $d^{(t)}$ (MRNN); 7. vanilla LSTM (LSTM); 8. MLSTM with homogeneous d (MLSTMF); and 9. MLSTM with dynamic $d^{(t)}$ (MLSTM).

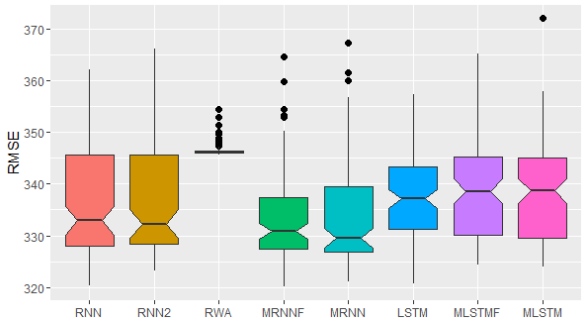

Figure 4. Boxplot of RMSE for 100 different initializations. Dataset: traffic.

In Table 2, we report overall performance of one-step forecasting regarding RMSE. MAE and MAPE are reported in the supplementary material. Boxplots are generated to give a better picture for comparison. We use the traffic dataset as an example and put boxplots for other datasets into the supplementary material. We can see that MRNN and MRNNF have a smaller average RMSE and smaller quantiles compared with others. MLSTM(F) does not have an obvious advantage over LSTM in terms of RMSE, and we suspect that this is due to the difficulty in training MLSTM(F). RWA is very stable with respect to initialization but does not have a competitive RMSE. The `arfima` routine in `R` automatically searches for an optimal global solution, and thus ARFIMA appears only in the comparison of the best performance.

Table 3. Best performance in terms of RMSE.

	ARFIMA	DJI (x100)	Traffic	Tree
ARFIMA	1.0260	0.2468	327.47	0.2773
RNN	1.0452	**0.2390**	320.29	0.2786
RNN2	1.0232	0.2402	323.15	0.2784
RWA	1.6742	0.2631	345.58	0.3047
MIST	1.0232	0.2401	337.49	0.2772
MRNNF	1.0230	0.2394	**320.09**	**0.2769**
MRNN	**1.0208**	0.2395	321.03	0.2770
LSTM	1.0272	0.2396	320.79	0.2771
MLSTMF	1.0280	0.2413	324.37	0.2773
MLSTM	1.0272	0.2412	324.00	0.2772

Thanks to the anonymous reviewers' comments, we present two-sample t-tests to compare the models more rigorously. Consider the null and alternative hypotheses H_0: mean(RMSE(Model)) $\geq$ mean(RMSE(Benchmark)) vs. H_1: mean(RMSE(Model)) $<$ mean(RMSE(Benchmark)). MRNN is significantly better than RNN at 5% significance level on all datasets, and it is significantly better than LSTM on all datasets except for DJI.

In Table 3, we report best performance of one-step forecasting regarding RMSE. Results in terms of MAE and MAPE are reported in the supplementary material. The best performance of MRNNF and MRNN are better than others on ARFIMA, traffic and tree, while remains competent on DJI.

4.2. Short Memory Dataset

For datasets without long memory effect or with long memory only in certain dimensions, the performance of our proposed models does not deteriorate. This claim is supported by an experiment on a synthetic dataset generated by RNN.

We generated a sequence of length 4001 (2000+1200+800) using model (10), which does not have long memory according to Corollary 1. We refer to this synthetic dataset as the RNN dataset. The boxplots of error measures are presented in the supplementary material. From the boxplots, we can see that the performance of MRNN(F) and MLSTM(F) is comparable with that of the true model RNN, except that the variation of the error measures is a bit larger.

4.3. Model Parameter K

We further explore more choices of K using the long memory datasets in section 4.1. Settings for MSLTM and LSTM are kept the same except for the hyperparameter K. We

Table 4. Overall performance on CMU-MOSI in terms of MAE, RMSE and MAPE.

	MAE	RMSE	MAPE
RNN	1.5028 (0.0186)	1.7368 (0.0171)	1.0314 (0.0339)
LSTM	1.4978 (0.0128)	1.7288 (0.0112)	**1.0146** (0.0186)
MRNNF50	**1.4953** (0.0216)	**1.7255** (0.0109)	1.0322 (0.0351)
MLSTMF50	1.4972 (0.0108)	1.7279 (0.0105)	1.0156 (0.0110)

compare the prediction performance of the proposed models with $K = 25, 50, 75$ or 100. For MRNN and MRNNF, the prediction is generally better for a larger K, and they have smaller average RMSE than all the baseline models regardless of the choice of K. Interestingly, the performance of MLSTM and MLSTMF gets better when K becomes smaller, and with $K = 25$, they can outperform LSTM on ARFIMA and traffic datasets. Thus, we recommend a large K for MRNN and MRNNF models, while for the more complicated MLSTM models, K deserves more investigation to balance expressiveness and optimization. More details and figures can be found in the supplementary material.

