Physica D 404 (2020) 132306

Contents lists available at ScienceDirect

Physica D

journal homepage: www.elsevier.com/locate/physd

Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network

Alex Sherstinsky

ARTICLE INFO

Article history:
Received 13 September 2019
Received in revised form 3 December 2019
Accepted 18 December 2019
Available online 21 January 2020
Communicated by B. Hamzi

Keywords:
RNN
RNN unfolding/unrolling
LSTM
External input gate
Convolutional input context windows

ABSTRACT

Because of their effectiveness in broad practical applications, LSTM networks have received a wealth of coverage in scientific journals, technical blogs, and implementation guides. However, in most articles, the inference formulas for the LSTM network and its parent, RNN, are stated axiomatically, while the training formulas are omitted altogether. In addition, the technique of "unrolling" an RNN is routinely presented without justification throughout the literature. The goal of this tutorial is to explain the essential RNN and LSTM fundamentals in a single document. Drawing from concepts in Signal Processing, we formally derive the canonical RNN formulation from differential equations. We then propose and prove a precise statement, which yields the RNN unrolling technique. We also review the difficulties with training the standard RNN and address them by transforming the RNN into the "Vanilla LSTM"[1] network through a series of logical arguments. We provide all equations pertaining to the LSTM system together with detailed descriptions of its constituent entities. Albeit unconventional, our choice of notation and the method for presenting the LSTM system emphasizes ease of understanding. As part of the analysis, we identify new opportunities to enrich the LSTM system and incorporate these extensions into the Vanilla LSTM network, producing the most general LSTM variant to date. The target reader has already been exposed to RNNs and LSTM networks through numerous available resources and is open to an alternative pedagogical approach. A Machine Learning practitioner seeking guidance for implementing our new augmented LSTM model in software for experimentation and research will find the insights and derivations in this treatise valuable as well.

1. Introduction

Since the original 1997 LSTM paper [1], numerous theoretical and experimental works have been published on the subject of this type of an RNN, many of them reporting on the astounding results achieved across a wide variety of application domains where data is sequential. The impact of the LSTM network has been notable in language modeling, speech-to-text transcription, machine translation, and other applications [2]. Inspired by the impressive benchmarks reported in the literature, some readers in academic and industrial settings decide to learn about the Long Short-Term Memory network (henceforth, "the LSTM network") in order to gauge its applicability to their own research or practical use-case. All major open source machine learning frameworks offer efficient, production-ready implementations of a number of RNN and LSTM network architectures. Naturally, some practitioners, even if new to the RNN/LSTM systems, take advantage of this access and cost-effectiveness and proceed straight to development and experimentation. Others seek to understand every aspect of the operation of this elegant and effective system in greater depth. The advantage of this lengthier path is that it affords an opportunity to build a certain degree of intuition that can prove beneficial during all phases of the process of incorporating an open source module to suit the needs of their research effort or a business application, preparing the dataset, troubleshooting, and tuning.

In a common scenario, this undertaking balloons into reading numerous papers, blog posts, and implementation guides in search of an "A through Z" understanding of the key principles and functions of the system, only to find out that, unfortunately, most of the resources leave one or more of the key questions about the basics unanswered. For example, the Recurrent Neural Network (RNN), which is the general class of a neural network that is the predecessor to and includes the LSTM network as a special case, is routinely simply stated without precedent, and unrolling is presented without justification. Moreover, the training equations are often omitted altogether, leaving the reader puzzled and searching for more resources, while having to reconcile disparate notation used therein. Even the most oft-cited and celebrated primers to date have fallen short of providing a

E-mail address: shers@alum.mit.edu.
URL: https://www.linkedin.com/in/alexsherstinsky.

[1] The nickname "Vanilla LSTM" symbolizes this model's flexibility and generality (Greff et al., 2015).

https://doi.org/10.1016/j.physd.2019.132306
0167-2789/

comprehensive introduction. The combination of descriptions and colorful diagrams alone is not actionable, if the architecture description is incomplete, or if important components and formulas are absent, or if certain core concepts are left unexplained.

As of the timeframe of this writing, a single self-contained primer that provides a clear and concise explanation of the Vanilla LSTM computational cell with well-labeled and logically composed schematics that go hand-in-hand with the formulas is still lacking. The present work is motivated by the conviction that a unifying reference, conveying the basic theory underlying the RNN and the LSTM network, will benefit the Machine Learning (ML) community.

The present article is an attempt to fill in this gap, aiming to serve as the introductory text that the future students and practitioners of RNN and LSTM network can rely upon for learning all the basics pertaining to this rich system. With the emphasis on using a consistent and meaningful notation to explain the facts and the fundamentals (while removing mystery and dispelling the myths), this backgrounder is for those inquisitive researchers and practitioners who not only want to know "how", but also to understand "why".

We focus on the RNN first, because the LSTM network is a type of an RNN, and since the RNN is a simpler system, the intuition gained by analyzing the RNN applies to the LSTM network as well. Importantly, the canonical RNN equations, which we derive from differential equations, serve as the starting model that stipulates a perspicuous logical path toward ultimately arriving at the LSTM system architecture.

The reason for taking the path of deriving the canonical RNN equations from differential equations is that even though RNNs are expressed as difference equations, differential equations have been indispensable for modeling neural networks and continue making a profound impact on solving practical data processing tasks with machine learning methods. On one hand, leveraging the established mathematical theories from differential equations in the continuous-time domain has historically led to a better understanding of the evolution of the related difference equations, since the difference equations are obtained from the corresponding original differential equations through discretization of the differential operators acting on the underlying functions [3–10]. On the other hand, considering the existing deep neurally-inspired architectures as the numerical methods for solving their respective differential equations aided by the recent advances in memory-efficient implementations has helped to successfully stabilize very large models at lower computational costs compared to their original versions [11–13]. Moreover, differential equations defined on the continuous time domain are a more natural fit for modeling certain real-life scenarios than the difference equations defined over the domain of evenly-discretized time intervals [14,15].

Our primary aspiration for this document, particularly for the sections devoted to the Vanilla LSTM system and its extensions, is to fulfill all of the following requirements:

(1) Intuitive — the notation and semantics of variables must be descriptive, explicitly and unambiguously mapping to their respective purposes in the system.
(2) Complete — the explanations and derivations must include both the inference equations ("forward pass" or "normal operation") and the training equations ("backward pass"), and account for all components of the system.
(3) General — the treatment must concern the most inclusive form of the LSTM system (i.e., the "Vanilla LSTM"), specifically including the influence of the cell's state on control nodes ("pinhole connections").
(4) Illustrative — the description must include a complete and clearly labeled cell diagram as well as the sequence diagram, leaving nothing to imagination or guessing (i.e., the imperative is: strive to minimize cognitive strain, do not leave anything as an "exercise for the reader" — everything should be explained and made explicit).
(5) Modular — the system must be described in such a way that the LSTM cell can be readily included as part of a pluggable architecture, both horizontally ("deep sequence") and vertically ("deep representation").
(6) Vector notation — the equations should be expressed in the matrix and vector form; it should be straightforward to plug the equations into a matrix software library (such as `numpy`) as written, instead of having to iterate through indices.

In all sources to date, one or more of the elements in the above list is not addressed[2] [17–37]. Hence, to serve as a comprehensive introduction, the present tutorial captures all the essential details. The practice of using a succinct vector notation and meaningful variable names as well as including the intermediate steps in formulas is designed to build intuition and make derivations easy to follow.

The rest of this document is organized as follows. Section 2 gives a principled background behind RNN systems. Then Section 3 formally arrives at RNN unrolling by proving a precise statement concerning approximating long sequences by a series of shorter, independent sub-sequences (segments). Section 4 presents the RNN training mechanism based on the technique, known as "Back Propagation Through Time", and explores the numerical difficulties, which occur when training on long sequences. To remedy these problems, Section 5 methodically constructs the Vanilla LSTM cell from the canonical RNN system (derived in Section 2) by reasoning through the ways of making RNN more robust. Section 6 provides a detailed explanation of all aspects of the Vanilla LSTM cell. Even though this section is intended to be self-contained, familiarity with the material covered in the preceding sections will be beneficial. The Augmented LSTM system, which embellishes the Vanilla LSTM system with the new computational components, identified as part of the exercise of transforming the RNN to the LSTM network, is presented in Section 7. Section 8 summarizes the covered topics and proposes future projects.

2. The roots of RNN

In this section, we will derive the Recurrent Neural Network (RNN) from differential equations [9,10]. Let $\vec{s}(t)$ be the value of the d-dimensional state signal vector and consider the general nonlinear first-order non-homogeneous ordinary differential equation, which describes the evolution of the state signal as a function of time, t:

$$\frac{d\vec{s}(t)}{dt} = \vec{f}(t) + \vec{\phi} \tag{1}$$

where $\vec{f}(t)$ is a d-dimensional vector-valued function of time, $t \in \mathbb{R}^+$, and $\vec{\phi}$ is a constant d-dimensional vector.

One canonical form of $\vec{f}(t)$ is:

$$\vec{f}(t) = \vec{h}\left(\vec{s}(t), \vec{x}(t)\right) \tag{2}$$

where $\vec{x}(t)$ is the d-dimensional input signal vector and $\vec{h}\left(\vec{s}(t), \vec{x}(t)\right)$ is a vector-valued function of vector-valued arguments.

[2] An article co-authored by one of the LSTM inventors provides a self-contained summary of the embodiment of an RNN, though not at an introductory level [16].

The resulting system,

$$\frac{d\vec{s}(t)}{dt} = \vec{h}\left(\vec{s}(t), \vec{x}(t)\right) + \vec{\phi} \tag{3}$$

comes up in many situations in physics, chemistry, biology, and engineering [38,39].

In certain cases, one starts with s and x as entirely "analog" quantities (i.e., functions not only of time, t, but also of another independent continuous variable, $\vec{\xi}$, denoting the coordinates in multi-dimensional space). Using this notation, the intensity of an input video signal displayed on a flat 2-dimensional screen would be represented as $x(\vec{\xi}, t)$ with $\vec{\xi} \in \mathbb{R}^2$. Sampling $x(\vec{\xi}, t)$ on a uniform 2-dimensional grid converts this signal to the representation $x(\vec{i}, t)$, where $\vec{i}$ is now a discrete 2-dimensional index. Finally, assembling the values of $x(\vec{i}, t)$ for all permutations of the components of the index, $\vec{i}$, into a column vector, produces $\vec{x}(t)$ as originally presented in Eq. (3) above.

One special case of $\vec{f}(t)$ in Eq. (2) is:

$$\vec{f}(t) = \vec{a}(t) + \vec{b}(t) + \vec{c}(t) \tag{4}$$

whose constituent terms, $\vec{a}(t)$, $\vec{b}(t)$, and $\vec{c}(t)$, are d-dimensional vector-valued functions of time, t. Eq. (4) is called the "Additive Model" in Brain Dynamics research literature, because it adds the terms, possibly nonlinear, that determine the rate of change of neuronal activities, or potentials, $\vec{s}(t)$. As a cornerstone of neural network research, the abstract form of the Additive Model in Eq. (4) has been particularized in many ways, including incorporating the effects of delays, imposing "shunting" (or "saturating") bounds on the state of the system, and other factors. Biologically motivated uses of the Additive Model span computational analyses of vision, decision making, reinforcement learning, sensory-motor control, short-term and long-term memory, and the learning of temporal order in language and speech [40]. It has also been noted that the Additive Model generalizes the Hopfield model [41], which, while rooted in biological plausibility, has been influential in physics and engineering [40,42]. In fact, a simplified and discretized form of the Additive Model played a key role in linking the nonlinear dynamical systems governing morphogenesis, one of the fundamental aspects of developmental biology, to a generalized version of the Hopfield network [41], and applying it to an engineering problem in image processing [4,43].

Consider a saturating Additive Model in Eq. (4) with the three constituent terms, $\vec{a}(t)$, $\vec{b}(t)$, and $\vec{c}(t)$, defined as follows:

$$\vec{a}(t) = \sum_{k=0}^{K_s-1} \vec{a}_k(\vec{s}(t - \tau_s(k))) \tag{5}$$

$$\vec{b}(t) = \sum_{k=0}^{K_r-1} \vec{b}_k(\vec{r}(t - \tau_r(k))) \tag{6}$$

$$\vec{r}(t - \tau_r(k)) = G\left(\vec{s}(t - \tau_r(k))\right) \tag{7}$$

$$\vec{c}(t) = \sum_{k=0}^{K_x-1} \vec{c}_k(\vec{x}(t - \tau_x(k))) \tag{8}$$

where $\vec{r}(t)$, the readout signal vector, is a warped version of the state signal vector, $\vec{s}(t)$. A popular choice for the element-wise nonlinear, saturating, and invertible "warping" (or "activation") function, $G(z)$, is an optionally scaled and/or shifted form of the hyperbolic tangent. Then the resulting system, obtained by substituting Eqs. (5)–(8) into Eq. (4) and inserting into Eq. (1), becomes:

$$\frac{d\vec{s}(t)}{dt} = \sum_{k=0}^{K_s-1} \vec{a}_k(\vec{s}(t - \tau_s(k))) + \sum_{k=0}^{K_r-1} \vec{b}_k(\vec{r}(t - \tau_r(k))) + \sum_{k=0}^{K_x-1} \vec{c}_k(\vec{x}(t - \tau_x(k))) + \vec{\phi} \tag{9}$$

$$\vec{r}(t - \tau_r(k)) = G\left(\vec{s}(t - \tau_r(k))\right) \tag{10}$$

Eq. (9) is a nonlinear ordinary delay differential equation (DDE) with discrete delays. Delay is a common feature of many processes in biology, chemistry, mechanics, physics, ecology, and physiology, among others, whereby the nature of the processes dictates the use of delay equations as the only appropriate means of modeling. In engineering, time delays often arise in feedback loops involving sensors and actuators [44].

Hence, the time rate of change of the state signal in Eq. (9) depends on three main components plus the constant ("bias") term, $\vec{\phi}$. The first ("analog") component, $\sum_{k=0}^{K_s-1} \vec{a}_k(\vec{s}(t - \tau_s(k)))$, is the combination of up to K_s time-shifted (by the delay time constants, $\tau_s(k)$) functions, $\vec{a}_k(\vec{s}(t))$, where the term "analog" underscores the fact that each $\vec{a}_k(\vec{s}(t))$ is a function of the (possibly time-shifted) state signal itself (i.e., not the readout signal, which is the warped version of the state signal). The second component, $\sum_{k=0}^{K_r-1} \vec{b}_k(\vec{r}(t - \tau_r(k)))$, is the combination of up to K_r time-shifted (by the delay time constants, $\tau_r(k)$) functions, $\vec{b}_k(\vec{r}(t))$, of the readout signal, given by Eq. (10), the warped (binary-valued in the extreme) version of the state signal. The third component, $\sum_{k=0}^{K_x-1} \vec{c}_k(\vec{x}(t - \tau_x(k)))$, representing the external input, is composed of the combination of up to K_x time-shifted (by the delay time constants, $\tau_x(k)$) functions, $\vec{c}_k(\vec{x}(t))$, of the input signal.[3]

The rationale behind choosing a form of the hyperbolic tangent as the warping function is that the hyperbolic tangent possesses certain useful properties. On one hand, it is monotonic and negative-symmetric with a quasi-linear region, whose slope can be regulated [45]. On the other hand, it is bipolarly-saturating (i.e., bounded at both the negative and the positive limits of its domain). The quasi-linear mode aides in the design of the system's parameters and in interpreting its behavior in the "small signal" regime (i.e., when $\left\|\vec{s}(t)\right\| \ll 1$). The bipolarly-saturating ("squashing") aspect, along with the proper design of the internal parameters of the functions $\vec{a}_k(\vec{s}(t))$ and $\vec{b}_k(\vec{r}(t))$, helps to keep the state of the system (and, hence, its output) bounded. The dynamic range of the state signals is generally unrestricted, but the readout signals are guaranteed to be bounded, while still carrying the state information with low distortion in the quasi-linear mode of the warping function (the "small signal" regime). If the system, described by Eqs. (9) and (10), is stable, then the state signals are bounded as well [46].

Eq. (9) is a nonlinear ordinary delay differential equation (DDE) with discrete delays. Delay is a common feature of many processes in biology, chemistry, mechanics, physics, ecology, and physiology, among others, whereby the nature of the processes dictates the use of delay equations as the only appropriate means of modeling. In engineering, time delays often arise in feedback loops involving sensors and actuators [44].

The time delay terms on the right hand side of Eq. (9) comprise the "memory" aspects of the system. They enable the quantity holding the instantaneous time rate of change of the state signal, $\frac{d\vec{s}(t)}{dt}$, to incorporate contributions from the state, the readout, and the input signal values, measured at different points in time, relative to the current time, t. Qualitatively, these temporal elements enrich the expressive power of the model by capturing causal and/or contextual information.

In neural networks, the time delay is an intrinsic part of the system and also one of the key factors that determines the dynamics.[4] Much of the pioneering research in recurrent networks

[3] The entire input signal, $\vec{c}(t)$, in Eq. (8) is sometimes referred to as the "external driving force" (or, simply, the "driving force") in physics.

[4] In neural networks, time delay occurs in the interaction between neurons; it is induced by the finite switching speed of the neuron and the communication time between neurons [44,47].

during the 1970s and the 1980s was founded on the premise that neuron processes and interactions could be expressed as systems of coupled DDEs [40,41]. Far from the actual operation of the human brain, based on what was already known at the time, these "neurally inspired" mathematical models have been shown to exhibit sufficiently interesting emerging behaviors for both, advancing the knowledge and solving real-world problems in various practical applications. While the major thrust of research efforts was concerned primarily with continuous-time networks, it was well understood that the learning procedures could be readily adapted to discrete systems, obtained from the original differential equations through sampling. We will also follow the path of sampling and discretization for deriving the RNN equations [10]. Over the span of these two decades, pivotal and lasting contributions were made in the area of training networks containing interneurons[5] with "Error Back Propagation" (or "Back Propagation of Error", or "Back Propagation" for short), a special case of a more general error gradient computation procedure. To accommodate recurrent networks, both continuous-time and discrete-time versions of "Back Propagation Through Time" have been developed on the foundation of Back Propagation and used to train the weights and time delays of these networks to perform a wide variety of tasks [48–53]. We will rely on Back Propagation Through Time for training the systems analyzed in this paper.

The contribution of each term on the right hand side of Eq. (9) to the overall system is qualitatively different from that of the others. The functions, $\vec{a}_k(\vec{s}(t-\tau_s(k)))$, of the ("analog") state signal in the first term have a strong effect on the stability of the system, while the functions, $\vec{b}_k(\vec{r}(t-\tau_r(k)))$, of the (bounded) readout signal in the second term capture most of the interactions that shape the system's long-term behavior. If warranted by the modeling requirements of the biological or physical system and/or of the specific datasets and use-cases in an engineering setting, the explicit inclusion of non-zero delay time constants in these terms provides the necessary weighting flexibility in the temporal domain (e.g., to account for delayed neural interactions) [54]. Thus, the parameters, K_s, K_r, and K_x representing the counts of the functions, $\vec{a}_k(\vec{s}(t-\tau_s(k)))$, $\vec{b}_k(\vec{r}(t-\tau_r(k)))$, and $\vec{c}_k(\vec{x}(t-\tau_x(k)))$, respectively (and the counts of the associated delay time constants, $\tau_s(k)$, $\tau_r(k)$, and $\tau_x(k)$, respectively, of these functions), in the system equations are chosen (or estimated by an iterative procedure) accordingly.

Suppose that $\vec{a}_k(\vec{s}(t-\tau_s(k)))$, $\vec{b}_k(\vec{r}(t-\tau_r(k)))$, and $\vec{c}_k(\vec{x}(t-\tau_x(k)))$ are linear functions of $\vec{s}$, $\vec{r}$, and $\vec{x}$, respectively. Then Eq. (9) becomes a nonlinear DDE with linear (matrix-valued) coefficients:

$$\begin{aligned}\frac{d\vec{s}(t)}{dt} = &\sum_{k=0}^{K_s-1} A_k(\vec{s}(t-\tau_s(k))) + \sum_{k=0}^{K_r-1} B_k(\vec{r}(t-\tau_r(k))) \\ &+ \sum_{k=0}^{K_x-1} C_k(\vec{x}(t-\tau_x(k))) + \vec{\phi}\end{aligned} \tag{11}$$

Furthermore, if the matrices, A_k, B_k, and C_k, are circulant (or block circulant), then the matrix–vector multiplication terms in Eq. (11) can be expressed as convolutions in the space of the elements of $\vec{s}$, $\vec{r}$, $\vec{x}$, and $\vec{\phi}$, each indexed by $\vec{i}$:

$$\begin{aligned}\frac{ds(\vec{i},t)}{dt} = &\sum_{k=0}^{K_s-1} a_k(\vec{i}) * s(\vec{i}, t-\tau_s(k)) + \sum_{k=0}^{K_r-1} b_k(\vec{i}) * r(\vec{i}, t-\tau_r(k)) \\ &+ \sum_{k=0}^{K_x-1} c_k(\vec{i}) * x(\vec{i}, t-\tau_x(k)) + \phi(\vec{i})\end{aligned} \tag{12}$$

The index, $\vec{i}$, is 1-dimensional if the matrices, A_k, B_k, and C_k, are circulant and multi-dimensional if they are block circulant.[6]

The summations of time delayed terms in Eq. (12) represent convolutions in the time domain with finite-sized kernels, consisting of the spatial convolutions $a_k(\vec{i}) * s(\vec{i})$, $b_k(\vec{i}) * r(\vec{i})$, and $c_k(\vec{i}) * x(\vec{i})$ as the coefficients for the three temporal components, respectively. In fact, if the entire data set (e.g., the input data set, $\vec{x}(t)$) is available a priori for all time ahead of the application of Eq. (12), then some of the corresponding time delays (e.g., $\tau_x(k)$) can be negative, thereby allowing the incorporation of "future" information for computing the state of the system at the present time, t. This will become relevant further down in the analysis.

Before proceeding, it is interesting to note that earlier studies linked the nonlinear dynamical system, formalized in Eq. (9) (with $K_s = K_r = K_x = 1$ and all τ_s, τ_r, and τ_x set to zero), to the generalization of a type of neural networks.[7] Specifically the variant, in which the functions $\vec{a}_k(\vec{s}(t))$, $\vec{b}_k(\vec{r}(t))$, and $\vec{c}_k(\vec{x}(t))$ are linear operators as in Eq. (11) (with $K_s = K_r = K_x = 1$ and all τ_s, τ_r, and τ_x set to zero) was shown to include the Continuous Hopfield Network [41] as a special case. Its close relative, in which these operators are further restricted to be convolutional as in Eq. (12) (again, with $K_s = K_r = K_x = 1$ and all τ_s, τ_r, and τ_x set to zero), was shown to include the Cellular Neural Network [55,56] as a special case [4,43,46].

Applying the simplifications:

$$\left.\begin{aligned} K_s &= 1 \\ \tau_s(0) &= 0 \\ A_0 &= A \\ K_r &= 1 \\ \tau_r(0) &= \tau_0 \\ B_0 &= B \\ K_x &= 1 \\ \tau_x(0) &= 0 \\ C_0 &= C \end{aligned}\right\} \tag{13}$$

(some of which will be later relaxed) to Eq. (11) turns it into:

$$\frac{d\vec{s}(t)}{dt} = A\vec{s}(t) + B\vec{r}(t-\tau_0) + C\vec{x}(t) + \vec{\phi} \tag{14}$$

Eq. (11), Eq. (12), and, hence, Eq. (14) are nonlinear first-order non-homogeneous DDEs. A standard numerical technique for evaluating these equations, or, in fact, any embodiments of Eq. (1), is to discretize them in time and compute the values of the input signals and the state signals at each time sample up to the required total duration, thereby performing numerical integration.

Denoting the duration of the sampling time step as $\triangle T$ and the index of the time sample as n in the application of the backward Euler discretization rule[8] to Eq. (14) yields[9]:

$$t = n\triangle T \tag{15}$$

$$\frac{d\vec{s}(t)}{dt} \approx \frac{\vec{s}(n\triangle T + \triangle T) - \vec{s}(n\triangle T)}{\triangle T} \tag{16}$$

[5] This term from neuroanatomy provides a biological motivation for considering networks containing multiple "hidden" layers, essentially what is dubbed "deep networks" and "deep learning" today.

[6] For example, the 2-dimensional shape of $\vec{i}$ is appropriate for image processing tasks.

[7] As mentioned earlier, a more appropriate phrase would be "neurally inspired" networks.

[8] The backward Euler method is a stable discretization rule used for solving ordinary differential equations numerically [57]. A simple way to express this rule is to substitute the forward finite difference formula into the definition of the derivative, relax the requirement $\triangle T \to 0$, and evaluate the function on the right hand side (i.e., the quantity that the derivative is equal to) at time, $t + \triangle T$.

[9] It is straightforward to extend the application of the discretization rule to the full Eq. (11), containing any or all the time delay terms and their corresponding matrix coefficients, without the above simplifications. So there is no loss of generality.

$$A\vec{s}(t) + B\vec{r}(t-\tau_0) + C\vec{x}(t) + \vec{\phi}$$
$$= A\vec{s}(n\triangle T) + B\vec{r}(n\triangle T - \tau_0) + C\vec{x}(n\triangle T) + \vec{\phi} \quad (17)$$
$$A\vec{s}(t+\triangle T) + B\vec{r}(t+\triangle T-\tau_0) + C\vec{x}(t+\triangle T) + \vec{\phi}$$
$$= A\vec{s}(n\triangle T+\triangle T) + B\vec{r}(n\triangle T+\triangle T-\tau_0) + C\vec{x}(n\triangle T+\triangle T) + \vec{\phi} \quad (18)$$
$$\frac{\vec{s}(n\triangle T+\triangle T)-\vec{s}(n\triangle T)}{\triangle T}$$
$$\approx A\vec{s}(n\triangle T+\triangle T) + B\vec{r}(n\triangle T+\triangle T-\tau_0) + C\vec{x}(n\triangle T+\triangle T) + \vec{\phi} \quad (19)$$

Now set the delay, τ_0, equal to the single time step. This can be interpreted as storing the value of the readout signal into memory at every time step to be used in the above equations at the next time step. After a single use, the memory storage can be overwritten with the updated value of the readout signal to be used at the next time step, and so forth.[10] Thus, setting $\tau_0 = \triangle T$ and replacing the approximation sign with an equal sign for convenience in Eq. (19) gives:

$$\frac{\vec{s}(n\triangle T+\triangle T)-\vec{s}(n\triangle T)}{\triangle T} = A\vec{s}(n\triangle T+\triangle T) + B\vec{r}(n\triangle T) + C\vec{x}(n\triangle T+\triangle T) + \vec{\phi} \quad (20)$$
$$\frac{\vec{s}((n+1)\triangle T)-\vec{s}(n\triangle T)}{\triangle T} = A\vec{s}((n+1)\triangle T) + B\vec{r}(n\triangle T) + C\vec{x}((n+1)\triangle T) + \vec{\phi} \quad (21)$$
$$\vec{s}((n+1)\triangle T)-\vec{s}(n\triangle T) = \triangle T\left(A\vec{s}((n+1)\triangle T) + B\vec{r}(n\triangle T) + C\vec{x}((n+1)\triangle T) + \vec{\phi}\right) \quad (22)$$

After performing the discretization, all measurements of time in Eq. (22) become integral multiples of the sampling time step, $\triangle T$. Now, $\triangle T$ can be dropped from the arguments, which leaves the time axis dimensionless. Hence, all the signals are transformed into sequences, whose domain is the discrete index, n, and Eq. (14) turns into a nonlinear first-order non-homogeneous difference equation [58]:

$$\vec{s}[n+1]-\vec{s}[n] = \triangle T\left(A\vec{s}[n+1] + B\vec{r}[n] + C\vec{x}[n+1] + \vec{\phi}\right) \quad (23)$$
$$\vec{s}[n+1] = \vec{s}[n] + \triangle T\left(A\vec{s}[n+1] + B\vec{r}[n] + C\vec{x}[n+1] + \vec{\phi}\right)$$
$$(I-(\triangle T)A)\,\vec{s}[n+1] = \vec{s}[n] + ((\triangle T)B)\,\vec{r}[n] + ((\triangle T)C)\,\vec{x}[n+1] + (\triangle T)\vec{\phi} \quad (24)$$

Defining:

$$W_s = (I-(\triangle T)A)^{-1} \quad (25)$$

and multiplying both sides of Eq. (24) by W_s leads to:

$$\vec{s}[n+1] = W_s\vec{s}[n] + ((\triangle T)W_sB)\,\vec{r}[n] + ((\triangle T)W_sC)\,\vec{x}[n+1] + \left((\triangle T)W_s\vec{\phi}\right)$$

which after shifting the index, n, forward by 1 step becomes:

$$\vec{s}[n] = W_s\vec{s}[n-1] + ((\triangle T)W_sB)\,\vec{r}[n-1] + ((\triangle T)W_sC)\,\vec{x}[n] + \left((\triangle T)W_s\vec{\phi}\right)$$
$$\vec{r}[n] = G(\vec{s}[n]) \quad (26)$$

[10] Again, the additional terms, containing similarly combined time delayed input signals (as will be shown to be beneficial later in this paper) and state signals, can be included in the discretization, relaxing the above simplifications as needed to suit the requirements of the particular problem at hand.

Defining two additional weight matrices and a bias vector,

$$W_r = (\triangle T)W_sB \quad (27)$$
$$W_x = (\triangle T)W_sC \quad (28)$$
$$\vec{\theta}_s = (\triangle T)W_s\vec{\phi} \quad (29)$$

transforms the above system into the canonical Recurrent Neural Network (RNN) form:

$$\vec{s}[n] = W_s\vec{s}[n-1] + W_r\vec{r}[n-1] + W_x\vec{x}[n] + \vec{\theta}_s \quad (30)$$
$$\vec{r}[n] = G(\vec{s}[n]) \quad (31)$$

The RNN formulation in Eq. (30), diagrammed in Fig. 1, will be later logically evolved into the LSTM system. Before that, it is beneficial to introduce the process of "unrolling"[11] and the notion of a "cell" of an RNN. These concepts will be simpler to describe using the standard RNN definition, which is derived next from Eq. (30) based on stability arguments.

For the system in Eq. (30) to be stable, every eigenvalue of $\hat{W} = W_s + W_r$ must lie within the complex-valued unit circle [38,58]. Since there is considerable flexibility in the choice of the elements of A and B to satisfy this requirement, setting $\triangle T = 1$ for simplicity is acceptable. As another simplification, let A be a diagonal matrix with large negative entries (i.e., $a_{ii} \ll 0$) on its main diagonal (thereby practically guaranteeing the stability of Eq. (14)). Then, from Eq. (25), $W_s \approx -A^{-1}$ will be a diagonal matrix with small positive entries, $\frac{1}{|a_{ii}|}$, on its main diagonal, which means that the explicit effect of the state signal's value from memory, $\vec{s}[n-1]$, on the system's trajectory will be negligible (the implicit effect through $\vec{r}[n-1]$ will still be present as long as $\|W_r\| > 0$). Thus, ignoring the first term in Eq. (30), reduces it to the standard RNN definition:

$$\vec{s}[n] = W_r\vec{r}[n-1] + W_x\vec{x}[n] + \vec{\theta}_s \quad (32)$$
$$\vec{r}[n] = G(\vec{s}[n]) \quad (33)$$

From Eq. (32), now only the matrix $\hat{W} \approx W_r \approx -A^{-1}B$ is responsible for the stability of the RNN. Consider the best case scenario, where B is a symmetric matrix ($B = B^T$). With this simplification, the essential matrix for analyzing the stability of Eq. (32) becomes $\tilde{W} = -\left[(V_B^TA^{-1})(V_B\Lambda_B)\right]$, where V_B is the orthogonal matrix of the eigenvectors of B, and Λ_B is the diagonal matrix of the eigenvalues of B (with the individual eigenvalues, λ_i, on the main diagonal of Λ_B). Since A is diagonal and V_B is orthogonal, $\tilde{W}$ is a diagonal matrix with the entries $\mu_i = \frac{\lambda_i}{|a_{ii}|}$ on its main diagonal. These quantities become the eigenvalues of the overall RNN system in Eq. (32) in the "small signal regime" ($\|\vec{s}[n]\| \ll 1$), each adding the mode of $(\mu_i)^n$, multiplied by its corresponding initial condition, to the trajectory of $\vec{s}[n]$. A necessary and sufficient condition for stability is that $0 < \mu_i < 1$, meaning that every eigenvalue, λ_i, of B must satisfy the condition $0 < \lambda_i < |a_{ii}|$. If any μ_i and λ_i fail to satisfy this condition, the system will be unstable, causing the elements of $\vec{r}[n]$ to either oscillate or saturate (i.e., enter the flat regions of the warping nonlinearity) at some value of the index, n.

An alternative to choosing the specific convenient form of A in Eq. (25) would be to (somewhat arbitrarily) treat W_s, W_r, W_x, and θ_s in Eq. (30) as mutually independent parameters and then set $W_s = 0$ to obtain the standard RNN definition (as in Eq. (32)). In this case, the above stability analysis still applies. In particular, the eigenvalues, μ_i, of W_r are subject to the same requirement, $0 < \mu_i < 1$, as a necessary and sufficient condition for stability.

[11] The terms "unrolling" and "unfolding" are used interchangeably in association with RNN systems.

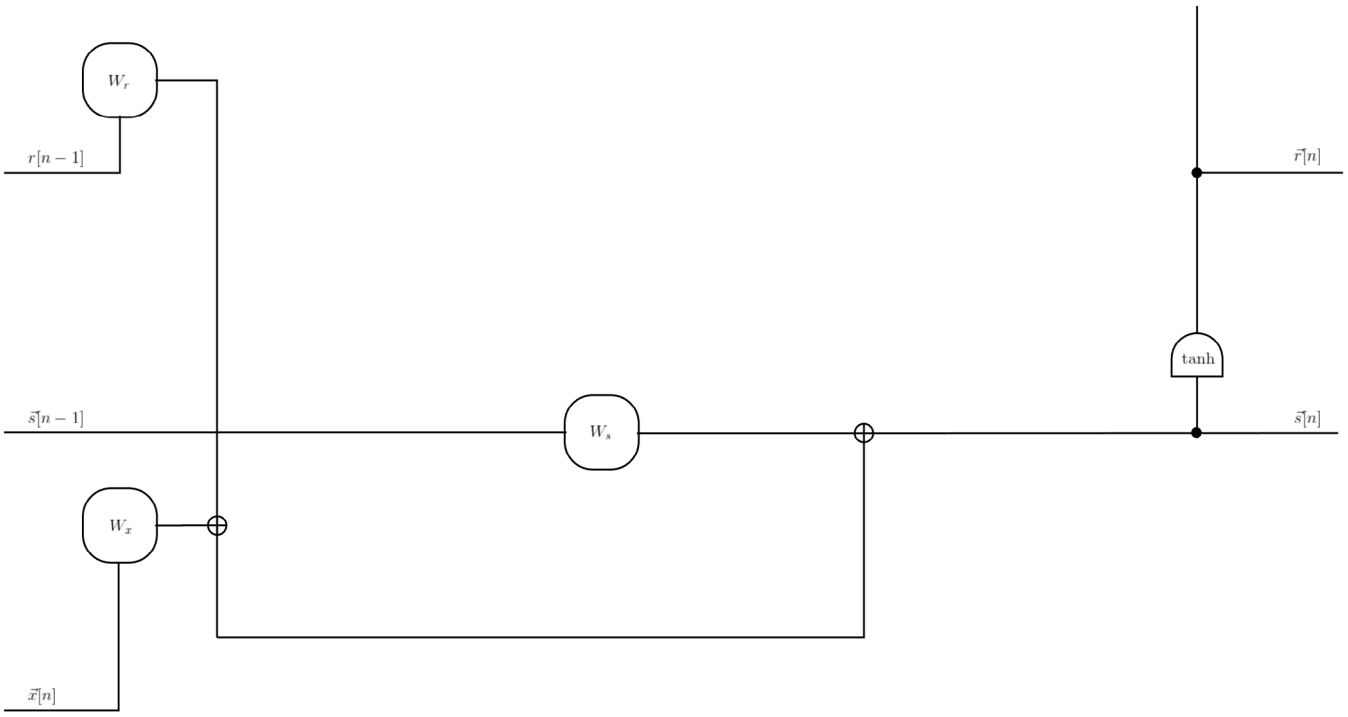


Fig. 1. Canonical RNN cell. The bias parameters, $\vec{\theta}_s$, have been omitted from the figure for brevity. It can be assumed to be included without the loss of generality by appending an additional element, always set to 1, to the input signal vector, $\vec{x}[n]$, and increasing the row dimensions of W_x by 1.

Stability considerations will be later revisited in order to justify the need to evolve the RNN to a more complex system, namely, the LSTM network.

We have shown that the RNN, as expressed by Eq. (30) (in the canonical form) or by Eq. (32) (in the standard form), essentially implements the Backward Euler numerical integration method for the ordinary DDE in Eq. (14). This "forward" direction of starting in the continuous-time domain (differential equation) and ending in the discrete-index domain (difference equation) implies that the original phenomenon being modeled is assumed to be fundamentally analog in nature, and that it is modeled in the discrete domain as an approximation for the purpose of realization. For example, the source signal could be the audio portion of a lecture, recorded on an analog tape (or on digital media as a finely quantized waveform and saved in an audio file). The original recording thus contains the spoken words as well as the intonation, various emphases, and other vocal modulations that communicate the content in the speaker's individual way as expressed through voice. The samples generated by a hypothetical discretization of this phenomenon, governed in this model by Eq. (14), could be captured as the textual transcript of the speech, saved in a document containing only the words uttered by the speaker, but ignoring all the intonation, emotion, and other analogous nuances. In this scenario, it is the sequence of words in the transcript of the lecture that the RNN will be employed to reproduce, not the actual audio recording of the speech. The key subtle point in this scenario is that applying the RNN as a model implies that the underlying phenomenon is governed by Eq. (14), whereby the role of the RNN is that of implementing the computational method for solving this DDE. In contrast, the "reverse" direction would be a more appropriate model in situations where the discrete signal is the natural starting domain, because the phenomenon originates as a sequence of samples. For example, a written essay originates as a sequence of words and punctuation, saved in a document. One can conjure up an analog rendition of this essay as being read by a narrator, giving life to the words and passages with intonation, pauses, and other expressions not present in the original text of the essay. While the starting point depends on how the original source of data is generated, both the continuous ("forward") and the discrete ("reverse") representations can serve as tools for gaining insight into the advantages and the limitations of the models under consideration.

3. RNN unfolding/unrolling

It is convenient to use the term "cell" when referring to Eqs. (30) and (32) in the uninitialized state. In other words, the sequence has been defined by these equations, but its terms not yet computed. Then the cell can be said to be "unfolded" or "unrolled" by specifying the initial conditions on the state signal, $\vec{s}[n]$, and numerically evaluating Eq. (30) or Eq. (32) for a finite range of discrete steps, indexed by n. This process is illustrated in Fig. 2.

Both Eqs. (30) and (32) are recursive in the state signal, $\vec{s}[n]$. Hence, due to the repeated application of the recurrence relation as part of the unrolling, the state signal, $\vec{s}[n]$, at some value of the index, n, no matter how large, encompasses the contributions of the state signal, $\vec{s}[k]$, and the input signal, $\vec{x}[k]$, for all indices, $k < n$, ending at $k = 0$, the start of the sequence [48,52]. Because of this attribute, the RNN belongs to the category of the "Infinite Impulse Response" (IIR) systems.

Define the vector-valued unit step function as:

$$\vec{u}[n] = \begin{cases} \vec{1}, & n \geq 0 \\ \vec{0}, & n < 0 \end{cases} \tag{34}$$

where $\vec{0}$ and $\vec{1}$ denote vectors, all of whose elements are equal to 0 and to 1, respectively. Then the vector-valued unit sample function, $\vec{\delta}[n]$, is defined by being $\vec{1}$ at $n = 0$, and $\vec{0}$ otherwise. In terms of $\vec{u}[n]$,

$$\vec{\delta}[n] = \vec{u}[n] - \vec{u}[n-1] \tag{35}$$

These functions are depicted in Fig. 3.

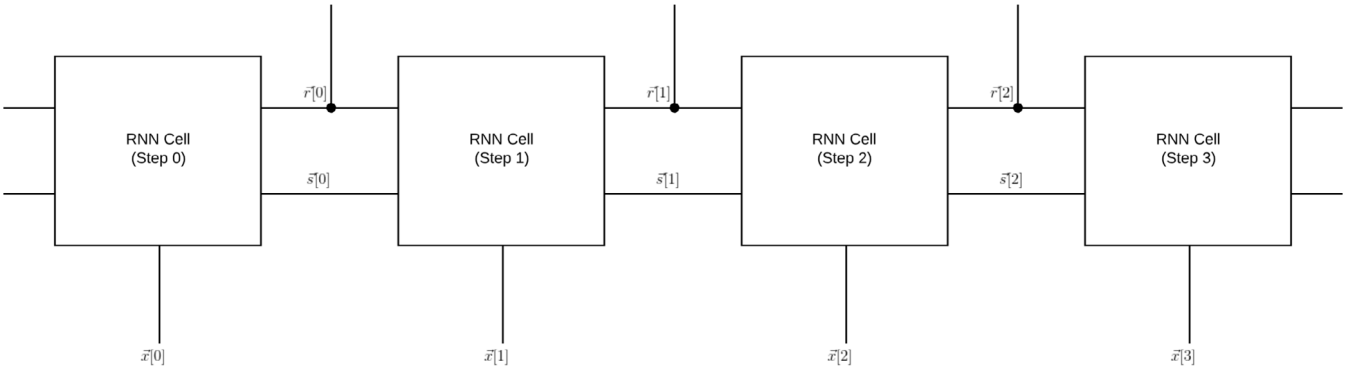
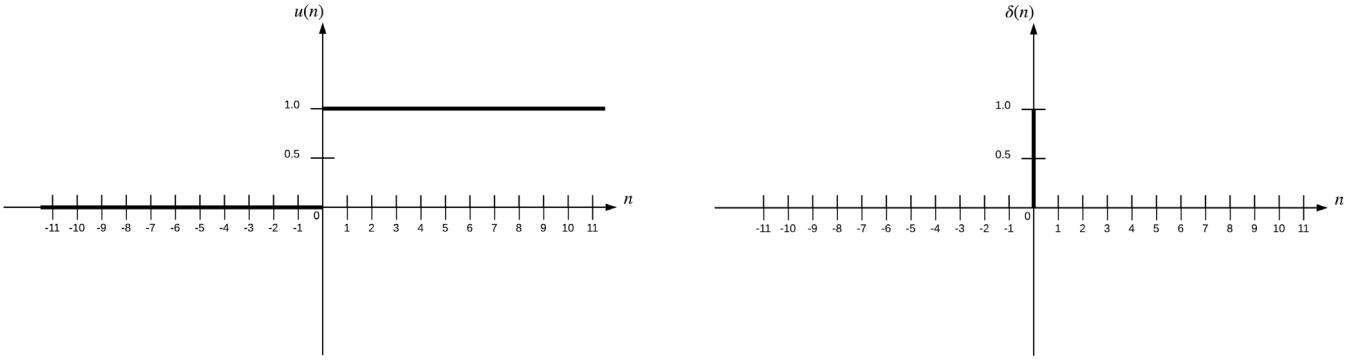


Fig. 2. Sequence of steps generated by unrolling an RNN cell.