4.4. Sentiment Analysis

As suggested by the reviewers, we present two more applications of our proposed model on natural language processing tasks. Comparisons between our proposed model and RNN/LSTM are made on two sentiment analysis datasets, CMU-MOSI (Zadeh et al., 2016a;b; 2018) and a paper reviews dataset (Keith et al., 2017). For faster computation, we fix the memory parameter d to be homogeneous and decrease K to 50 in MRNN and MLSTM.

CMU-MOSI contains acoustic, language and visual information from videos. Each sample in CMU-MOSI is annotated with a value ranging from -3 to 3. A larger annotation indicates more positive sentiment. The models are all trained using the MAE loss, and the overall performance is reported in Table 4. We conduct the same two-sample t-tests for MRNNF50 against RNN/LSTM using measure MAE, and the p-values are 0.004 and 0.156. Corresponding p-values for MLSTMF50 are 0.005 and 0.349.

The Paper Reviews dataset (Keith et al., 2017) contains 405 textual reviews evaluated with a 5-point scale. In preprocessing, we removed empty reviews and English reviews, leading to 382 remaining instances. For simplicity, we use a 2-layer network structure with a fully connected classifier at the output. The first layer uses RNN, LSTM, MRNNF50 or MLSTMF50, and the 2nd layer is fixed to be LSTM.

The overall performance is reported in Table 5 and the best performance is reported in Table 6. Using MLSTMF50 as the first layer leads to a significant improvement in all measures over LSTM. For the best performance, MRNNF50 achieves the highest accuracy, while MLSTMF50 has clear-cut advantages on all other metrics. Considering accuracy, the p-values for MLSTMF50 against RNN/LSTM are < 0.001 and 0.040, and that for MRNNF are < 0.001 and 0.105. Our proposed network component can be combined with existing ones to improve performance.

Table 5. Overall performance on Paper Reviews in terms of accuracy, precision, recall and cross-entropy loss (CEloss).

	Accuracy	Precision	Recall	CEloss
RNN	0.2836 (0.0348)	0.1786 (0.0606)	0.2248 (0.0350)	1.5787 (0.0348)
LSTM	0.3021 (0.0468)	0.1724 (0.0697)	0.2274 (0.0332)	1.5752 (0.0189)
MRNNF50	0.3096 (0.0373)	0.1692 (0.0839)	0.2224 (0.0428)	1.5704 (0.0328)
MLSTMF50	**0.3110** (0.0204)	**0.2254** (0.0707)	**0.2594** (0.0262)	**1.4758** (0.0218)

Table 6. Best performance of the models on Paper Reviews.

	Accuracy	Precision	Recall	CEloss
RNN	0.3600	0.3951	0.3093	1.5204
LSTM	0.3800	0.4304	0.3225	1.5512
MRNNF50	**0.4000**	0.3992	0.3178	1.5209
MLSTMF50	0.3600	**0.4621**	**0.3596**	**1.4489**

5. Conclusion

This paper first proves that RNN and LSTM do not have long memory from a time series perspective. By getting use of fractionally integrated processes, we propose the corresponding modifications such that they can handle the long-range dependence data. MRNN and MRNNF are shown to have advantages in forecasting time series with long-term dependency, and a combination of MLSTMF50 and LSTM layers significantly improves over a pure LSTM network on a paper reviews dataset.

In terms of future work, it is interesting to know whether the memory filter can bring similar advantages to other variants of recurrent networks or feed-forward networks for sequence modeling. Moreover, MRNN and MLSTM with dynamic d is time consuming compared with other models, we leave model simplification and exploring faster optimization approaches to future work. In term of filter design, by Definition 3, many other slow decaying patterns can also be explored to model long memory sequences. For example, we may let $w_j(d) = j^{-d-1}$ directly. Last but not least, currently we only learn a best filter, and it is inspiring to extract long memory via filter banks as in signal processing.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This project is sponsored by Huawei Innovation Research Program (HIRP).

References

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

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

Bouri, E., Gil-Alana, L. A., Gupta, R., and Roubaud, D. Modelling long memory volatility in the bitcoin market: Evidence of persistence and structural breaks. *International Journal of Finance & Economics*, 24(1):412–426, 2019.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 77–87, 2017.

Cheng, J., Dong, L., and Lapata, M. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Davidson, J. Moment and memory properties of linear conditional heteroscedasticity models, and a new model. *Journal of Business & Economic Statistics*, 22(1):16–29, 2004.

Ding, Z., Granger, C. W., and Engle, R. F. A long memory property of stock market returns and a new model. *Journal of Empirical Finance*, 1:83–106, 1993.

DiPietro, R., Rupprecht, C., Navab, N., and Hager, G. D. Analyzing and exploiting narx recurrent neural networks for long-term dependencies. *arXiv preprint arXiv:1702.07805*, 2017.