(a) The unit step function.

(b) The unit sample function.

Fig. 3. The unit step and the unit sample ("impulse") functions plotted (in one data dimension) against the discrete index, n.

Example 1. The IIR (i.e., unending) nature of the sequences, governed by these equations, can be readily demonstrated by letting $\vec{s}[-1] = \vec{0}$ be the initial condition, setting $\vec{x}[n] = \vec{\delta}[n]$, the unit sample stimulus (i.e., the "impulse"), and computing the response, $\vec{s}[n]$, to this "impulse" for several values of the index, n, in order to try to recognize a pattern. In the case of Eq. (32) with $\vec{\theta}_s = \vec{0}$, the sequence of $\vec{s}[n]$ values will be:

$$\begin{aligned}
&\vec{s}[n=-1] = \vec{0} \\
&\vec{s}[n=0] = W_x\vec{1} \\
&\vec{s}[n=1] = W_r G(W_x\vec{1}) \\
&\vec{s}[n=2] = W_r G(W_r G(W_x\vec{1})) \\
&\vec{s}[n=3] = W_r G(W_r G(W_r G(W_x\vec{1}))) \\
&\vec{s}[n=4] = W_r G(W_r G(W_r G(W_r G(W_x\vec{1})))) \\
&\qquad \cdots
\end{aligned} \tag{36}$$

and so forth. Evidently, it is defined for every positive n, even when the input is only a single impulse at $n = 0$.

In practice, it is desirable to approximate a sequence with an infinite support (IIR), such as Eq. (30) or Eq. (32), by a "Finite Impulse Response" (FIR) sequence. The rationale is that FIR systems have certain advantages over IIR systems. One advantage is guaranteed stability — FIR systems are intrinsically stable. Another advantage is that FIR systems are realizable with finite computational resources. An FIR system will take a finite number of steps to compute the output from the input and will require a finite number of memory locations for storing intermediate results and various coefficients. Moreover, the computational complexity and storage requirements of an FIR system are known at design time.

Denote the sequence of the "ground truth" output values by $\vec{v}[n]$ for any value of the index, n, and let N be the length of the sequence, $\vec{v}[n]$, where N can be an arbitrarily large integer (e.g., the total number of samples in the training set, or the number of inputs presented to the system for inference over the lifetime of the system, etc.). Suppose that $\vec{v}[n]\,\big\rfloor_{0\le n\le N-1}$ is subdivided into M non-overlapping varying-length segments with K_m samples per segment, where every K_m is finite, and $K_m \le N$. It can be assumed that M is an integer with $M \ge 1$ (if needed, the RNN system in Eq. (32) can be "padded" with extra $\vec{x}[n] = \vec{0}$ input terms for this to hold).

Formally, let $\vec{v}[n]\,\big\rfloor_{0\le n\le N-1}$ be the sequence of the ground truth output values for any value of the index, n, and assume that there exists a partitioning of $\vec{v}[n]\,\big\rfloor_{0\le n\le N-1}$ into M non-overlapping segments, $\vec{v}_m[n]$, $0 \le m \le M-1$:

$$\vec{v}[n]\,\big\rfloor_{0\le n\le N-1} = \sum_{m=0}^{M-1} \vec{v}_m[n] \tag{37}$$

For subdividing a sequence into M non-overlapping segments, consider a vector-valued "rectangular" window function, $\vec{w}_0[n]$, which has the value of $\vec{1}$ within the window $0 \le n \le K_0 - 1$ and $\vec{0}$ otherwise. In terms of the vector-valued unit step function, $\vec{u}[n]$, $\vec{w}_0[n]$ is defined as:

$$\vec{w}_0[n] = \vec{u}[n] - \vec{u}[n - K_0] \tag{38}$$

Combining Eq. (35) with Eq. (38) provides an alternative ("sampling") definition of $\vec{w}_0[n]$:

$$\begin{aligned}
\vec{w}_0[n] = \vec{u}[n] - \vec{u}[n-1] \\
+ \vec{u}[n-1] - \vec{u}[n-2] \\
+ \vec{u}[n-2] - \vec{u}[n-3]
\end{aligned}$$

$$
\begin{aligned}
&+ \ldots + \\
&+ \vec{u}[n-(K_0-2)] - \vec{u}[n-(K_0-1)] \\
&+ \vec{u}[n-(K_0-1)] - \vec{u}[n-K_0] \\
&= \sum_{k=0}^{K_0-1} \vec{\delta}[n-k] \qquad (39)
\end{aligned}
$$

Then from Eq. (39), the RNN sequence can be sampled in its entirety by the full N-samples-long window:

$$
\begin{aligned}
\vec{w}[n] &= \vec{u}[n] - \vec{u}[n-N] \\
&= \sum_{m=0}^{M-1} \left(\sum_{k=j(m)}^{j(m)+K_m-1} \vec{\delta}[n-k] \right) \qquad (40) \\
&= \sum_{m=0}^{M-1} \vec{w}_m[n] \qquad (41)
\end{aligned}
$$

where:

$$
j(m) = \begin{cases} \sum_{i=0}^{m-1} K_i, & 1 \le m \le M-1 \\ 0, & m = 0 \end{cases} \qquad (42)
$$

and:

$$
\vec{w}_m[n] = \sum_{k=j(m)}^{j(m)+K_m-1} \vec{\delta}[n-k] \qquad (43)
$$

Under the change of indices,

$$
\begin{aligned}
&l = k - j(m) \\
&k = j(m) + l \\
&l \longrightarrow k
\end{aligned}
$$

Eq. (43) becomes:

$$
\vec{w}_m[n] = \sum_{k=0}^{K_m-1} \vec{\delta}[n-j(m)-k] \qquad (44)
$$

Eq. (44) indicates that each $\vec{w}_m[n]$ is a rectangular window, whose size is K_m samples. Hence, "extracting" a K_m-samples-long segment with the index, m, from the overall ground truth output sequence, $\vec{v}[n]\big\rfloor_{0\le n\le N-1}$, amounts to multiplying this sequence by $\vec{w}_m[n]$:

$$
\begin{aligned}
\vec{v}_m[n]\big\rfloor_{0\le n\le N-1} &= \vec{w}_m[n] \odot \vec{v}[n]\big\rfloor_{0\le n\le N-1} \qquad (45) \\
&= \begin{cases} \vec{v}[n], & j(m) \le n \le j(m)+K_m-1 \\ \vec{0}, & \text{otherwise} \end{cases} \qquad (46)
\end{aligned}
$$

where $j(m)$ is given by Eq. (42). According to Eq. (46), the segment-level ground truth output subsequence, $\vec{v}_m[n]\big\rfloor_{0\le n\le N-1}$, in Eq. (45) will have non-zero values for the given value of the segment index, m, where $0 \le m \le M-1$, only when the index, n, is in the range $j(m) \le n \le j(m)+K_m-1$. This is in agreement with Eq. (37).

Define $\mathcal{Q}(\langle\vec{r}[n]\rangle)$ as an invertible map that transforms an ensemble of the readout signals of the RNN system, $\langle\vec{r}[n]\rangle$, into an ensemble of observable output signals, $\langle\vec{y}[n]\rangle$, for $0 \le n \le N-1$:

$$
\langle\vec{y}[n]\rangle \equiv \mathcal{Q}(\langle\vec{r}[n]\rangle) \qquad (47)
$$

In addition, define $\mathcal{L}$ as an "objective function" (or "merit function" [59]) that measures the cost of the observable output of the system deviating from the desired ground truth output values, given the input data, supplied over the entire range of the values of the index, n:

$$
\mathcal{L}\left(\langle\vec{y}[n]\rangle\big\rfloor_{0\le n\le N-1}, \langle\vec{v}[n]\rangle\big\rfloor_{0\le n\le N-1}\right) \qquad (48)
$$

where $\langle\vec{y}[n]\rangle\big\rfloor_{0\le n\le N-1}$ denotes the ensemble of all N members of the sequence of the observable output variables, $\vec{y}[n]$, and $\langle\vec{v}[n]\rangle\big\rfloor_{0\le n\le N-1}$ denotes the ensemble of all N members of the sequence of the ground truth output values, $\vec{v}[n]$.

As shorthand, combine all parameters of the standard RNN system in Eq. (32) under one symbol, Θ:

$$
\Theta \equiv \left\{W_r, W_x, \vec{\theta}_s\right\} \qquad (49)
$$

Proposition 1. *Given the standard RNN system in Eq. (32) parameterized by Θ, defined in Eq. (49), assume that there exists a value of Θ, at which the objective function, $\mathcal{L}$, defined in Eq. (48) for an N-samples-long sequence, is close to an optimum as measured by some acceptable bound. Further, assume that there exist non-zero finite constants, M and K_m, such that $K_m < N$, where $0 \le m \le M-1$, and that the ground truth output sequence, $\vec{v}[n]\big\rfloor_{0\le n\le N-1}$, can be partitioned into mutually independent segment-level ground truth output subsequences, $\vec{v}_m[n]\big\rfloor_{0\le n\le N-1}$, for different values of the segment index, m, as specified in Eq. (46). Then a single, reusable RNN cell, unrolled for an adjustable number of steps, K_m, is computationally sufficient for seeking Θ that optimizes $\mathcal{L}$ over the training set and for inferring outputs from unseen inputs.*

Proof. The objective function in Eq. (48) computes the error in the system's performance during training, validation, and testing phases as well as tracks its generalization metrics on the actual application data during the inference phase. By the assumption, $\mathcal{L}$ can be optimized. This implies that when $\mathcal{L}$ is acceptably close to an optimum, the observable output ensemble from the RNN system approximates the ground truth output ensemble within a commensurately acceptable tolerance bound:

$$
\langle\vec{y}[n]\rangle\big\rfloor_{0\le n\le N-1} \approx \langle\vec{v}[n]\rangle\big\rfloor_{0\le n\le N-1} \qquad (50)
$$

Segmenting the RNN system's output sequence by the same procedure as was used in Eq. (45) to segment the ground truth output sequence gives:

$$
\begin{aligned}
\vec{y}_m[n]\big\rfloor_{0\le n\le N-1} &= \vec{w}_m[n] \odot \vec{y}[n]\big\rfloor_{0\le n\le N-1} \qquad (51) \\
&= \begin{cases} \vec{y}[n], & j(m) \le n \le j(m)+K_m-1 \\ \vec{0}, & \text{otherwise} \end{cases} \qquad (52)
\end{aligned}
$$

where $j(m)$ is given by Eq. (42). According to Eq. (52), the segment-level output subsequence, $\vec{y}_m[n]\big\rfloor_{0\le n\le N-1}$, in Eq. (51) will have non-zero values for the given value of the segment index, m, where $0 \le m \le M-1$, only when the index, n, is in the range $j(m) \le n \le j(m)+K_m-1$.

By the assumption that the segment-level ensembles of the ground truth output subsequences are mutually independent, the objective function in Eq. (48) is separable and can be expressed as a set of M independent segment-level components, $\left\{\mathcal{L}_m\left(\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1}, \langle\vec{v}_m[n]\rangle\big\rfloor_{0\le n\le N-1}\right)\right\}$, $0 \le m \le M-1$, combined by a suitable function, $\mathcal{C}$:

$$
\begin{aligned}
&\mathcal{L}\left(\langle\vec{y}[n]\rangle\big\rfloor_{0\le n\le N-1}, \langle\vec{v}[n]\rangle\big\rfloor_{0\le n\le N-1}\right) \\
&= \mathcal{C}\left(\left\{\mathcal{L}_m\left(\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1}, \langle\vec{v}_m[n]\rangle\big\rfloor_{0\le n\le N-1}\right)\right\}\big\rfloor_{0\le m\le M-1}\right) \qquad (53)
\end{aligned}
$$

Then by Eqs. (50) and (53),

$$
\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1} \approx \langle\vec{v}_m[n]\rangle\big\rfloor_{0\le n\le N-1} \qquad (54)
$$

for all values of the segment index, m, where $0 \le m \le M-1$. In other words, the tracking of the ground truth output by the observable output of the RNN system at the entire N-sample ensemble level must hold at the K_m-samples-long segment level, too, for all segments.

Since $\mathcal{Q}(\langle\vec{r}[n]\rangle)$ is invertible,

$$\langle\vec{r}_m[n]\rangle\big\rfloor_{0\le n\le N-1} = \mathcal{Q}^{-1}\left(\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1}\right) \tag{55}$$

and since the warping function, $G(z)$, in Eq. (33) is invertible, then for any value of the sample index, n,

$$\vec{s}_m[n]\big\rfloor_{0\le n\le N-1} = G^{-1}\left(\vec{r}_m[n]\big\rfloor_{0\le n\le N-1}\right) \tag{56}$$

According to Eqs. (50), (55), and (56), $\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$, $\langle\vec{r}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$, and $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$ are all functions of random variables. Let $\langle\vec{v}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$ and $\langle\vec{v}_l[n]\rangle\big\rfloor_{0\le n\le N-1}$ be the ground truth output subsequence ensembles, belonging to any two segments, whose indices are m and l, respectively, with $m \neq l$. By the assumption, $\langle\vec{v}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$ and $\langle\vec{v}_l[n]\rangle\big\rfloor_{0\le n\le N-1}$ are independent random variables. Because the functions of independent variables are also independent, it follows that at the segment level the observable output signal subsequence ensembles, $\langle\vec{y}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$ and $\langle\vec{y}_l[n]\rangle\big\rfloor_{0\le n\le N-1}$, are independent, the readout signal subsequence ensembles, $\langle\vec{r}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$ and $\langle\vec{r}_l[n]\rangle\big\rfloor_{0\le n\le N-1}$, are independent, and the state signal subsequences, $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$ and $\vec{s}_l[n]\big\rfloor_{0\le n\le N-1}$, are independent.

The mutual independence of the state signal subsequences, $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$, for different values of the segment index, m, places a restriction on the initial conditions of these subsequences. Specifically, the initial condition for the state signal subsequence of one segment cannot be a function of samples belonging to either the state signal subsequence or the input signal subsequence of another segment for any value of the index, n.

Performing the element-wise multiplication of the input sequence, $\vec{x}[n]\big\rfloor_{0\le n\le N-1}$, by the sampling window, $\vec{w}_m[n]$, extracts a segment-level input sequence with the index, m:

$$\vec{x}_m[n]\big\rfloor_{0\le n\le N-1} = \vec{w}_m[n]\odot\vec{x}[n]\big\rfloor_{0\le n\le N-1} \tag{57}$$

$$= \begin{cases}\vec{x}[n], & j(m)\le n\le j(m)+K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{58}$$

where $j(m)$ is given by Eq. (42). According to Eq. (58), the segment-level input subsequence, $\vec{x}_m[n]\big\rfloor_{0\le n\le N-1}$, in Eq. (57) will have non-zero values for the given value of the segment index, m, where $0\le m\le M-1$, only when the index, n, is in the range $j(m)\le n\le j(m)+K_m-1$.

Due to recursion, the members of the state signal sequence, $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$, in an RNN system can in general depend on the entire input signal sequence assembly. However, since under the present assumptions the segment-level state signal subsequences, $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$ and $\vec{s}_l[n]\big\rfloor_{0\le n\le N-1}$, belonging to different segments, are independent, the dependency of $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$ on the input signal must be limited to the same segment-level subsequence (i.e., with segment index, m). If we define $\mathcal{F}\left(\langle\vec{x}_m[n]\rangle\big\rfloor_{j(m)\le n\le j(m)+K_m-1}\right)$ as a map that transforms the segment-level input signal subsequence assembly, $\langle\vec{x}_m[n]\rangle\big\rfloor_{0\le n\le N-1}$, into the segment-level state signal subsequence assembly, $\langle\vec{s}_m[n]\rangle\big\rfloor_{j(m)\le n\le j(m)+K_m-1}$, then the standard RNN system definition in Eq. (32) at the K_m-samples-long segment level can be expressed as:

$$\langle\vec{s}_m[n]\rangle\big\rfloor_{j(m)\le n\le j(m)+K_m-1} = \mathcal{F}\left(\langle\vec{x}_m[n]\rangle\big\rfloor_{j(m)\le n\le j(m)+K_m-1}\right) \tag{59}$$

Hence, for any $0\le m,l\le M-1$ with $m\neq l$, the restriction,

$$\vec{s}_m[n=j(m)-1]\perp\begin{cases}\langle\vec{s}_l[n]\rangle, & j(l)\le n\le j(l)+K_l-1\\ \langle\vec{x}_l[n]\rangle, & j(l)\le n\le j(l)+K_l-1\end{cases} \tag{60}$$

must be enforced in order to satisfy the independence of the segment-level state signal subsequences. The only way to achieve this is to set $\vec{s}_m[n=j(m)-1]$ to a random vector or to $\vec{0}$. The latter choice is adopted here for simplicity.

Thus, substituting Eqs. (58) and (60) into Eq. (32) yields the RNN system equations for an individual segment:

$$\vec{s}_m[n] = \begin{cases}W_r\vec{r}_m[n-1]\\ +W_x\vec{x}_m[n]+\vec{\theta}_s, & j(m)\le n\le j(m)+K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{61}$$

$$\vec{r}_m[n] = \begin{cases}G(\vec{s}_m[n]), & j(m)\le n\le j(m)+K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{62}$$

$$\vec{s}_m[n=j(m)-1]=\vec{0} \tag{63}$$

$$0\le m\le M-1 \tag{64}$$

Making the index substitution,

$$n\longrightarrow n+j(m)$$

shifts the segment-level subsequences, $\vec{r}_m[n]\big\rfloor_{0\le n\le N-1}$, $\vec{s}_m[n]\big\rfloor_{0\le n\le N-1}$, and $\vec{x}_m[n]\big\rfloor_{0\le n\le N-1}$, by $-j(m)$ samples:

$$\begin{aligned}\vec{\tilde{x}}_m[n] &\equiv \vec{x}_m[n+j(m)]\big\rfloor_{0\le n\le N-1}\\ &= \begin{cases}\vec{x}[n+j(m)], & j(m)\le n+j(m)\le j(m)+K_m-1\\ \vec{0}, & \text{otherwise}\end{cases}\\ &= \begin{cases}\vec{x}[n+j(m)], & 0\le n\le K_m-1\\ \vec{0}, & \text{otherwise}\end{cases}\end{aligned} \tag{65}$$

$$\vec{\tilde{r}}_m[n] \equiv \vec{r}_m[n+j(m)]\big\rfloor_{0\le n\le N-1} \tag{66}$$

$$\begin{aligned}\vec{\tilde{s}}_m[n] &\equiv \vec{s}_m[n+j(m)]\big\rfloor_{0\le n\le N-1}\\ &= \begin{cases}W_r\vec{r}_m[n+j(m)-1]\\ +W_x\vec{x}_m[n+j(m)]+\vec{\theta}_s, & j(m)\le n+j(m)\\ & \le j(m)+K_m-1\\ \vec{0}, & \text{otherwise}\end{cases}\\ &= \begin{cases}W_r\vec{\tilde{r}}_m[n-1]+W_x\vec{\tilde{x}}_m[n]+\vec{\theta}_s, & 0\le n\le K_m-1\\ \vec{0}, & \text{otherwise}\end{cases}\end{aligned} \tag{67}$$

$$\vec{\tilde{s}}_m[n=-1]\equiv\vec{s}_m[n+j(m)=j(m)-1]=\vec{0} \tag{68}$$

Simplified, these equations reduce to the form of the standard RNN system, unrolled for K_m steps, for any segment with the index, m, where $0\le m\le M-1$:

$$\vec{\tilde{s}}_m[n=-1]=\vec{0} \tag{69}$$

$$\vec{\tilde{s}}_m[n] = \begin{cases}W_r\vec{\tilde{r}}_m[n-1]+W_x\vec{\tilde{x}}_m[n]+\vec{\theta}_s, & 0\le n\le K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{70}$$

$$\vec{\tilde{r}}_m[n] = \begin{cases}G(\vec{\tilde{s}}_m[n]), & 0\le n\le K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{71}$$

$$\vec{\tilde{x}}_m[n] = \begin{cases}\vec{x}[n+j(m)], & 0\le n\le K_m-1\\ \vec{0}, & \text{otherwise}\end{cases} \tag{72}$$

It follows that the shifted segment-level state signal subsequences, $\vec{\tilde{s}}_m[n]\big\rfloor_{0\le n\le K_m-1}$, for different values of the segment index, m, where $0\le m\le M-1$, are mutually independent. In addition, from Eqs. (71), (70), and (72), the non-zero values of the resulting sequences, $\vec{\tilde{r}}_m[n]\big\rfloor_{0\le n\le K_m-1}$, $\vec{\tilde{s}}_m[n]\big\rfloor_{0\le n\le K_m-1}$, and $\vec{\tilde{x}}_m[n]\big\rfloor_{0\le n\le K_m-1}$, are confined to $0\le n\le K_m-1$ for any value of the segment index, m, where $0\le m\le M-1$.

As the sample index, n, traverses the segment-level range, $0\le n\le K_m-1$, for every segment with the index, m, where $0\le m\le M-1$, the input subsequence, $\vec{\tilde{x}}_m[n]$, takes on all the available

values of the input sequence, $\vec{x}[n]$, segment by segment. Similarly to the original RNN system in Eq. (32), the input signal, $\vec{\tilde{x}}_m[n]$ (the external driving force), is the only independent variable of the RNN system, unrolled for K_m steps, in Eq. (70). Together with the mutual independence of $\vec{\tilde{s}}_m[n]\,\big\rfloor_{0\le n\le K_m-1}$ for different segments, this makes the computations of the RNN system, unrolled for K_m steps, generic for all segments. The only signal that retains the dependence on the segment index, m, is the input. Dropping the segment subscript, m, from the variables representing the state signal and the readout signal results in the following prototype formulation of the RNN system, unrolled for K_m steps:

$$\vec{\tilde{s}}[n=-1]=\vec{0} \tag{73}$$

$$\vec{\tilde{s}}[n]=\begin{cases} W_r\vec{\tilde{r}}[n-1]+W_x\vec{\tilde{x}}_m[n]+\vec{\theta}_s, & 0\le n\le K_m-1 \\ \vec{0}, & \text{otherwise} \end{cases} \tag{74}$$

$$\vec{\tilde{r}}[n]=\begin{cases} G(\vec{\tilde{s}}[n]), & 0\le n\le K_m-1 \\ \vec{0}, & \text{otherwise} \end{cases} \tag{75}$$

$$\vec{\tilde{x}}_m[n]=\begin{cases} \vec{x}[n+j(m)], & 0\le n\le K_m-1 \\ \vec{0}, & \text{otherwise} \end{cases} \tag{76}$$

$$0\le m\le M-1 \tag{77}$$

where $j(m)$ is given by Eq. (42).

The same prototype variable-length RNN computation, unrolled for K_m steps, can process all segments, one at a time. After initializing the segment's state signal using Eq. (73) and selecting the input samples for the segment using Eq. (76), Eqs. (74) and (75) are applied for K_m steps, $0\le n\le K_m-1$. This procedure can then be applied to the next segment using the same computational module, and then to the next segment, and so on, until the inputs comprising all M segments have been processed. Moreover, the mutual independence of the segments facilitates parallelism, whereby the computation of $\vec{\tilde{s}}_m[n]$, $\vec{\tilde{r}}_m[n]$, and $\vec{y}_m[n]$ for $0\le n\le K_m-1$ can be carried out for all M segments concurrently. □

Remark 1. Proposition 1 and its proof do not formally address advanced RNN architectures, such as Gated Recurrent Unit (GRU), Attention Networks, and complex models comprised of multiple LSTM networks.

Remark 2. While the segment-level state signal subsequences, $\vec{\tilde{s}}_m[n]\,\big\rfloor_{0\le n\le K_m-1}$, are independent, there is no independence requirement on the input signal subsequences, $\vec{\tilde{x}}_m[n]$, belonging to the different segments. It has been shown that dependencies in the input signal can be de-correlated by appropriately trained hidden layer weights, W_x, thus maintaining the independence of $\vec{\tilde{s}}_m[n]\,\big\rfloor_{0\le n\le K_m-1}$ [3,60].

Remark 3. It is important to emphasize that no IIR-to-FIR conversion method is optimal in the absolute sense. Finding an optimal FIR approximation to an IIR system can only be done with respect to a certain measure of fidelity and performance. In practice, one must settle for an approximation to the ideal form of the output signal. The success of an approximation technique depends on the degree to which the resulting FIR system can be adapted to fit the specific data distribution and achieve acceptable quality metrics for the given application's requirements.

Unrolling (or unfolding) for a finite number of steps is a standard, straightforward technique for approximating RNNs by FIR sequences. However, due to the truncation inherent in limiting the number of steps, the resulting unfolded RNN model introduces artificial discontinuities in the approximated version of the target output sequence. In general, the more steps are included in the unrolled RNN subsequence, the closer it can get to the desired output samples, but the less efficient the system becomes, due to the increased number of computations. Nevertheless, if the underlying distribution governing the application generates the sequence under consideration as a series of independent segments (subsequences), then by Proposition 1, an unfolded RNN model aligned with each segment can be trained to reproduce outputs from inputs in a way that aims to satisfy the appropriate criteria of merit faithfully. In this sense, Proposition 1 for RNNs loosely resembles in spirit the Sampling Theorem in the field of Discrete-Time Signal Processing [58]. The method of unrolling is also applicable to the scenarios where attention can be restricted to those "present" sections of the output that are influenced to an arbitrarily small extent by the portions of the "past" or the "future" of the input beyond some arbitrarily large but finite step [61]. As a matter of fact, in certain situations, the raw data set may be amenable to pre-processing, without losing much of the essential information. Suppose that after a suitable cleanup, treating the overall sequence as a collection of independent segments becomes a reasonable assumption. Then by Proposition 1, the adverse effects of truncation can be reduced by adjusting the number of samples comprising the window of the unrolled RNN system. Moreover, whenever Proposition 1 applies, the segments can be processed by the unrolled RNN system in any order (because they are assumed to be independent). This flexibility is utilized by the modules that split the original data set into segments and feed the batches of segmented training samples to the computational core of system.

Conversely, if the assumptions of Proposition 1 are violated, then truncating the unrolling will prevent the model from adequately fitting the ground truth output. To illustrate this point, suppose that an RNN system, unrolled for a relatively few steps, is being used to fit the target sequence that exhibits extremely long-range dependencies. The unrolled RNN subsequence will be trained under the erroneous assumptions, expecting the ground truth to be a series of short independent subsequences. However, because of its relatively narrow window, this RNN subsequence will not be able to encompass enough samples to capture the dependencies present in the actual data. Under-sampling the distribution will limit the flow of information from the training samples to the parameters of the model, leaving it in the constant state of making poor predictions. As a symptom, the model will repeatedly encounter unexpected variations during training, causing the objective function to oscillate, never converging to an adequate optimum. During inference, the generated sequence will suffer from severe jitter and distortion when compared to the expected output.

Remark 4. According to Proposition 1, the RNN unrolling technique is justified by partitioning a single output sequence into multiple independent subsequences and placing restrictions on the initialization of the state between subsequences. However, adhering to these conditions may be problematic in terms of modeling sequences in practical applications. Oftentimes, the output subsequences exhibit some inter-dependence and/or the initial state of one subsequence is influenced by the final state of another subsequence. In practice, if the choice for the initial conditions of the state of subsequences is consistent with the process by which the application generates the samples of the input sequence, then a favorable subdivision of the output sequence into acceptably independent subsequences can be found empirically through experimentation and statistical analysis.

4. RNN training difficulties

Proposition 1 establishes that Eqs. (73)–(77) together with Eq. (42) specify the truncated unrolled RNN system that realizes the standard RNN system, given by Eqs. (32) and (33). We now segue to the analysis of the training technique for obtaining the weights in the truncated unrolled RNN system, with the focus on Eqs. (74) and (75).

Once the infinite RNN sequence in Eq. (32) is truncated (or unrolled to a finite length), the resulting system, given in Eq. (74), becomes inherently stable. However, RNN systems are problematic in practice, despite their stability. During training, they suffer from the well-documented issues, known as "vanishing gradients" and "exploding gradients" [1,62,63]. These difficulties become pronounced when the dependencies in the target subsequence span a large number of samples, requiring the window of the unrolled RNN model to be commensurately wide in order to capture these long-range dependencies.

Truncated unrolled RNN systems, such as Eq. (74), are commonly trained using "Back Propagation Through Time" (BPTT), which is the "Back Propagation" technique adapted for sequences [64–67]. The essence of Back Propagation is the repeated application of the chain rule of differentiation. Computationally, the action of unrolling Eq. (32) for K steps amounts to converting its associated directed graph having a delay and a cycle, into a directed acyclic graph (DAG) corresponding to Eq. (74). For this reason, while originally Back Propagation was restricted to feedforward networks only, subsequently, it has been successfully applied to recurrent networks by taking advantage of the very fact that for every recurrent network there exists an equivalent feedforward network with identical behavior for a finite number of steps [68–70].

As a supervised training algorithm, BPTT utilizes the available $\vec{x}_m[n]$ and $\vec{\tilde{r}}[n]$ data pairs (or the respective pairs of some mappings of these quantities) in the training set to compute the parameters of the system, Θ, defined in Eq. (49), so as to optimize an objective function, E, which depends on the readout signal, $\vec{\tilde{r}}[n]$, at one or more values of the index, n. If Gradient Descent (or another "gradient type" algorithm) is used to optimize E, then BPTT provides a consistent procedure for deriving the elements of $\frac{\partial E}{\partial \Theta}$ through a repeated application of the chain rule.[12]

By assuming that the conditions of Proposition 1 apply, the objective function, E, takes on the same form for all segments. Let us now apply BPTT to Eq. (74). Suppose that E depends on the readout signal, $\vec{\tilde{r}}[n]$, at some specific value of the index, n. Then it is reasonable to wish to measure the total gradient of E with respect to $\vec{\tilde{r}}[n]$:

$$\vec{\chi}[n] \equiv \vec{\nabla}_{\vec{\tilde{r}}[n]} E = \frac{\partial E}{\partial \vec{\tilde{r}}[n]} \tag{78}$$

Since $\vec{\tilde{r}}[n]$ is explicitly dependent on $\vec{\tilde{s}}[n]$, it follows that $\vec{\tilde{s}}[n]$ also influences E, and one should be interested in measuring the total gradient of E with respect to $\vec{\tilde{s}}[n]$:

$$\vec{\psi}[n] \equiv \vec{\nabla}_{\vec{\tilde{s}}[n]} E = \frac{\partial E}{\partial \vec{\tilde{s}}[n]} \tag{79}$$

Quite often in practice, the overall objective function is defined as the sum of separate contributions involving the readout signal, $\vec{\tilde{r}}[n]$, at each individual value of the index, n:

$$E = \sum_{n=0}^{K_m - 1} E(\vec{\tilde{r}}[n]) \tag{80}$$

Because of the presence of the individual penalty terms, $E(\vec{\tilde{r}}[n])$, in Eq. (80) for the overall objective function of the system, it may be tempting to use the chain rule directly with respect to $\vec{\tilde{r}}[n]$ in isolation and simply conclude that $\vec{\chi}[n]$ in Eq. (78) is equal to $\frac{\partial E(\vec{\tilde{r}}[n])}{\partial \vec{\tilde{r}}[n]} \odot \frac{dG_d(\vec{z})}{d\vec{z}}\rfloor_{z=\vec{\tilde{s}}[n]}$, where the $\odot$ operator denotes the element-wise vector product. However, this would miss an important additional component of the gradient with respect to the state signal. The subtlety is that for an RNN, the state signal, $\vec{\tilde{s}}[n]$, at $n = k$ also influences the state signal, $\vec{\tilde{s}}[n]$, at $n = k + 1$ [19,63]. The dependency of $\vec{\tilde{s}}[n+1]$ on $\vec{\tilde{s}}[n]$ through $\vec{\tilde{r}}[n]$ becomes apparent by rewriting Eq. (74) at the index, $n + 1$:

$$\begin{aligned} \vec{\tilde{s}}[n+1] &= W_r \vec{\tilde{r}}[n] + W_x \vec{\tilde{x}}_m[n+1] + \vec{\theta}_s \\ \vec{\tilde{r}}[n] &= G(\vec{\tilde{s}}[n]) \\ \vec{\tilde{r}}[n+1] &= G(\vec{\tilde{s}}[n+1]) \end{aligned} \tag{81}$$

Hence, accounting for both dependencies, while applying the chain rule, gives the expressions for the total partial derivative of the objective function with respect to the readout signal and the state signal at the index, n:

$$\vec{\chi}[n] = \frac{\partial E(\vec{\tilde{r}}[n])}{\partial \vec{\tilde{r}}[n]} + W_r \vec{\psi}[n+1] \tag{82}$$

$$\vec{\psi}[n] = \vec{\chi}[n] \odot \frac{dG(\vec{z})}{d\vec{z}}\rfloor_{z=\vec{\tilde{s}}[n]} \tag{83}$$

$$= \left(\frac{\partial E(\vec{\tilde{r}}[n])}{\partial \vec{\tilde{r}}[n]} + W_r \vec{\psi}[n+1] \right) \odot \frac{dG(\vec{z})}{d\vec{z}}\rfloor_{z=\vec{\tilde{s}}[n]} \tag{84}$$

Eqs. (82) and (84) show that the total partial derivatives of the objective function form two sequences, which progress in the "backward" direction of the index, n. These sequences represent the dual counterparts of the sequence generated by unrolling Eq. (74) in the "forward" direction of the index, n. Therefore, just as Eq. (74) requires the initialization of the segment's state signal using Eq. (73), the sequence formed by the total partial derivative of the objective function with respect to the state signal (commonly designated as the "the error gradient") requires that Eq. (82) must also be initialized:

$$\vec{\psi}[n = K_m] = \vec{0} \tag{85}$$

Applying the chain rule to Eq. (74) and using Eq. (84), gives the expressions for the derivatives of the model's parameters:

$$\frac{\partial E}{\partial \Theta}[n] = \left\{ \frac{\partial E}{\partial W_r}[n], \frac{\partial E}{\partial W_x}[n], \frac{\partial E}{\partial \vec{\theta}_s}[n] \right\} \tag{86}$$

$$\frac{\partial E}{\partial W_r}[n] = \vec{\psi}[n] \vec{\tilde{r}}^T[n-1] \tag{87}$$

$$\frac{\partial E}{\partial W_x}[n] = \vec{\psi}[n] \vec{\tilde{x}}_m^T[n] \tag{88}$$

$$\frac{\partial E}{\partial \vec{\theta}_s}[n] = \vec{\psi}[n] \tag{89}$$

$$\frac{dE}{d\Theta} = \sum_{n=0}^{K_m - 1} \frac{\partial E}{\partial \Theta}[n] \tag{90}$$

Note that for an RNN cell, unrolled for K_m steps in order to cover a segment containing K_m training samples, the same set of the model parameters, Θ, is shared by all the steps. This is because Θ is the parameter of the RNN system as a whole. Consequently, the

[12] The name "Back Propagation Through Time" reflects the origins of recurrent neural networks in continuous-time domain and differential equations. While not strictly accurate, given the discrete nature of the RNN system under consideration, "Back Propagation Through Time" is easy to remember, carries historical significance, and should not be the source of confusion.

total derivative of the objective function, E, with respect to the model parameters, Θ, has to include the contributions from all steps of the unrolled sequence. This is captured in Eq. (90), which can now be used as part of optimization by Gradient Descent. Another key observation is that according to Eqs. (87), (88), and (89), all of the quantities essential for updating the parameters of the system, Θ, during training are directly proportional to $\vec{\psi}[n]$.

When the RNN system is trained using BPTT, the error gradient signal flows in the reverse direction of the index, n, from that of the sequence itself. Let $\left\langle \vec{\psi}[k] \right\rangle \rfloor_{0 \leq k < n}$ denote all terms of the sequence, each of whose elements, $\vec{\psi}[k]$, is the gradient of E with respect to the state signal, $\vec{s}[k]$, at the index, k, for all $k < n$, ending at $\vec{\psi}[k = 0]$, the start of the sequence. Then Eq. (84) reveals that $\vec{\psi}[n]$, the gradient of E with respect to the state signal, $\vec{s}[n]$, at some value of the index, n, no matter how large, can influence the entire ensemble, $\left\langle \vec{\psi}[k] \right\rangle \rfloor_{0 \leq k < n}$. Furthermore, by Proposition 1, $\vec{\psi}[n]$ depends on the truncated ensemble, $\left\langle \vec{\psi}[k] \right\rangle \rfloor_{n < k \leq K_m - 1}$. Thus, of a particular interest is the fraction of $\vec{\psi}[n]$ that is retained from back propagating $\vec{\psi}[l]$, where $l \gg n$. This component of the gradient of the objective function is responsible for adjusting the model's parameters, Θ, in a way that uses the information available at one sample to reduce the cost of the system making an error at a distant sample. If these types of contributions to $\vec{\psi}[n] \rfloor_{0 \leq n \leq K_m - 1}$ are well-behaved numerically, then the model parameters learned by using the Gradient Descent optimization procedure will able to incorporate the long-range interactions among the samples in the RNN window effectively during inference.

Expanding the recursion in Eq. (84) from the step with the index, n, to the step with the index, $l \leq K_m - 1$, where $l \gg n$, gives:

$$\frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} = \prod_{k=n+1}^{l} W_r \odot \frac{dG(\vec{z})}{d\vec{z}} \rfloor_{z = \vec{s}[k]} \tag{91}$$

From Eq. (91), the magnitude of the overall Jacobian matrix, $\frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]}$, depends on the product of $l - n$ individual Jacobian matrices, $W_r \odot \frac{dG(\vec{z})}{d\vec{z}} \rfloor_{z = \vec{s}[k]}$.[13] Even though the truncated unrolled RNN system is guaranteed to be stable by design, since in the case of long-range interactions the unrolled window size, K_m, and the distance between the samples of interest, $l - n$, are both large, the stability analysis is helpful in estimating the magnitude of $\frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]}$ in Eq. (91). If all eigenvalues, μ_i, of W_r satisfy the requirement for stability, $0 < \mu_i < 1$, then $\|W_r\| < 1$. Combined with the fact that $\left\| \frac{dG(\vec{z})}{d\vec{z}} \right\| < 1$ (which follows from the choice of the warping function advocated in Section 2), this yields:

$$\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\| \sim \left(\|W_r\| \cdot \left\| \frac{dG(\vec{z})}{d\vec{z}} \right\| \right)^{l-n} \tag{92}$$

$$\sim \|W_r\|^{l-n} \cdot \left\| \frac{dG(\vec{z})}{d\vec{z}} \right\|^{l-n} \approx 0 \tag{93}$$

Conversely, if at least one eigenvalue of W_r violates the requirement for stability, the term $\|W_r\|^{l-n}$ will grow exponentially. This can lead to two possible outcomes for the RNN system in Eq. (74). In one scenario, as the state signal, $\vec{s}[n]$, grows, the elements of the readout signal, $\vec{r}[n]$, eventually saturate at the "rails" (the flat regions) of the warping function. Since in the saturation regime, $\frac{dG(\vec{z})}{d\vec{z}} = \vec{0}$, the result is again $\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\| \approx 0$. In another, albeit rare, scenario, the state signal, $\vec{s}[n]$, is initially biased in the quasi-linear region of the warping function, where $\frac{dG(\vec{z})}{d\vec{z}} \neq \vec{0}$. If the input, $\vec{x}_m[n]$, then guides the system to stay in this mode for a large number of steps, $\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\|$ will grow, potentially resulting in an overflow. Consequently, training the standard RNN system on windows spanning many data samples using Gradient Descent is hampered by either vanishing or exploding gradients, regardless of whether or not the system is large-signal stable. In either case, as long as Gradient Descent optimization is used for training the RNN, regulating $\vec{\psi}[n]$ will be challenging in practice, leaving no reliable mechanism for updating the parameters of the system, Θ, in a way that would enable the trained RNN model to infer both $\vec{r}[n]$ and $\vec{r}[l \gg n]$ optimally.[14] The most effective solution so far is the Long Short-Term Memory (LSTM) cell architecture [1,19,63,67].

5. From RNN to vanilla LSTM network

The Long Short-Term Memory (LSTM) network was invented with the goal of addressing the vanishing gradients problem. They key insight in the LSTM design was to incorporate nonlinear, data-dependent controls into the RNN cell, which can be trained to ensure that the gradient of the objective function with respect to the state signal (the quantity directly proportional to the parameter updates computed during training by Gradient Descent) does not vanish [1]. The LSTM cell can be rationalized from the canonical RNN cell by reasoning about Eq. (30) and introducing changes that make the system robust and versatile.

In the RNN system, the observable readout signal of the cell is the warped version of the cell's state signal itself. A weighted copy of this warped state signal is fed back from one step to the next as part of the update signal to the cell's state. This tight coupling between the readout signal at one step and the state signal at the next step directly impacts the gradient of the objective function with respect to the state signal. This impact is compounded during the training phase, culminating in the vanishing/exploding gradients.

Several modifications to the cell's design can be undertaken to remedy this situation. As a starting point, it is useful to separate the right hand side of Eq. (30) (the cell's updated state signal at a step with the index, n) into two parts[15]:

$$\vec{s}[n] = \vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right) + \vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right) \tag{94}$$

$$\vec{r}[n] = G_d(\vec{s}[n]) \tag{95}$$

$$\vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right) = W_s \vec{s}[n-1] \tag{96}$$

$$\vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right) = W_r \vec{r}[n-1] + W_x \vec{x}[n] + \vec{\theta}_s \tag{97}$$

where $G_d(\vec{z})$ is the hyperbolic tangent as before.[16] The first part, $\vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right)$, carries forward the contribution from the state

[13] By convention, the element-wise multiplication by a vector is equivalent to the multiplication by a diagonal matrix, in which the elements of the vector occupy the main diagonal.

[14] A detailed treatment of the difficulties encountered in training RNNs is presented in [63]. The problem is defined using Eq. (80), which leads to the formulas for the gradients of the individual objective function at each separate step with respect to the model's parameters. Then the behavior of these formulas as a function of the index of the step is analyzed, following the approach in [71]. In contrast, the present analysis follows the method described in [62] and [18,19,64,65]. The total gradient of the objective function with respect to the state signal at each step is pre-computed using Eq. (84). Then the behavior of the members of this sequence as a function of the number of steps separating them is analyzed. The results and conclusions of these dual approaches are, of course, identical.

[15] In the remainder of this document, the prototype segment notation of Eq. (74) is being omitted for simplicity. Unless otherwise specified, it is assumed that all operations are intended to be on the domain of a segment, where the sample index, n, traverses the steps of the segment, $0 \leq n \leq K_m - 1$, for every segment with the index, m, in $0 \leq m \leq M - 1$.

[16] Throughout this document, the subscript "c" stands for "control", while the subscript "d" stands for "data".