El Hihi, S. and Bengio, Y. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in Neural Information Processing Systems*, pp. 493–499, 1996.

Grange, C. and Joyeux, R. An introduction to long-range time series models and fractional differencing. *Journal of Time Series Analysis*, 4:221–238, 1980.

Graves, A., Wayne, G., and Danihelka, I. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

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

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

Hosking, J. Fractional differencing. *Biometrika*, 68:165–176, 1981.

Keith, B., Fuentes, E., and Meneses, C. A hybrid approach for sentiment analysis applied to paper. In *Proceedings of ACM SIGKDD Conference, Halifax, Nova Scotia, Canada*, pp. 10, 2017.

Keren, G. and Schuller, B. Convolutional rnn: an enhanced model for extracting features from sequential data. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 3412–3419. IEEE, 2016.

Levy, O., Lee, K., FitzGerald, N., and Zettlemoyer, L. Long short-term memory as a dynamically computed element-wise weighted sum. *ACL*, 2018.

Li, M., Li, W. K., and Li, G. A new hyperbolic garch model. *Journal of Econometrics*, 189(2):428–436, 2015.

Li, R., Wu, Z., Ning, Y., Sun, L., Meng, H., and Cai, L. Spectro-temporal modelling with time-frequency lstm and structured output layer for voice conversion. In *Interspeech*.

Lin, T., Horne, B. G., Tino, P., and Giles, C. L. Learning long-term dependencies in narx recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, 1996.

Meyn, S. P. and Tweedie, R. L. *Markov Chains and Stochastic Stability*. Springer Science & Business Media, 2012.

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

Musunuru, N. et al. Modeling long range dependence in wheat food price returns. *International Journal of Economics and Finance*, 11(9):1–46, 2019.

Olah, C. Understanding lstm networks, 2015. URL https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Oliva, J. B., Póczos, B., and Schneider, J. The statistical recurrent unit. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2671–2680. JMLR. org, 2017.

Ostmeyer, J. and Cowell, L. Machine learning on sequential data using a recurrent weighted average. *Neurocomputing*, 331:281–288, 2019.

Radwan, A. G., Soliman, A. M., and Elwakil, A. S. First-order filters generalized to the fractional domain. *Journal of Circuits, Systems, and Computers*, 17(01):55–66, 2008.

Radwan, A. G., Elwakil, A. S., and Soliman, A. M. On the generalization of second-order filters to the fractional-order domain. *Journal of Circuits, Systems, and Computers*, 18(02):361–386, 2009a.

Radwan, A. G., Soliman, A., Elwakil, A. S., and Sedeek, A. On the stability of linear systems with fractional-order elements. *Chaos, Solitons & Fractals*, 40(5):2317–2328, 2009b.

Sabzikar, F., McLeod, A. I., and Meerschaert, M. M. Parameter estimation for artfima time series. *Journal of Statistical Planning and Inference*, 200:129–145, 2019.

Stepleton, T., Pascanu, R., Dabney, W., Jayakumar, S. M., Soyer, H., and Munos, R. Low-pass recurrent neural networks-a memory architecture for longer-term correlation discovery. *arXiv preprint arXiv:1805.04955*, 2018.

Torre, K., Delignieres, D., and Lemoine, L. Detection of long-range dependence and estimation of fractal exponents through arfima modelling. *British Journal of Mathematical and Statistical Psychology*, 60(1):85–106, 2007.

tsdl. tsdl: Time Series Data Library. https://pkg.yangzhuoranyang.com/tsdl/, 2018. Accessed: 2019-01-14.

UCI. UCI Machine Learning Repository metro interstate traffic volume data set. https://archive.ics.uci.edu/ml/datasets/Metro+Interstate+Traffic+Volume, 2019. Accessed: 2019-12-28.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Yu, R., Zheng, S., Anandkumar, A., and Yue, Y. Long-term forecasting using tensor-train rnns. *CoRR*, abs/1711.00073, 2017. URL http://arxiv.org/abs/1711.00073.

Zadeh, A., Zellers, R., Pincus, E., and Morency, L.-P. Mosi: multimodal corpus of sentiment intensity and subjectivity analysis in online opinion videos. *arXiv preprint arXiv:1606.06259*, 2016a.

Zadeh, A., Zellers, R., Pincus, E., and Morency, L.-P. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intelligent Systems*, 31(6):82–88, 2016b.

Zadeh, A., Liang, P. P., Poria, S., Vij, P., Cambria, E., and Morency, L.-P. Multi-attention recurrent network for human communication comprehension. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhang, S., Wu, Y., Che, T., Lin, Z., Memisevic, R., Salakhutdinov, R. R., and Bengio, Y. Architectural complexity measures of recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1822–1830, 2016.

Zhu, Q., Zheng, Y., and Li, G. Linear double autoregression. *Journal of Econometrics*, 207(1):162–174, 2018.