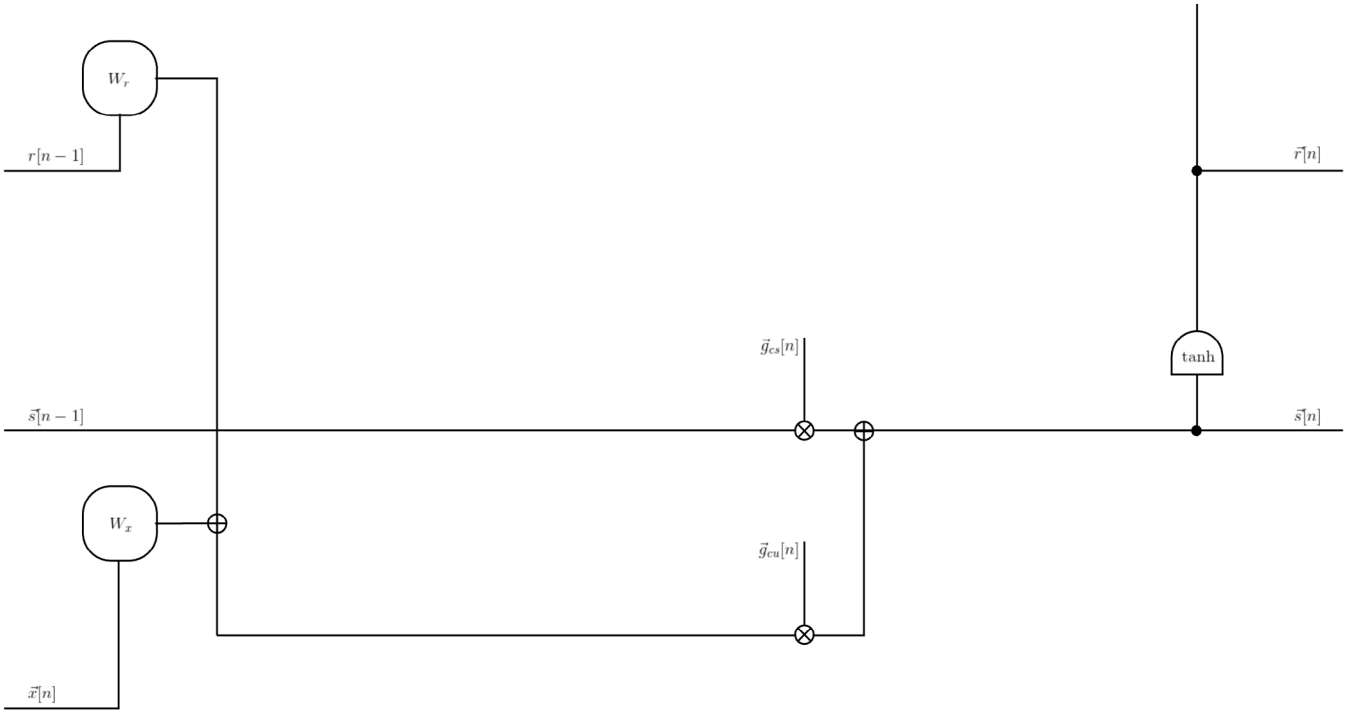


Fig. 4. Expanding the canonical RNN system by adding the "control state" gate, $\vec{g}_{cs}[n]$, to control the amount of the state signal, retained from the previous step and the "control update" gate, $\vec{g}_{cu}[n]$, to regulate the amount of the update signal — to be injected into the state signal at the current step.

signal at the previous step. The second part, $\vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right)$, represents the update information, consisting of the combination of the readout signal from the previous step and the input signal (the external driving force) at the current step (plus the bias vector, $\vec{\theta}_s$).[17] According to Eq. (94), the state signal blends both sources of information in equal proportions at every step. These proportions can be made adjustable by multiplying the two quantities by the special "gate" signals, $\vec{g}_{cs}[n]$ ("control state") and $\vec{g}_{cu}[n]$ ("control update"), respectively:

$$\vec{s}[n] = \vec{g}_{cs}[n] \odot \vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right) + \vec{g}_{cu}[n] \odot \vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right) \tag{98}$$

$$\vec{0} \le \vec{g}_{cs}[n], \vec{g}_{cu}[n] \le \vec{1} \tag{99}$$

The elements of gate signals are non-negative fractions. The shorthand notation, $\vec{g}[n] \in [\vec{0}, \vec{1}]$ (alternatively, $\vec{0} \le \vec{g}[n] \le \vec{1}$), means that the values of all elements of a vector-valued gate signal, $\vec{g}[n]$, at a step with the index, n, lie on a closed segment between 0 and 1.

The gate signals, $\vec{g}_{cs}[n]$ and $\vec{g}_{cu}[n]$, in Eqs. (98) and (99) provide a mechanism for exercising a fine-grained control of the two types of contributions to the state signal at every step. Specifically, $\vec{g}_{cs}[n]$ makes it possible to control the amount of the state signal, retained from the previous step, and $\vec{g}_{cu}[n]$ regulates the amount of the update signal — to be injected into the state signal at the current step.[18]

From the derivation of the standard RNN system in Section 2, W_s in Eq. (96) is a diagonal matrix with positive fractions, $\frac{1}{|a_{ii}|}$, on its main diagonal. Hence, since the elements of $\vec{g}_{cs}[n]$ are also fractions, setting:

$$W_s = I \tag{100}$$

in $\vec{g}_{cs}[n] \odot W_s$ is acceptable as long as the gate functions are parametrizable and their parameters are learned during training. Under these conditions, Eq. (96) can be simplified to:

$$\vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right) = \vec{s}[n-1] \tag{101}$$

so that Eq. (98) becomes:

$$\begin{aligned}\vec{s}[n] &= \vec{g}_{cs}[n] \odot \vec{\mathcal{F}}_s\left(\vec{s}[n-1]\right) + \vec{g}_{cu}[n] \odot \vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right) \\ &= \vec{g}_{cs}[n] \odot \vec{s}[n-1] + \vec{g}_{cu}[n] \odot \vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right)\end{aligned} \tag{102}$$

Hence, the contribution from the state signal at the previous step remains fractional, insuring the stability of the overall system. Diagrammatically, the insertion of the expanded controls from Eq. (102) into the canonical RNN system of Eq. (30) transforms Fig. 1 into Fig. 4.

While the update term, $\vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right)$, as a whole is now controlled by $\vec{g}_{cu}[n]$, the internal composition of $\vec{\mathcal{F}}_u\left(\vec{r}[n-1], \vec{x}[n]\right)$ itself needs to be examined. According to Eq. (97), the readout signal from the previous step and the input signal at the current step constitute the update candidate signal on every step with the index, n, with both of these terms contributing in equal proportions. The issue with always utilizing $W_r\vec{r}[n-1]$ in its entirety is that when $\vec{g}_{cu}[n] \sim 1$, $\vec{s}[n-1]$ and $\vec{s}[n]$ become connected through W_r and the warping function. Based on Eq. (91),

[17] In discrete systems, the concept of time is only of historical significance. The proper terminology would be to use the word "adjacent" when referring to quantities at the neighboring steps. Here, the terms "previous", "current", and "next" are sometimes used only for convenience purposes. The RNN systems can be readily unrolled in the opposite direction, in which case all indices are negated and the meaning of "previous" and "next" is reversed. As a matter of fact, improved performance has been attained with bi-directional processing in a variety of applications [16,19,24,25,30,72]. Moreover, if the input data is prepared ahead of time and is made available to the system in its entirety, then the causality restriction can be relaxed altogether. This can be feasible in applications, where the entire training data set or a collection of independent training data segments is gathered before processing commences. Non-causal processing (i.e., a technique characterized by taking advantage of the input data "from the future") can be advantageous in detecting the presence of "context" among data samples. Utilizing the information at the "future" steps as part of context for making decisions at the "current" step is often beneficial for analyzing audio, speech, and text.

[18] The significance of the element-wise control of the "content" signals (here called the "data" signals) exerted by the gates in the LSTM network has been independently recognized and researched by others [73].

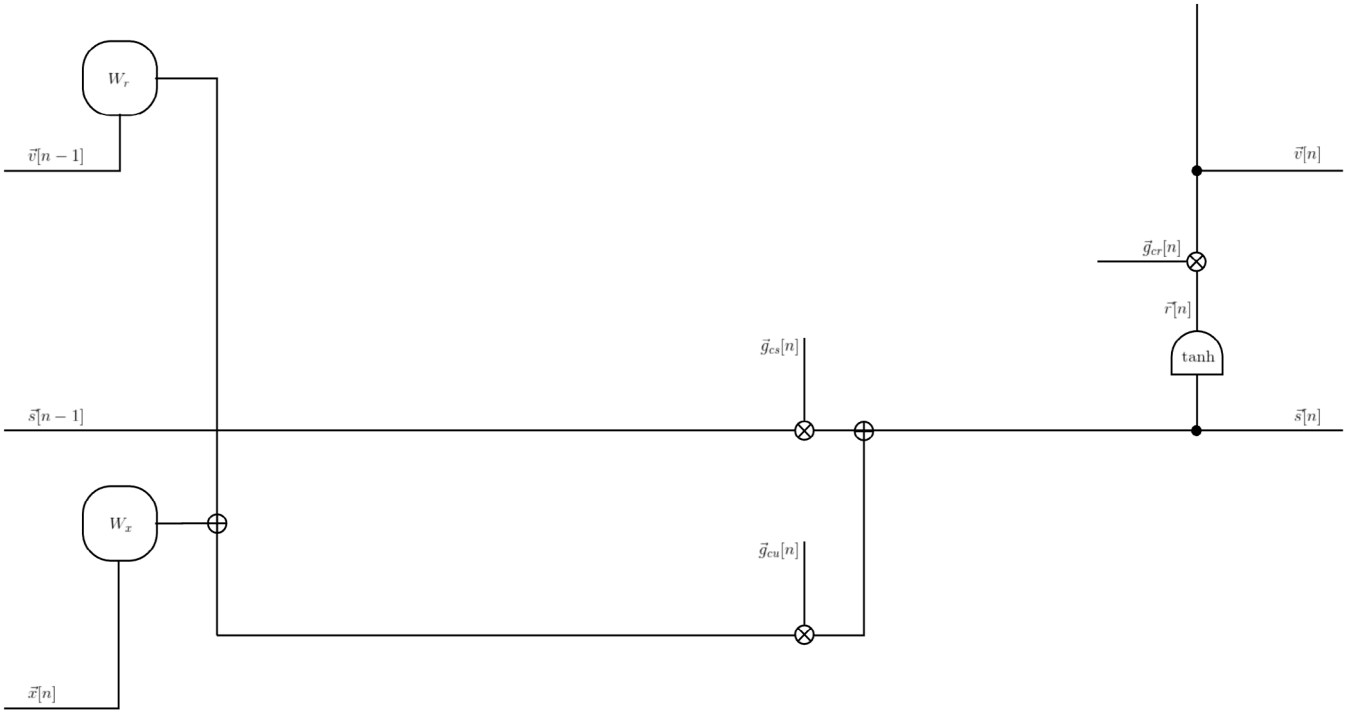


Fig. 5. The "control readout" gate, $\vec{g}_{cr}[n]$, determines the fractional amount of the readout signal that becomes the cell's observable value signal at the current step.

this link constrains the gradient of the objective function with respect to the state signal, thus predisposing the system to the vanishing/exploding gradients problem. To throttle this feedback path, the readout signal, $\vec{r}[n]$, will be apportioned by another gate signal, $\vec{g}_{cr}[n]$ ("control readout"), as follows:

$$\vec{v}[n] = \vec{g}_{cr}[n] \odot \vec{r}[n] \tag{103}$$

$$\vec{0} \leq \vec{g}_{cr}[n] \leq \vec{1} \tag{104}$$

The gating control, $\vec{g}_{cr}[n]$, determines the fractional amount of the readout signal that becomes the cell's observable value signal at the step with the index, n. Thus, using $\vec{v}[n-1]$ in place of $\vec{r}[n]$ in Eq. (97) transforms it into:

$$\vec{\mathcal{F}}_u\left(\vec{v}[n-1], \vec{x}[n]\right) = W_r\vec{v}[n-1] + W_x\vec{x}[n] + \vec{\theta}_s \tag{105}$$

The RNN cell schematic diagram, expanded to accommodate the control readout gate, introduced in Eq. (103), and the modified recurrence relationship, employed in Eq. (105), appears in Fig. 5.

Even though the external input does not affect the system's stability or impact its susceptibility to vanishing/exploding gradients, pairing the input with its own "control input" gate makes the system more flexible.

Multiplying the external input signal, $\vec{x}[n]$, in Eq. (97) by a dedicated gate signal, $\vec{g}_{cx}[n]$, turns Eq. (105) into:

$$\vec{\mathcal{F}}_u\left(\vec{v}[n-1], \vec{x}[n]\right) = W_r\vec{v}[n-1] + \vec{g}_{cx}[n] \odot W_x\vec{x}[n] + \vec{\theta}_s \tag{106}$$

According to Eqs. (103) and (106), utilizing both the control readout gate, $\vec{g}_{cr}[n]$, and the control input gate, $\vec{g}_{cx}[n]$, allows for the update term, $\vec{\mathcal{F}}_u\left(\vec{v}[n-1], \vec{x}[n]\right)$, to contain an arbitrary mix of the readout signal and the external input. The control input gate signal, $\vec{g}_{cx}[n]$, will be later incorporated as part of extending the Vanilla LSTM cell. For now, it is assumed for simplicity that $\vec{g}_{cx}[n] = \vec{1}$, so Eq. (106) reduces to Eq. (105).

The dynamic range of the value signal of the cell, $\vec{v}[n]$, is determined by the readout signal, $\vec{r}[n]$, which is bounded by the warping nonlinearity, $G_d(z)$. In order to maintain the same dynamic range while absorbing the contributions from the input signal, $\vec{x}[n]$ (or $\vec{g}_{cx}[n] \odot \vec{x}[n]$ if the control input gate is part of the system architecture), the aggregate signal, $\vec{\mathcal{F}}_u\left(\vec{v}[n-1], \vec{x}[n]\right)$, is tempered by the saturating warping nonlinearity, $G_d(z)$, so as to produce the update candidate signal, $\vec{u}[n]$:

$$\vec{u}[n] = G_d\left(\vec{\mathcal{F}}_u\left(\vec{v}[n-1], \vec{x}[n]\right)\right) \tag{107}$$

Thus, replacing the update term in Eq. (102) with $\vec{u}[n]$, given by Eq. (107), finally yields[19]:

$$\vec{s}[n] = \vec{g}_{cs}[n] \odot \vec{s}[n-1] + \vec{g}_{cu}[n] \odot \vec{u}[n] \tag{108}$$

which is a core constituent of the set of formulas defining the cell of the Vanilla LSTM network. According to Eq. (108), the state signal of the cell at the current step is a weighted combination of the state signal of the cell at the previous step and the aggregation of historical and novel update information available at the present step. The complete data path of the Vanilla LSTM cell, culminating from fortifying the canonical RNN system with gating controls and signal containment, is illustrated in Fig. 6.

Example 2. For an idealized illustration of the ability of the LSTM cell to propagate the error gradient unattenuated, set $\vec{g}_{cs}[n]$ to $\vec{1}$ and both, $\vec{g}_{cu}[n]$ and $\vec{g}_{cr}[n]$, to $\vec{0}$ for all steps in the segment. Then $\vec{s}[n] = \vec{s}[n-1]$ and $\vec{\psi}[n] = \vec{\psi}[n+1]$ for all steps in the segment. The inventors of the LSTM network named this mode the "Constant Error Carousel" (CEC) to underscore that the error gradient is recirculated and the state signal of the cell is refreshed on every step.[20] Essentially, the multiplicative gate units open

[19] Note the notation change: in the rest of the document, the symbol, $\vec{u}[n]$, has the meaning of the update candidate signal (not the vector-valued unit step function).

[20] However, the design of the LSTM network does not address explicitly the exploding gradients problem. During training, the derivatives can still become excessively large, leading to numerical problems. To prevent this, all derivatives of the objective function with respect to the state signal are renormalized to lie within a predefined range [19,22,63].

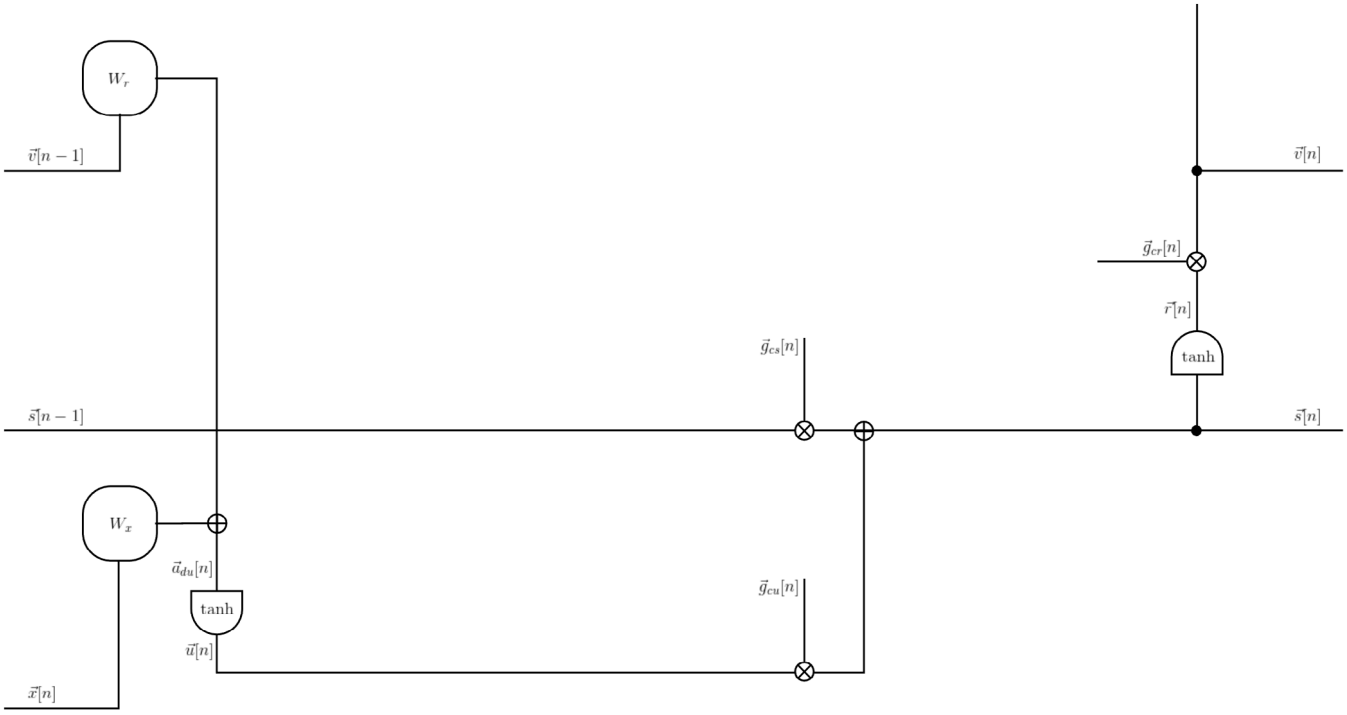


Fig. 6. In the Vanilla LSTM network, the state signal of the cell at the current step is a weighted combination of the state signal of the cell at the previous step and the aggregation of historical and novel update information available at the present step.

and close access to constant error gradient flow through CEC as part of the operation of the LSTM cell[21] [1,74].

In Section 4, we saw that the error gradient determines the parameter updates for training the standard RNN by Gradient Descent. It will become apparent in Section 6.9 that the same relationship holds for the Vanilla LSTM network as well. The difference is that because of the gates, the function for the error gradient of the LSTM network accommodates Gradient Descent better than that of the standard RNN does. As will be shown in Section 6.10, under certain provisions regarding the model parameters, the unrolled Vanilla LSTM cell operates in the CEC mode. If such a parameter combination emerges during training, then the parameter update information, embedded in the error gradient signal, will be back-propagated over a large number of steps of a training subsequence, imparting sensitivity to the long-range dependencies to the model parameters through the parameter update step of Gradient Descent. If the training process steers the model parameters toward causing $\left\|\vec{g}_{cs}[n]\right\| = 1$ (as in Example 2), then the LSTM network circumvents the vanishing gradient problem in this asymptotic case.

Analogously to the standard RNN, the Vanilla LSTM network, trained by Gradient Descent, can also learn the short-range dependencies among the samples of the subsequences, comprising the training data. Suppose that during training the model parameters cause $\left\|\vec{g}_{cs}[n]\right\| < 1$ (unlike in Example 2). Then, as will be elaborated in Section 6.10, the error gradient signal will decline, eventually vanishing over a finite number of steps, even if during training $\left\|\vec{g}_{cu}[n]\right\| > 0$ and/or $\left\|\vec{g}_{cr}[n]\right\| > 0$ so as to admit (by Eq. (108)) the contributions from the update candidate signal, $\vec{u}[n]$, into the composition of the state signal.

It remains to define the expressions for the gate signals, $\vec{g}_{cs}[n]$, $\vec{g}_{cr}[n]$, and $\vec{g}_{cu}[n]$. Assuming that the system will be trained with BPTT, all of its constituent functions, including the functions for the gate signals, must be differentiable. A convenient function that is continuous, differentiable, monotonically increasing, and maps the domain $(-\infty, \infty)$ into the range $(0, 1)$ is the logistic function:

$$G_c(z) \equiv \sigma(z) \equiv \frac{1}{1+e^{-z}} \tag{109}$$

$$= \frac{1+\tanh(\frac{z}{2})}{2} \tag{110}$$

$$= \frac{1+G_d(\frac{z}{2})}{2} \tag{111}$$

which is a shifted, scaled, and re-parameterized replica of the hyperbolic tangent, used as the warping function, $G_d(z)$, for the data signals in RNN and LSTM systems. When operating on vector arguments, $G_c(\vec{z})$ is computed by applying Eq. (109) to each element of the vector, $\vec{z}$, separately; the same rule applies to $G_d(\vec{z})$.

In order to determine the fractional values of the control signals, $\vec{g}_{cs}[n]$, $\vec{g}_{cu}[n]$, and $\vec{g}_{cr}[n]$, at the step with the index, n, all the data signals, from as close as possible to the index of the current step, are utilized. Specifically, for both, $\vec{g}_{cs}[n]$, which determines the fraction of the state signal, $\vec{s}[n-1]$, from the previous step and $\vec{g}_{cu}[n]$, which determines the fraction of the update candidate signal, $\vec{u}[n]$, from the current step, the available data signals are $\vec{s}[n-1]$, $\vec{v}[n-1]$, and $\vec{x}[n]$. However, note that for $\vec{g}_{cr}[n]$, which determines the fraction of the readout signal, $\vec{r}[n]$, from the current step, the available data signals are $\vec{s}[n]$, $\vec{v}[n-1]$, and $\vec{x}[n]$. This is because by Eq. (103), $\vec{r}[n]$ is available at the junction of the cell, where $\vec{g}_{cr}[n]$ is computed, and hence, by Eq. (95), $\vec{s}[n]$ is necessarily available. The input to each gate is presented as a linear combination of all the data signals available to it:

$$\vec{z}_{cs}[n] = W_{x_{cs}}\vec{x}[n] + W_{s_{cs}}\vec{s}[n-1] + W_{v_{cs}}\vec{v}[n-1] + \vec{\theta}_{cs} \tag{112}$$

$$\vec{z}_{cu}[n] = W_{x_{cu}}\vec{x}[n] + W_{s_{cu}}\vec{s}[n-1] + W_{v_{cu}}\vec{v}[n-1] + \vec{\theta}_{cu} \tag{113}$$

[21] Because of the access control functionality provided by the gates, the LSTM cell is sometimes interpreted as a differentiable version of the digital static random access memory cell [19].

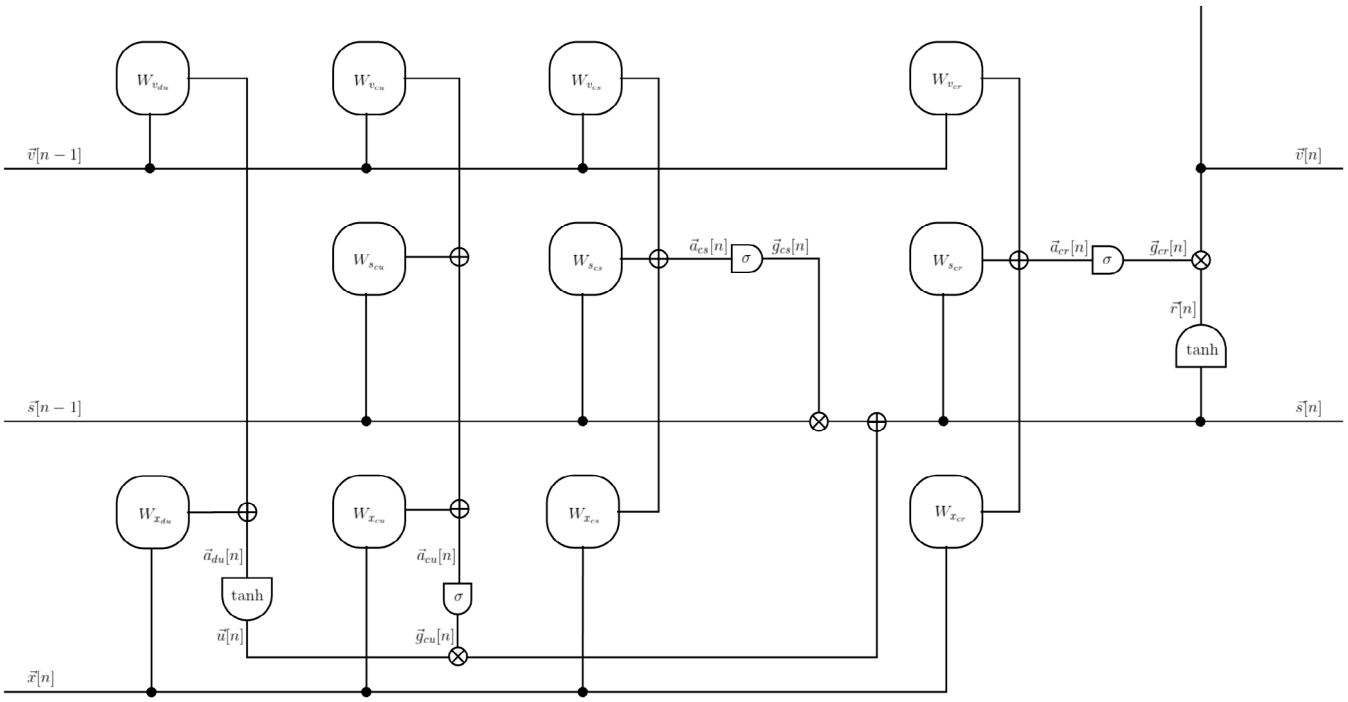


Fig. 7. Vanilla LSTM network cell. The bias parameters, $\vec{b}$, have been omitted from the figure for brevity. They can be assumed to be included without the loss of generality by appending an additional element, always set to 1, to the input signal vector, $\vec{x}[n]$, and increasing the row dimensions of all corresponding weight matrices by 1.

$$\vec{z}_{cr}[n] = W_{x_{cr}}\vec{x}[n] + W_{s_{cr}}\vec{s}[n] + W_{v_{cr}}\vec{v}[n-1] + \vec{\theta}_{cr} \tag{114}$$

Accumulating the available data signals linearly makes the application of the chain rule for BPTT straightforward, while providing a rich representation of the system's data as an input to each gate at every step. As the model parameters, $\{W_{x_{cr}}, W_{x_{cu}}, W_{x_{cs}}, W_{s_{cs}}, W_{v_{cs}}, \vec{\theta}_{cs}, W_{s_{cu}}, W_{v_{cu}}, \vec{\theta}_{cu}, W_{s_{cr}}, W_{v_{cr}}, \vec{\theta}_{cr}\}$, in Eqs. (112), (113), and (114) are being trained, the gate functions, given by:

$$\vec{g}_{cs}[n] = G_c(\vec{z}_{cs}[n]) \tag{115}$$

$$\vec{g}_{cu}[n] = G_c(\vec{z}_{cu}[n]) \tag{116}$$

$$\vec{g}_{cr}[n] = G_c(\vec{z}_{cr}[n]) \tag{117}$$

become attuned to the flow of and the variations in the training data through the system at every step. During inference, this enables the gates to modulate their corresponding data signals adaptively, utilizing all the available information at every step. In particular, the gates help to detect and mitigate the detrimental ramifications of artificial boundaries, which arise in the input sequences, due to the implicit truncation, caused by unrolling [75–77]. The gates make the LSTM system a robust model that compensates for the imperfections in the external data and is capable of generating high quality output sequences.

This concludes the derivation of the Vanilla LSTM network. The next section presents a formal self-contained summary of the Vanilla LSTM system, including the equations for training it using BPTT.

6. The vanilla LSTM network mechanism in detail

6.1. Overview

Suppose that an LSTM cell is unrolled for K steps. The LSTM cell at the step with the index, n (in the sequence of K steps), accepts the input signal, $\vec{x}[n]$, and computes the externally-accessible (i.e., observable) signal, $\vec{v}[n]$. The internal state signal of the cell at the step with the index, n, is maintained in $\vec{s}[n]$, which is normally not observable by entities external to the cell.[22] However, the computations, associated with the cell at the next adjacent step in the increasing order of the index, n (i.e., the LSTM step at the index, $n+1$), are allowed to access $\vec{s}[n]$, the state signal of the LSTM cell at the step with the index, n.

The key principle of the LSTM cell centers around organizing its internal operations according to two qualitatively different, yet cooperating, objectives: data and the control of data. The data components prepare the candidate data signals (ranging between -1 and 1), while the control components prepare the "throttle" signals (ranging between 0 and 1). Multiplying the candidate data signal by the control signal apportions the fractional amount of the candidate data that is allowed to propagate to its intended nodes in the cell. Hence, if the control signal is 0, then 0% of the candidate data amount will propagate. Conversely, if the control signal is 1, then 100% of the candidate data amount will propagate. Analogously, for intermediate values of the control signal (in the range between 0 and 1), the corresponding percentage of the candidate data amount will be made available to the next function in the cell.

As depicted in Fig. 7, the Vanilla LSTM cell contains three candidate-data/control stages: update, state, and readout.

6.2. Notation

The following notation is used consistently throughout this section to define the Vanilla LSTM cell:

- n — index of a step in the segment (or subsequence); $n = 0, \ldots, K-1$

[22] In certain advanced RNN and LSTM configurations, such as Attention Networks, the state signal is externally observable and serves as an important component of the objective function.

- K — number of steps in the unrolled segment (or subsequence)
- G_c — monotonic, bipolarly-saturating warping function for control/throttling purposes (acts as a "gate")
- G_d — monotonic, negative-symmetric, bipolarly-saturating warping function for data bounding purposes
- d_x — dimensionality of the input signal to the cell
- d_s — dimensionality of the state signal of the cell
- $\vec{x} \in \mathbb{R}^{d_x}$ — the input signal to the cell
- $\vec{s} \in \mathbb{R}^{d_s}$ — the state signal of the cell
- $\vec{v} \in \mathbb{R}^{d_s}$ — the observable value signal of the cell for external purposes (e.g., for connecting one step to the next adjacent step of the same cell in the increasing order of the step index, n; as input to another cell in the cascade of cells; for connecting to the signal transformation filter for data output; etc.)
- $\vec{a} \in \mathbb{R}^{d_s}$ — an accumulation node of the cell (linearly combines the signals from the preceding step and the present step as net input to a warping function at the present step; each cell contains several purpose-specific control and data accumulation nodes)
- $\vec{u} \in \mathbb{R}^{d_s}$ — the update candidate signal for the state signal of the cell
- $\vec{r} \in \mathbb{R}^{d_s}$ — the readout candidate signal of the cell
- $g \in \mathbb{R}^{d_s}$ — a gate output signal of the cell for control/throttling purposes
- $E \in \mathbb{R}$ — objective (cost) function to be minimized as part of the model training procedure
- $\vec{x}^T \vec{v}$ — vector–vector inner product (yields a scalar)
- $\vec{x}\vec{v}^T$ — vector–vector outer product (yields a matrix)
- $W\vec{v}$ — matrix–vector product (yields a vector)
- $\vec{x} \odot \vec{v}$ — element-wise vector product (yields a vector)

6.3. Control/throttling ("gate") nodes

The Vanilla LSTM cell uses three gate types:

- control of the fractional amount of the update candidate signal used to comprise the state signal of the cell at the present step with the index, n
- control of the fractional amount of the state signal of the cell at the adjacent lower-indexed step, $n-1$, used to comprise the state signal of the cell at the present step with the index, n
- control of the fractional amount of the readout candidate signal used to release as the externally-accessible (observable) signal of the cell at the present step with the index, n

6.4. Data set standardization

Before the operation of the LSTM network (or its parent, RNN) can commence, the external training data set, $\vec{x}_0[n]$, needs to be standardized, such that all elements of the input to the network, $\vec{x}[n]$, have the mean of 0 and the standard deviation of 1 over the training set:

$$\vec{\mu} = \frac{1}{N}\sum_{n=0}^{N-1} \vec{x}_0[n] \tag{118}$$

$$\mathcal{V} = \frac{1}{N-1}\sum_{n=0}^{N-1} \left(\vec{x}_0[n] - \vec{\mu}\right)\left(\vec{x}_0[n] - \vec{\mu}\right)^T \tag{119}$$

$$\vec{x}[n] = \left[diag\left(\sqrt{\mathcal{V}_{ii}}\right)\right]^{-1}\left(\vec{x}_0[n] - \vec{\mu}\right) \tag{120}$$

Applying the transformations in Eqs. (118), (119), and (120) to the external training samples, $\vec{x}_0[n]$, accomplishes this task. In these equations, N is the number of samples in the training set, $\vec{\mu}$ is the sample mean, and $\mathcal{V}$ is the sample auto-covariance matrix of the training set.[23]

6.5. Warping (activation) functions

As described in Section 6.1, the warping function for control needs to output a value between 0 and 1. The sigmoidal (also known as "logistic") nonlinearity is a good choice, because it is bipolarly-saturating between these values and is monotonic, continuous, and differentiable:

$$G_c(z) \equiv \sigma(z) \equiv \frac{1}{1+e^{-z}} = \frac{1+\tanh(\frac{z}{2})}{2} = \frac{1+G_d(\frac{z}{2})}{2} \tag{121}$$

Related to this function, the hyperbolic tangent is a suitable choice for the warping function for data bounding purposes:

$$G_d(z) \equiv \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\sigma(2z) - 1 = 2G_c(2z) - 1 \tag{122}$$

because it is monotonic, negative-symmetric, and bipolarly-saturating at -1 and 1 (i.e., one standard deviation of $\vec{x}[n]$ in each direction). This insures that the data warping function, $G_d(z)$, will support both negative and positive values of the standardized incoming data signal, $\vec{x}[n]$, in Eq. (120), and keep it bounded within that range (the "squashing" property).

6.6. Vanilla LSTM cell model parameters

The Vanilla LSTM cell model uses the following fifteen (15) parameter entities (with their respective dimensions and designations as indicated below):

6.6.1. Parameters of the accumulation node, $\vec{a}_{cu}[n]$, of the gate that controls the fractional amount of the update candidate signal, $\vec{u}[n]$, used to comprise the state signal of the cell at the present step with the index, n

- $W_{x_{cu}} \in \mathbb{R}^{d_s \times d_x}$ — the matrix of weights connecting the input signal, $\vec{x}[n]$, at the present step with the index, n, to the "control update" accumulation node, $\vec{a}_{cu}[n]$, of the cell at the present step with the index, n
- $W_{s_{cu}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the state signal, $\vec{s}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control update" accumulation node, $\vec{a}_{cu}[n]$, of the cell at the present step with the index, n
- $W_{v_{cu}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the externally-accessible (observable) value signal, $\vec{v}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control update" accumulation node, $\vec{a}_{cu}[n]$, of the cell at the present step with the index, n
- $\vec{b}_{cu} \in \mathbb{R}^{d_s}$ — the vector of bias elements for the "control update" accumulation node, $\vec{a}_{cu}[n]$, of the cell at the present step with the index, n

6.6.2. Parameters of the accumulation node, $\vec{a}_{cs}[n]$, of the gate that controls the fractional amount of the state signal of the cell, $\vec{s}[n-1]$, at the adjacent lower-indexed step, $n-1$, used to comprise the state signal of the cell at the present step with the index, n

- $W_{x_{cs}} \in \mathbb{R}^{d_s \times d_x}$ — the matrix of weights connecting the input signal, $\vec{x}[n]$, at the present step with the index, n, to the "control state" accumulation node, $\vec{a}_{cs}[n]$, of the cell at the present step with the index, n

[23] The test and validation sets should be standardized with the mean and standard deviation of the training set [22].

- $W_{s_{cs}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the state signal, $\vec{s}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control state" accumulation node, $\vec{a}_{cs}[n]$, of the cell at the present step with the index, n
- $W_{v_{cs}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the externally-accessible (observable) value signal, $\vec{v}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control state" accumulation node, $\vec{a}_{cs}[n]$, of the cell at the present step with the index, n
- $\vec{b}_{cs} \in \mathbb{R}^{d_s}$ — the vector of bias elements for the "control state" accumulation node, $\vec{a}_{cs}[n]$, of the cell at the present step with the index, n

6.6.3. Parameters of the accumulation node, $\vec{a}_{cr}[n]$, of the gate that controls the fractional amount of the readout candidate signal, $\vec{r}[n]$, used to release as the externally-accessible (observable) value signal of the cell at the present step with the index, n

- $W_{x_{cr}} \in \mathbb{R}^{d_s \times d_x}$ — the matrix of weights connecting the input signal, $\vec{x}[n]$, at the present step with the index, n, to the "control readout" accumulation node, $\vec{a}_{cr}[n]$, of the cell at the present step with the index, n
- $W_{s_{cr}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the state signal, $\vec{s}[n]$, at the present step with the index, n, to the "control readout" accumulation node, $\vec{a}_{cr}[n]$, of the cell at the present step with the index, n
- $W_{v_{cr}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the externally-accessible (observable) value signal, $\vec{v}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control readout" accumulation node, $\vec{a}_{cr}[n]$, of the cell at the present step with the index, n
- $\vec{b}_{cr} \in \mathbb{R}^{d_s}$ — the vector of bias elements for the "control readout" accumulation node, $\vec{a}_{cr}[n]$, of the cell at the present step with the index, n

6.6.4. Parameters of the accumulation node, $\vec{a}_{du}[n]$, for the data warping function that produces the update candidate signal, $\vec{u}[n]$, of the cell at the present step with the index, n

- $W_{x_{du}} \in \mathbb{R}^{d_s \times d_x}$ — the matrix of weights connecting the input signal, $\vec{x}[n]$, at the present step with the index, n, to the "data update" accumulation node, $\vec{a}_{du}[n]$, of the cell at the present step with the index, n
- $W_{v_{du}} \in \mathbb{R}^{d_s \times d_s}$ — the matrix of weights connecting the externally-accessible (observable) value signal, $\vec{v}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "data update" accumulation node, $\vec{a}_{du}[n]$, of the cell at the present step with the index, n
- $\vec{b}_{du} \in \mathbb{R}^{d_s}$ — the vector of bias elements for the "data update" accumulation node, $\vec{a}_{du}[n]$, of the cell at the present step with the index, n

6.6.5. All model parameters, which must be learned, combined (for notational convenience)

- All parameters of the LSTM network are commonly concatenated and represented as a whole by Θ:

$$\Theta \equiv \Big\{ W_{x_{cu}}, W_{s_{cu}}, W_{v_{cu}}, \vec{b}_{cu}, W_{x_{cs}}, W_{s_{cs}}, W_{v_{cs}}, \vec{b}_{cs}, W_{x_{cr}}, W_{s_{cr}}, W_{v_{cr}}, \vec{b}_{cr} W_{x_{du}}, W_{v_{du}}, \vec{b}_{du} \Big\} \tag{123}$$

- Arranged "thematically" (attributed by the type of an accumulation), Θ can be written as:

$$\Theta \equiv \begin{Bmatrix} W_{x_{cu}}, & W_{s_{cu}}, & W_{v_{cu}}, & \vec{b}_{cu}, \\ W_{x_{cs}}, & W_{s_{cs}}, & W_{v_{cs}}, & \vec{b}_{cs}, \\ W_{x_{cr}}, & W_{s_{cr}}, & W_{v_{cr}}, & \vec{b}_{cr}, \\ W_{x_{du}}, & & W_{v_{du}}, & \vec{b}_{du} \end{Bmatrix}. \tag{124}$$

6.7. Summary of the main entities (generalized)

The following glossary lists the main entities of the model in a generalized way (i.e., without the subscripts, indices, etc.). Note that the special quantities $\vec{\psi}, \vec{\chi}, \vec{\alpha}, \vec{\rho}, \vec{\gamma}$ will be defined in Section 6.9.

$$\left.\begin{array}{l} \vec{x} \in \mathbb{R}^{d_x} \\ \vec{s}, \vec{v}, \vec{a}, \vec{u}, \vec{r}, \vec{g} \in \mathbb{R}^{d_s} \\ \vec{\psi}, \vec{\chi}, \vec{\alpha}, \vec{\rho}, \vec{\gamma} \in \mathbb{R}^{d_s} \\ W_x \in \mathbb{R}^{d_s \times d_x} \\ W_s, W_v \in \mathbb{R}^{d_s \times d_s} \\ \vec{b} \in \mathbb{R}^{d_s} \\ E \\ N \\ K \\ n = 0, \ldots, K-1 \end{array}\right\} \tag{125}$$

6.8. Vanilla LSTM system equations ("forward pass")

It is important to highlight the general pattern of computations that govern the processes, according to which any RNN cell, and the LSTM network cell in particular, unrolled for K steps, generates sequences of samples. Namely, the quantities that characterize the step of the cell at the index, n, of the sequence depend on the quantities that characterize the step of the cell at the index, $n-1$, of the sequence.[24] The following equations fully define the Vanilla LSTM cell:

$$\vec{a}_{cu}[n] = W_{x_{cu}}\vec{x}[n] + W_{s_{cu}}\vec{s}[n-1] + W_{v_{cu}}\vec{v}[n-1] + \vec{b}_{cu} \tag{126}$$

$$\vec{a}_{cs}[n] = W_{x_{cs}}\vec{x}[n] + W_{s_{cs}}\vec{s}[n-1] + W_{v_{cs}}\vec{v}[n-1] + \vec{b}_{cs} \tag{127}$$

$$\vec{a}_{cr}[n] = W_{x_{cr}}\vec{x}[n] + W_{s_{cr}}\vec{s}[n] + W_{v_{cr}}\vec{v}[n-1] + \vec{b}_{cr} \tag{128}$$

$$\vec{a}_{du}[n] = W_{x_{du}}\vec{x}[n] + W_{v_{du}}\vec{v}[n-1] + \vec{b}_{du} \tag{129}$$

$$\vec{u}[n] = G_d(\vec{a}_{du}[n]) \tag{130}$$

$$\vec{g}_{cu}[n] = G_c(\vec{a}_{cu}[n]) \tag{131}$$

$$\vec{g}_{cs}[n] = G_c(\vec{a}_{cs}[n]) \tag{132}$$

$$\vec{g}_{cr}[n] = G_c(\vec{a}_{cr}[n]) \tag{133}$$

$$\vec{s}[n] = \vec{g}_{cs}[n] \odot \vec{s}[n-1] + \vec{g}_{cu}[n] \odot \vec{u}[n] \tag{134}$$

$$\vec{r}[n] = G_d(\vec{s}[n]) \tag{135}$$

$$\vec{v}[n] = \vec{g}_{cr}[n] \odot \vec{r}[n] \tag{136}$$

The schematic diagram of the Vanilla LSTM cell, defined by Eqs. (126)–(136), is presented in Fig. 7, and the snapshot of unrolling it (for only 4 steps as an illustration) appears in Fig. 8. In order to make it easier to isolate the specific functions performed by the components of the Vanilla LSTM cell, its schematic diagram is redrawn in Fig. 9, with the major stages comprising the cell's architecture marked by dashed rectangles annotated by the names of the respective enclosed stages.

6.9. Vanilla LSTM system derivatives ("backward pass")

This section derives the equations that are necessary for training the Vanilla LSTM network cell, unrolled for K steps, using Back

[24] If the cell is unrolled in the opposite direction, then $n-1$ is replaced by $n+1$, and the direction of evaluating the steps is reversed. For the bi-directional unrolling, the steps in both the positive and the negative directions of the index, n, need to be evaluated [72]. Here, only the positive direction is considered.

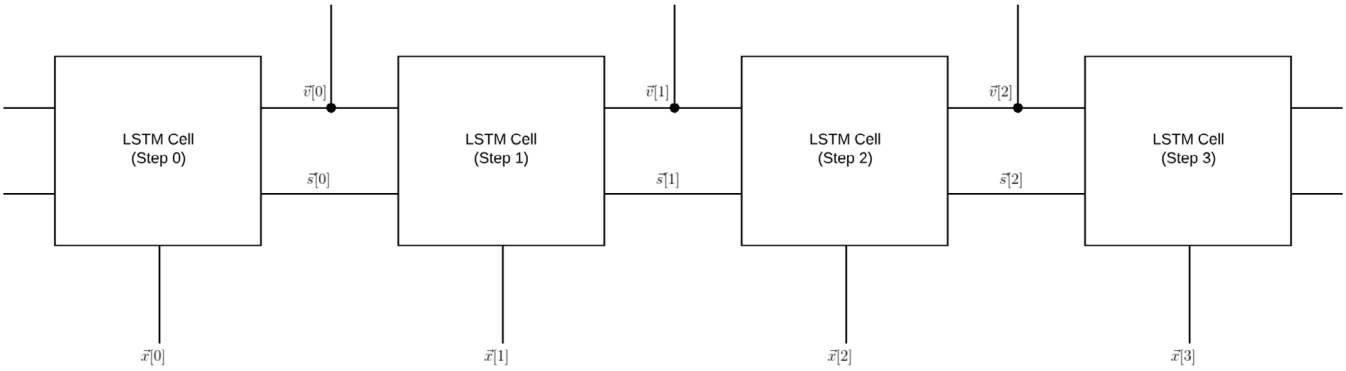


Fig. 8. Sequence of steps generated by unrolling a cell of the LSTM network (displaying 4 steps for illustration).

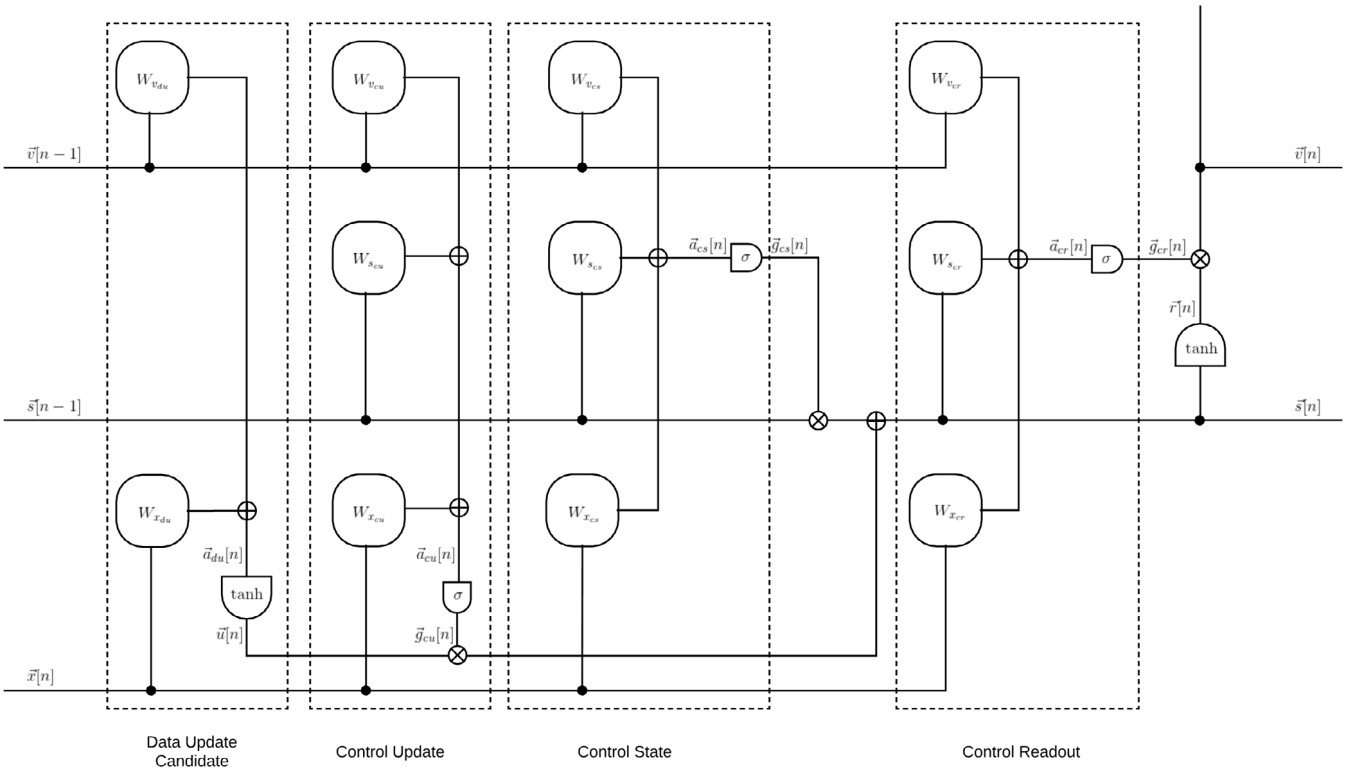


Fig. 9. Vanilla LSTM network cell from Fig. 7, with the stages of the system delineated by dashed rectangles and annotations that depict the function of each stage. As before, the bias parameters, $\vec{b}$, have been omitted from the figure for brevity. (They can be assumed to be included without the loss of generality by appending an additional element, always set to 1, to the input signal vector, $\vec{x}[n]$, and increasing the row dimensions of all corresponding weight matrices by 1.).

Propagation Through Time (BPTT). To obtain the update equations for the parameters of the system, two auxiliary "backward-moving" gradient sequences, indexed by n, are computed first: $\vec{\chi}[n]$, the total partial derivative of the objective function, E, with respect to the externally-accessible (observable) value signal, $\vec{v}[n]$, and $\vec{\psi}[n]$, the total partial derivative of the objective function, E, with respect to the state signal, $\vec{s}[n]$. The decision to "anchor" the chain rule at the border of the cell is made judiciously, guided by the principles of modular design. Expressing every intra-cell total partial derivative in terms of $\vec{\chi}[n]$ (instead of explicitly computing the total partial derivative of the objective function, E, with respect to each variable of the cell [28]) reduces the number of intermediate variables. This makes the equations for the backward pass straightforward and a natural fit for an implementation as a pluggable module [19,64,65].

Due to the backward-moving recursion of $\vec{\chi}[n]$ and $\vec{\psi}[n]$ (the gradient sequences propagate in the direction opposite to that of the state signal, $\vec{s}[n]$, as a function of the step index, n), the values of $\vec{\chi}[n]$ and $\vec{\psi}[n]$ at the index, n, depend on the values of the same quantities at the index, $n+1$, subject to the initial conditions. Once $\vec{\chi}[n]$ and $\vec{\psi}[n]$ are known, they are used to compute the total partial derivatives of the objective function, E, with respect to the accumulation nodes for each value of the index, n. These intermediate gradient sequences, named $\vec{\alpha}_{cs}[n]$, $\vec{\alpha}_{cu}[n]$, $\vec{\alpha}_{cr}[n]$, and $\vec{\alpha}_{du}[n]$, allocate the amounts contributed by the signals associated with the step at the index, n, to the total partial derivatives of the objective function, E, with respect to the model parameters. By the definition of the total derivative, these contributions have to be summed across all steps, $0 \leq n \leq K-1$, to produce the total partial derivatives of the objective function, E, with respect to the model parameters.

During the inference phase of the LSTM system, only $\vec{x}[n]$ (the input signal) and $\vec{v}[n]$ (the value signal) are externally accessible (i.e., observable). The cell accepts the input signal at each step and

computes the value signal for all steps. All the other intermediate signals are available only to the internal components and nodes of the cell, with the exception of $\vec{s}[n]$ (state signal) and $\vec{v}[n]$ (value signal), which serve both the inter- and the intra-step purposes throughout the unrolled sequence.

The cell's value signal, $\vec{v}[n]$, at the step with the index, n, can be further transformed to produce the output signal, $\vec{y}[n]$ (e.g., a commonly used form of $\vec{y}[n]$ may be obtained by computing a linear transformation of $\vec{v}[n]$, followed by a softmax operator, or a different decision function). Likewise, the input signal, too, may result from the transformation of the original raw data. For example, one kind of input pre-processing can convert the vocabulary "one-hot" vector into a more compact representation. Also, for applications where the input data set can be collected for the entire segment at once, input samples that lie within a small window surrounding the given step can be combined so as to enhance the system's "attention" to context. A non-causal input filter, designed for this purpose, will be introduced in Section 7.1 as part of extending the Vanilla LSTM cell.

We start by computing the derivatives of the warping functions from their definitions in Eqs. (121) and (122), respectively:

$$\frac{dG_c(z)}{dz} = G_c(z)(1 - G_c(z)) \tag{137}$$

$$\frac{dG_d(z)}{dz} = 1 - (G_d(z))^2 \tag{138}$$

Next, we anchor the chain rule at the border of the cell by defining $\vec{\chi}[n]$ as the total partial derivative of the objective function, E, with respect to the externally-accessible (observable) value signal, $\vec{v}[n]$, as follows:

$$\vec{\chi}[n] \equiv \vec{\nabla}_{\vec{v}[n]} E = \frac{\partial E}{\partial \vec{v}[n]} \tag{139}$$

As will become imminently evident, having $\vec{\chi}[n]$ not only makes training equations for the Vanilla LSTM cell amenable for a modular implementation at the step level, but also greatly simplifies them.

We also define the total partial derivatives of the objective function, E, with respect to three intermediate (i.e., away from the border) variables and another border variable of the Vanilla LSTM cell:

$$\vec{\rho}[n] \equiv \vec{\nabla}_{\vec{r}[n]} E = \frac{\partial E}{\partial \vec{r}[n]} \tag{140}$$

$$\vec{\gamma}[n] \equiv \vec{\nabla}_{\vec{g}[n]} E = \frac{\partial E}{\partial \vec{g}[n]} \tag{141}$$

$$\vec{\alpha}[n] \equiv \vec{\nabla}_{\vec{a}[n]} E = \frac{\partial E}{\partial \vec{a}[n]} \tag{142}$$

$$\vec{\psi}[n] \equiv \vec{\nabla}_{\vec{s}[n]} E = \frac{\partial E}{\partial \vec{s}[n]} \tag{143}$$

The border quantity in Eq. (143), $\vec{\psi}[n]$, is of special significance as it is the total partial derivative of the objective function, E, with respect to the state signal, $\vec{s}[n]$, at the index, n, of the Vanilla LSTM cell. As in the standard RNN, all parameter updates in the Vanilla LSTM network depend on $\vec{\psi}[n]$, making it the most important error gradient sequence of the system.

The backward pass equations are obtained by utilizing these border and intermediate derivatives in the application of the chain rule to the Vanilla LSTM cell, defined by Eqs. (126)–(136):

$$\vec{\chi}[n] = \left(\frac{\partial \vec{y}[n]}{\partial \vec{v}[n]}\right)^T \left(\frac{\partial E}{\partial \vec{y}[n]}\right) + \vec{f}_{\chi}[n+1] \tag{144}$$

$$\vec{\rho}[n] = \left(\frac{\partial \vec{v}[n]}{\partial \vec{r}[n]}\right)^T \left(\frac{\partial E}{\partial \vec{v}[n]}\right) = (\vec{\nabla}_{\vec{v}[n]} E) \odot \vec{g}_{cr}[n] = \vec{\chi}[n] \odot \vec{g}_{cr}[n] \tag{145}$$

$$\vec{\gamma}_{cr}[n] = \frac{\partial E}{\partial \vec{v}[n]} \frac{\partial \vec{v}[n]}{\partial \vec{g}_{cr}[n]} = (\vec{\nabla}_{\vec{v}[n]} E) \odot \vec{r}[n] = \vec{\chi}[n] \odot \vec{r}[n] \tag{146}$$

$$\vec{\alpha}_{cr}[n] = \vec{\gamma}_{cr}[n] \odot \frac{\partial \vec{g}_{cr}[n]}{\partial \vec{a}_{cr}[n]} = \vec{\gamma}_{cr}[n] \odot \frac{dG_c(z)}{dz}\Big\rfloor_{z=\vec{a}_{cr}[n]}$$
$$= \vec{\chi}[n] \odot \vec{r}[n] \odot \frac{dG_c(z)}{dz}\Big\rfloor_{z=\vec{a}_{cr}[n]} \tag{147}$$

$$\vec{\psi}[n] = \vec{\rho}[n] \odot \frac{\partial \vec{r}[n]}{\partial \vec{s}[n]} + \frac{\partial \vec{a}_{cr}[n]}{\partial \vec{s}[n]} \vec{\alpha}_{cr}[n] + \vec{f}_{\psi}[n+1] \tag{148}$$

$$= \vec{\rho}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{s}[n]} + W_{s_{cr}} \vec{\alpha}_{cr}[n] + \vec{f}_{\psi}[n+1] \tag{149}$$

$$= \vec{\chi}[n] \odot \vec{g}_{cr}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{s}[n]} + W_{s_{cr}} \vec{\alpha}_{cr}[n] + \vec{f}_{\psi}[n+1] \tag{150}$$

$$\vec{\alpha}_{cs}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{g}_{cs}[n]} \odot \frac{\partial \vec{g}_{cs}[n]}{\partial \vec{a}_{cs}[n]}$$
$$= \vec{\psi}[n] \odot \vec{s}[n-1] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cs}[n]} \tag{151}$$

$$\vec{\alpha}_{cu}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{g}_{cu}[n]} \odot \frac{\partial \vec{g}_{cu}[n]}{\partial \vec{a}_{cu}[n]}$$
$$= \vec{\psi}[n] \odot \vec{u}[n] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cu}[n]} \tag{152}$$

$$\vec{\alpha}_{du}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{u}[n]} \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{du}[n]}$$
$$= \vec{\psi}[n] \odot \vec{g}_{cu}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{du}[n]} \tag{153}$$

where:

$$\vec{f}_{\chi}[n+1] = W_{v_{cu}} \vec{\alpha}_{cu}[n+1] + W_{v_{cs}} \vec{\alpha}_{cs}[n+1] + W_{v_{cr}} \vec{\alpha}_{cr}[n+1] + W_{v_{du}} \vec{\alpha}_{du}[n+1] \tag{154}$$

$$\vec{f}_{\psi}[n+1] = W_{s_{cu}} \vec{\alpha}_{cu}[n+1] + W_{s_{cs}} \vec{\alpha}_{cs}[n+1] + \vec{g}_{cs}[n+1] \odot \vec{\psi}[n+1] \tag{155}$$

are the portions of the total derivative of the objective function, E, with respect to the cell's value signal and the cell's state signal, respectively, contributed by the quantities evaluated at the step with the index, $n+1$.

The total partial derivatives of the objective function, E, with respect to the model parameters at the step with the index, n, are directly proportional to the "accumulation derivatives",[25] given by Eqs. (147), (151), (152), and Eq. (153). Hence, by referring once again to the definition of the Vanilla LSTM cell in Eqs. (126)–(136), we obtain:

$$\frac{\partial E}{\partial W_{x_{cu}}}[n] = \vec{\alpha}_{cu}[n] \vec{x}^T[n] \tag{156}$$

$$\frac{\partial E}{\partial W_{s_{cu}}}[n] = \vec{\alpha}_{cu}[n] \vec{s}^T[n-1] \tag{157}$$

$$\frac{\partial E}{\partial W_{v_{cu}}}[n] = \vec{\alpha}_{cu}[n] \vec{v}^T[n-1] \tag{158}$$

$$\frac{\partial E}{\partial \vec{b}_{cu}}[n] = \vec{\alpha}_{cu}[n] \tag{159}$$

$$\frac{\partial E}{\partial W_{x_{cs}}}[n] = \vec{\alpha}_{cs}[n] \vec{x}^T[n] \tag{160}$$

$$\frac{\partial E}{\partial W_{s_{cs}}}[n] = \vec{\alpha}_{cs}[n] \vec{s}^T[n-1] \tag{161}$$

[25] As noted in Section 5, during training, the total derivatives of the objective function, E, with respect to the cell's accumulation signals ("accumulation derivatives") can become excessively large. In order to prevent these kinds of numerical problems, all accumulation derivatives are clipped to lie between -1 and 1, a range that is suitable for the particular choices of the control and data warping functions [19,22,63].

$$\frac{\partial E}{\partial W_{v_{cs}}}[n] = \vec{\alpha}_{cs}[n]\vec{v}^{T}[n-1] \tag{162}$$

$$\frac{\partial E}{\partial \vec{b}_{cs}}[n] = \vec{\alpha}_{cs}[n] \tag{163}$$

$$\frac{\partial E}{\partial W_{x_{cr}}}[n] = \vec{\alpha}_{cr}[n]\vec{x}^{T}[n] \tag{164}$$

$$\frac{\partial E}{\partial W_{s_{cr}}}[n] = \vec{\alpha}_{cr}[n]\vec{s}^{T}[n] \tag{165}$$

$$\frac{\partial E}{\partial W_{v_{cr}}}[n] = \vec{\alpha}_{cr}[n]\vec{v}^{T}[n-1] \tag{166}$$

$$\frac{\partial E}{\partial \vec{b}_{cr}}[n] = \vec{\alpha}_{cr}[n] \tag{167}$$

$$\frac{\partial E}{\partial W_{x_{du}}}[n] = \vec{\alpha}_{du}[n]\vec{x}^{T}[n] \tag{168}$$

$$\frac{\partial E}{\partial W_{v_{du}}}[n] = \vec{\alpha}_{du}[n]\vec{v}^{T}[n-1] \tag{169}$$

$$\frac{\partial E}{\partial \vec{b}_{du}}[n] = \vec{\alpha}_{du}[n] \tag{170}$$

Arranged congruently with Eq. (124), the total partial derivative of the objective function, E, with respect to the model parameters, Θ, at the step with the index, n, is:

$$\frac{\partial E}{\partial \Theta}[n] = \left\{ \begin{array}{cccc} \frac{\partial E}{\partial W_{x_{cu}}}[n], & \frac{\partial E}{\partial W_{s_{cu}}}[n], & \frac{\partial E}{\partial W_{v_{cu}}}[n], & \frac{\partial E}{\partial \vec{b}_{cu}}[n], \\ \frac{\partial E}{\partial W_{x_{cs}}}[n], & \frac{\partial E}{\partial W_{s_{cs}}}[n], & \frac{\partial E}{\partial W_{v_{cs}}}[n], & \frac{\partial E}{\partial \vec{b}_{cs}}[n], \\ \frac{\partial E}{\partial W_{x_{cr}}}[n], & \frac{\partial E}{\partial W_{s_{cr}}}[n], & \frac{\partial E}{\partial W_{v_{cr}}}[n], & \frac{\partial E}{\partial \vec{b}_{cr}}[n], \\ \frac{\partial E}{\partial W_{x_{du}}}[n], & & \frac{\partial E}{\partial W_{v_{du}}}[n], & \frac{\partial E}{\partial \vec{b}_{du}}[n] \end{array} \right\}. \tag{171}$$

When the Vanilla LSTM cell is unrolled for K steps in order to cover one full segment of training samples, the same set of the model parameters, Θ, is shared by all the steps. This is because Θ is the parameter of the Vanilla LSTM cell as a whole. Consequently, the total derivative of the objective function, E, with respect to the model parameters, Θ, has to include the contributions from all steps of the unrolled sequence:

$$\frac{dE}{d\Theta} = \sum_{n=0}^{K-1} \frac{\partial E}{\partial \Theta}[n] \tag{172}$$

The result from Eq. (172) can now be used as part of optimization by Gradient Descent. In practice, Eq. (172) is computed for a batch of segments,[26] and the sum of the parameter gradients over all segments in the batch is then supplied to the Gradient Descent algorithm for updating the model parameters.[27]

6.10. Error gradient sequences in vanilla LSTM system

Section 5 mentions that because of the action of the gates, the LSTM network is more compatible with the Gradient Descent training procedure than the standard RNN system is. As discussed in Sections 4 and 6.9, for Gradient Descent to be effective, the elements of $\frac{\partial E}{\partial \Theta}[n]$ in Eq. (171) must be well-behaved numerically. In particular, this implies that the intermediate gradient sequences, $\vec{\alpha}_{cs}[n]$, $\vec{\alpha}_{cu}[n]$, $\vec{\alpha}_{cr}[n]$, and $\vec{\alpha}_{du}[n]$, and hence the border gradient sequences, $\vec{\chi}[n]$ and $\vec{\psi}[n]$, must be able to sustain a steady flow of information over long ranges of the step index, n. Expanding Eq. (150) produces:

$$\vec{\psi}[n] = \vec{\chi}[n] \odot \vec{g}_{cr}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{s}[n]}$$

$$+ W_{s_{cr}}\vec{\chi}[n] \odot \vec{r}[n] \odot \frac{dG_c(z)}{dz}\Big\rfloor_{z=\vec{a}_{cr}[n]} + \vec{f}_{\psi}[n+1] \tag{173}$$

$$= \left(\left(\frac{\partial \vec{y}[n]}{\partial \vec{v}[n]}\right)^{T}\left(\frac{\partial E}{\partial \vec{y}[n]}\right) + \vec{f}_{\chi}[n+1]\right) \odot \vec{g}_{cr}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{s}[n]} + W_{s_{cr}}\left(\left(\frac{\partial \vec{y}[n]}{\partial \vec{v}[n]}\right)^{T}\left(\frac{\partial E}{\partial \vec{y}[n]}\right) + \vec{f}_{\chi}[n+1]\right) \odot \vec{r}[n] \odot \frac{dG_c(z)}{dz}\Big\rfloor_{z=\vec{a}_{cr}[n]} + \vec{f}_{\psi}[n+1] \tag{174}$$

According to Eqs. (154) and (155), both $\vec{\chi}[n]$ and $\vec{\psi}[n]$ depend on $\vec{\psi}[n+1]$. Hence, we can follow the approach in Section 4 to analyze the dependence of $\vec{\psi}[n]$ on $\left\langle \vec{\psi}[k] \right\rangle \Big\rfloor_{n<k\leq K-1}$ in order to gauge the sensitivity of the LSTM system to factors conducive to gradient decay. Applying the change of indices, $n \longrightarrow k-1$, and the chain rule to Eq. (174) yields[28]:

$$\frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]} = \left(\frac{\partial \vec{\psi}[k-1]}{\partial \vec{f}_{\chi}[k]}\right)\left(\frac{\partial \vec{f}_{\chi}[k]}{\partial \vec{\psi}[k]}\right) + \left(\frac{\partial \vec{\psi}[k-1]}{\partial \vec{f}_{\psi}[k]}\right)\left(\frac{\partial \vec{f}_{\psi}[k]}{\partial \vec{\psi}[k]}\right) \tag{175}$$

$$\begin{aligned} &= \left(\frac{\partial \vec{\psi}[k-1]}{\partial \vec{f}_{\chi}[k]}\right)\left\{\left(\frac{\partial \vec{f}_{\chi}[k]}{\partial \vec{\alpha}_{cu}[k]}\right)\left(\frac{\partial \vec{\alpha}_{cu}[k]}{\partial \vec{\psi}[k]}\right) + \left(\frac{\partial \vec{f}_{\chi}[k]}{\partial \vec{\alpha}_{cs}[k]}\right)\left(\frac{\partial \vec{\alpha}_{cs}[k]}{\partial \vec{\psi}[k]}\right) + \left(\frac{\partial \vec{f}_{\chi}[k]}{\partial \vec{\alpha}_{du}[k]}\right)\left(\frac{\partial \vec{\alpha}_{du}[k]}{\partial \vec{\psi}[k]}\right)\right\} \\ &+ \left(\frac{\partial \vec{\psi}[k-1]}{\partial \vec{f}_{\psi}[k]}\right)\left\{\left(\frac{\partial \vec{f}_{\psi}[k]}{\partial \vec{\alpha}_{cu}[k]}\right)\left(\frac{\partial \vec{\alpha}_{cu}[k]}{\partial \vec{\psi}[k]}\right) + \left(\frac{\partial \vec{f}_{\psi}[k]}{\partial \vec{\alpha}_{cs}[k]}\right)\left(\frac{\partial \vec{\alpha}_{cs}[k]}{\partial \vec{\psi}[k]}\right) + \operatorname{diag}\left[\vec{g}_{cs}[k]\right]\right\} \end{aligned} \tag{176}$$

$$\begin{aligned} &= \left(\operatorname{diag}\left[\vec{g}_{cr}[k-1] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{s}[k-1]}\right] + W_{s_{cr}}\operatorname{diag}\left[\vec{r}[k-1] \odot \frac{dG_c(z)}{dz}\Big\rfloor_{z=\vec{a}_{cr}[k-1]}\right]\right) \\ &\times \left\{W_{v_{cu}}\operatorname{diag}\left[\vec{u}[k] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cu}[k]}\right] + W_{v_{cs}}\operatorname{diag}\left[\vec{s}[k-1] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cs}[k]}\right] + W_{v_{du}}\operatorname{diag}\left[\vec{g}_{cu}[k] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{du}[k]}\right]\right\} \\ &+ \left(W_{s_{cu}}\operatorname{diag}\left[\vec{u}[k] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cu}[k]}\right] + W_{s_{cs}}\operatorname{diag}\left[\vec{s}[k-1] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cs}[k]}\right] + \operatorname{diag}\left[\vec{g}_{cs}[k]\right]\right) \end{aligned} \tag{177}$$

$$= \operatorname{diag}\left[\vec{u}[k] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big\rfloor_{z=\vec{a}_{cu}[k]}\right]$$

[26] Depending on the application, the batch sizes typically range between 16 and 128 segments.

[27] Regularization, while outside of the scope of the present article, is an essential aspect of machine learning model training process [78].

[28] The notation, $\operatorname{diag}\left[\vec{z}\right]$, represents a diagonal matrix, in which the elements of the vector, $\vec{z}$, occupy the main diagonal.

$$\times \left\{ W_{v_{cu}} \left(\mathrm{diag}\left[\vec{g}_{cr}[k-1] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{s}[k-1]} \right] W_{s_{cr}} \mathrm{diag}\left[\vec{r}[k-1] \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cr}[k-1]} \right] \right) + W_{s_{cu}} \right\}$$
$$+ \mathrm{diag}\left[\vec{s}[k-1] \odot \frac{dG_c(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{cs}[k]} \right]$$
$$\times \left\{ W_{v_{cs}} \left(\mathrm{diag}\left[\vec{g}_{cr}[k-1] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{s}[k-1]} \right] + W_{s_{cr}} \mathrm{diag}\left[\vec{r}[k-1] \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cr}[k-1]} \right] \right) + W_{s_{cs}} \right\}$$
$$+ \mathrm{diag}\left[\vec{g}_{cu}[k] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{du}[k]} \right]$$
$$\times \left\{ W_{v_{du}} \left(\mathrm{diag}\left[\vec{g}_{cr}[k-1] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{s}[k-1]} \right] + W_{s_{cr}} \mathrm{diag}\left[\vec{r}[k-1] \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cr}[k-1]} \right] \right) \right\}$$
$$+ \mathrm{diag}\left[\vec{g}_{cs}[k] \right] \tag{178}$$
$$= Q\left(k-1, k; \tilde{\Theta}\right) + \mathrm{diag}\left[\vec{g}_{cs}[k] \right] \tag{179}$$

where $\tilde{\Theta} = \left\{ W_{s_{cu}}, W_{v_{cu}}, W_{s_{cs}}, W_{v_{cs}}, W_{s_{cr}}, W_{v_{du}} \right\}$, and $Q\left(k-1, k; \tilde{\Theta}\right)$ subsumes all the terms in $\frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]}$, excluding $\mathrm{diag}\left[\vec{g}_{cs}[k] \right]$.

Extrapolating $\frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]}$ from the step with the index, n, to the step with the index, $l \leq K-1$, where $l \gg n$, gives:

$$\frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} = \prod_{k=n+1}^{l} \frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]} \tag{180}$$

Assuming that the issue of "exploding gradients" is handled as a separate undertaking, the present focus is on the effectiveness of the LSTM network at assuaging the "vanishing gradients" problem. If the value of the total partial derivative of the objective function, E, with respect to the state signal, $\vec{s}[l]$, at the index, $l \leq K-1$, where $l \gg n$, is considered to be an impulse of the error gradient, then Eq. (180) computes the fractional amount of this corrective stimulus that did not dissipate across the large number $(l-n)$ of steps and is preserved in $\vec{\psi}[n]$, thereby able to contribute to updating the model parameters.

The propensity of the LSTM system toward diminishing error gradients during training can be assessed by evaluating the different modes of Eq. (179) that can cause $\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\| \approx 0$ in Eq. (180) when $l-n$ is large. A sufficient condition for driving the residual $\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\|$ to zero is maintaining $\left\| \frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]} \right\| < 1$ at each step with the index, k. There are three possibilities for this outcome:

- $Q\left(k-1, k; \tilde{\Theta}\right) = \left[0\right]$ and $\vec{g}_{cs}[k] = \vec{0}$ for all values of the step index, k. This is the case of the network being perpetually "at rest" (i.e., in a trivial state), which is not interesting from the practical standpoint.
- $\vec{g}_{cs}[k] \approx \vec{1}$ and $Q\left(k-1, k; \tilde{\Theta}\right) = -\mathrm{diag}\left[\vec{g}_{cs}[k] \right]$; in other words, $Q\left(k-1, k; \tilde{\Theta}\right)$ and $\mathrm{diag}\left[\vec{g}_{cs}[k] \right]$ "cancel each other out" for some value of the step index, k. However, satisfying this condition would require a very careful orchestration of all signals, which is highly unlikely to occur in practice, making this pathological case unrealistic.
- The spectral radius of $\left[Q\left(k-1, k; \tilde{\Theta}\right) + \mathrm{diag}\left[\vec{g}_{cs}[k] \right] \right]$ in Eq. (179) is less than unity for all values of the step index, k. In this situation, the error gradient will degrade to negligible levels after a sufficiently large number of steps. Nevertheless, this behavior would not be due to a degenerate mode of the system, but as a consequence of the particular patterns, occurring in the training data. In other words, some dependencies are naturally short-range.

For all remaining cases, the magnitude of $\frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]}$ is governed by the triangle inequality:

$$\left\| \frac{\partial \vec{\psi}[k-1]}{\partial \vec{\psi}[k]} \right\| \leq \left\| Q\left(k-1, k; \tilde{\Theta}\right) \right\| + \left\| \mathrm{diag}\left[\vec{g}_{cs}[k] \right] \right\| \tag{181}$$

The most emblematic regime of the LSTM network arises when $\left\| Q\left(k-1, k; \tilde{\Theta}\right) \right\| < 1$. Examining the terms in Eq. (178) exposes multiple ways of restricting signals and parameters that would create favorable circumstances for this to hold. The following list prescribes several plausible alternatives (all conditions in each arrangement must be satisfied)[29]:

- $\left\| W_{s_{cu}} \right\| < \frac{1}{2}$, $\left\| W_{v_{cu}} \right\| < \frac{1}{2}$, $\left\| W_{s_{cs}} \right\| < \frac{1}{2}$, $\left\| W_{v_{cs}} \right\| < \frac{1}{2}$, $\left\| W_{v_{du}} \right\| < 1$
- the state signal saturates the readout data warping function, $\left\| W_{s_{cr}} \right\| < \frac{1}{2}$, $\left\| W_{s_{cu}} \right\| < \frac{1}{2}$, $\left\| W_{s_{cs}} \right\| < \frac{1}{2}$
- the state signal saturates the readout data warping function, the accumulation signal for the control readout gate saturates its control warping function, $\left\| W_{s_{cu}} \right\| < \frac{1}{2}$, $\left\| W_{s_{cs}} \right\| < \frac{1}{2}$
- the control readout gate is turned off, $\left\| W_{s_{cu}} \right\| < \frac{1}{2}$, $\left\| W_{s_{cs}} \right\| < \frac{1}{2}$
- the accumulation signals for the control update gate and the control state gate saturate their respective control warping functions, the update candidate accumulation signal saturates the update candidate data warping function
- the control update gate is turned off, the control state gate is turned off

Since the difference between the step indices, $l-n$, is large when the network is trained to represent long-range dependencies, the powers of the $Q\left(k-1, k; \tilde{\Theta}\right)$ terms become negligible, ultimately leading to:

$$\left\| \frac{\partial \vec{\psi}[n]}{\partial \vec{\psi}[l]} \right\| \sim \prod_{k=n+1}^{l} \left\| \mathrm{diag}\left[\vec{g}_{cs}[k] \right] \right\| \leqslant 1 \tag{182}$$

Unlike $Q\left(k-1, k; \tilde{\Theta}\right)$, $\mathrm{diag}\left[\vec{g}_{cs}[k] \right]$ in Eq. (179) has no attenuating factors (the multiplier of $\vec{g}_{cs}[n+1] \odot \vec{\psi}[n+1]$ in Eq. (155) is the identity matrix). As long as the elements of $\vec{g}_{cs}[n]$ are fractions, the error gradient will naturally decay. However, if the model is trained to saturate $\vec{g}_{cs}[n]$ at $\vec{1}$, then the error gradient is recirculated through Constant Error Carousel.

7. Extensions to the Vanilla LSTM network

Since its invention, many variants and extensions of the original LSTM network model have been researched and utilized in practice. In this section, we will evolve the Vanilla LSTM architecture, derived in Section 5 and explained in depth in Section 6, along three avenues. Based on the analysis in Section 2 as well as the discussions in Sections 5 and 6.9, we will expand the

[29] Note that all data signals in Eq. (178) ($\vec{u}$, $\vec{s}$, and $\vec{r}$) have the dynamic range of 2, because they are at the output node of the warping function, $G_d(\vec{z})$, which is the hyperbolic tangent. Hence, for the respective term to have the norm of < 1, the associated parameter matrices must have the norm of $< \frac{1}{2}$.

input from consisting of a single sample to combining multiple samples within a small context window. In addition, as proposed in Section 5, we will introduce a new gate for controlling this richer input signal. Besides these two novel extensions, we will also include the "recurrent projection layer" in the augmented model, because it proved to be advantageous in certain sequence modeling applications [24].

7.1. External input context windows

We will represent the external input context windows by linear filters that have matrix-valued coefficients and operate on the sequence of input samples along the dimension of the steps of the sequence produced by unrolling the LSTM cell. In Eqs. (126)–(129), the matrix–vector products, $W_{x_{cu}}\vec{x}[n]$, $W_{x_{cs}}\vec{x}[n]$, $W_{x_{cr}}\vec{x}[n]$, and $W_{x_{du}}\vec{x}[n]$, respectively, which involve a single input sample, $\vec{x}[n]$, will be replaced by the convolutions of the context window filters, $W_{x_{cu}}[n]$, $W_{x_{cs}}[n]$, $W_{x_{cr}}[n]$, and $W_{x_{du}}[n]$, respectively, with the input signal, $\vec{x}[n]$, thereby involving all input samples within the context window in the computation of the respective accumulation signal. We choose the context window filters to be non-causal (i.e., with the non-zero coefficients defined only for $n \leq 0$). This will enable the accumulation signals to utilize the input samples from the "future" steps of the unrolled LSTM cell without excessively increasing the number of parameters to be learned, since the input samples from the "past" steps will be already absorbed by the state signal, $\vec{s}[n]$, due to recurrence. After making these substitutions, Eqs. (126)–(129) become:

$$\vec{a}_{cu}[n] = W_{x_{cu}}[n] * \vec{x}[n] + W_{s_{cu}}\vec{s}[n-1] + W_{v_{cu}}\vec{v}[n-1] + \vec{b}_{cu} \tag{183}$$

$$\vec{a}_{cs}[n] = W_{x_{cs}}[n] * \vec{x}[n] + W_{s_{cs}}\vec{s}[n-1] + W_{v_{cs}}\vec{v}[n-1] + \vec{b}_{cs} \tag{184}$$

$$\vec{a}_{cr}[n] = W_{x_{cr}}[n] * \vec{x}[n] + W_{s_{cr}}\vec{s}[n] + W_{v_{cr}}\vec{v}[n-1] + \vec{b}_{cr} \tag{185}$$

$$\vec{a}_{du}[n] = W_{x_{du}}[n] * \vec{x}[n] + W_{v_{du}}\vec{v}[n-1] + \vec{b}_{du} \tag{186}$$

To examine the convolutional terms in more detail, let every context window filter, $W_x[n]$ (with the respective subscript), have L non-zero matrix-valued terms. For example, if $L = 4$, then $W_x[0]$, $W_x[-1]$, $W_x[-2]$, and $W_x[-3]$ will be non-zero.[30]

By the definition of the discrete convolution,

$$W_x[n] * \vec{x}[n] = \sum_{l=-L+1}^{0} W_x[l]\vec{x}[n-l] = \sum_{l=0}^{L-1} W_x[-l]\vec{x}[n+l] \tag{187}$$

In the above example, the result on the left hand side of Eq. (187) for each step with the index, n, will be influenced by the window spanning 4 input samples: $\vec{x}[0]$, $\vec{x}[1]$, $\vec{x}[2]$, and $\vec{x}[3]$.

If we redefine $W_x[n]$ to be non-zero for $n \geq 0$, then Eq. (187) simplifies to:

$$W_x[n] * \vec{x}[n] = \sum_{l=0}^{L-1} W_x[l]\vec{x}[n+l] \tag{188}$$

The dependence of the left hand side of Eq. (188) on the input samples from the "future" steps of the unrolled LSTM cell is readily apparent from the expression for the convolution sum on the right hand side of Eq. (188).

By taking advantage of the available input samples within a small window surrounding each step of the sequence, the system can learn to "discern" the context in which the given step occurs. The inspiration for this "look-ahead" extension comes from the way people sometimes find it beneficial to read forward to the end of the sentence in order to better understand a phrase occurring in the earlier part of the sentence. It would be interesting to explore the relative trade-offs between the cost of adding a small number of parameter matrices to Θ so as to accommodate the input context windows with the computational burden of training a bi-directional LSTM network [72], and to compare the performance of the two architectures on several data sets.[31]

7.2. Recurrent projection layer

Another modification of the Vanilla LSTM cell redefines the cell's value signal to be the product of an additional matrix of weights[32] with the LSTM cell's value signal from Eq. (136). To insert the recurrent projection layer into the Vanilla LSTM cell, we adjust Eq. (136) as follows:

$$\vec{q}[n] = \vec{g}_{cr}[n] \odot \vec{r}[n] \tag{189}$$

$$\vec{v}[n] = W_{q_{dr}}\vec{q}[n] \tag{190}$$

where $W_{q_{dr}}$ implements the recurrent projection layer,[33] and the intermediate cell quantity, $\vec{q}[n]$ (which we will call the cell's "qualifier" signal), replaces what used to be the cell's value signal in the Vanilla LSTM cell. The new value signal of the cell from Eq. (190) will now be used for computing the accumulation signals in Eqs. (183)–(186).

Let d_v denote the dimensionality of the observable value signal of the cell; then $\vec{v} \in \mathbb{R}^{d_v}$ and $W_{q_{dr}} \in \mathbb{R}^{d_v \times d_s}$. The degree to which the dimensionality reduction of the cell's value signal can be tolerated for the given application directly contributes to speeding up the training phase of the system. By allowing $d_v < d_s$, the matrix multipliers of all the terms involving $\vec{v}[n-1]$, which dominate Eqs. (183)–(186) (or Eqs. (126)–(129) in the absence of the external input context windows), will contain correspondingly fewer columns. In contrast, in the Vanilla LSTM cell as covered in Section 6 (i.e., without the recurrent projection layer), d_v must equal d_s, since $\vec{v}[n]$ is on the same data path as $\vec{s}[n]$, with no signal transformations along the data path between them. Hence, the addition of the recurrent projection layer to the Vanilla LSTM cell brings about the flexibility of trading off the representational capacity of the cell with the computational cost of learning its parameters [24].

7.3. Controlling external input with a new gate

Section 5 argues that the two components of the data update accumulation node, $\vec{a}_{du}[n]$, in the Vanilla LSTM cell are not treated the same way from the standpoint of control. While the readout candidate signal is throttled by the control readout gate, $\vec{g}_{cr}[n]$, the external input is always injected at the full 100% contribution of its signal strength. This is not as much of an issue for the control accumulation nodes ($\vec{a}_{cu}[n]$, $\vec{a}_{cs}[n]$, and $\vec{a}_{cr}[n]$), because they influence the scaling of the data signals, not the relative mixing of the data signals themselves. However, since the data update accumulation node, $\vec{a}_{du}[n]$, is directly in the path of the cell's state signal, the ability to regulate both components of

[30] As the members of the expanded LSTM model's parameter set, Θ, the new matrices and bias vectors are learned during the training phase.

[31] The non-causal input context windows can be readily employed as part of the bi-directional RNN or LSTM network. The number of additional parameter matrices to support input convolutions will double compared to the uni-directional case, because the samples covered by the windows are situated on the positive side along the direction of the sequence.

[32] This new matrix of weights, to be learned as part of training, is known as the "recurrent projection layer" [24].

[33] In our nomenclature, the matrix of weights, $W_{q_{dr}}$, links the cell's "qualifier" signal, $\vec{q}[n]$, with the cell's value signal, $\vec{v}[n]$, along the readout data path of the cell.

$\vec{a}_{du}[n]$ can improve the cell's capacity to adapt to the nuances and fluctuations in the training data. For instance, the control readout gate, $\vec{g}_{cr}[n]$, can diminish the effect of the cell's readout signal from the adjacent step in favor of making the external input signal at the given step more prominent in the make up of the cell's state signal. Likewise, having the additional flexibility to fine-tune the external input component of $\vec{a}_{du}[n]$ at the same granularity as its readout component (i.e., at the level of the individual step with the index, n) would provide a means for training the LSTM cell to suppress interference due to noisy or spurious input samples.

As a mechanism for adjusting the contribution of the external input, Eq. (106) introduced the control input gate, $\vec{g}_{cx}[n]$, which we will apply to the convolution term in Eq. (186). Analogously to the other gates, $\vec{g}_{cx}[n]$ is computed by taking the warping function for control, given by Eq. (121), of the accumulation signal for controlling the input, element by element.

From Section 6.6 and Eq. (186), the data update accumulation node, $\vec{a}_{du}[n]$, is followed by the data warping function that produces the update candidate signal, $\vec{u}[n]$, of the cell at the present step with the index, n. The new control input gate, $\vec{g}_{cx}[n]$, will throttle $W_{x_{du}}[n] * \vec{x}[n]$, the term representing the composite external input signal in Eq. (186) for $\vec{a}_{du}[n]$. Letting $\vec{\xi}_{x_{du}}[n] \equiv W_{x_{du}}[n] * \vec{x}[n]$, where $\vec{\xi}_{x_{du}}[n]$ denotes the composite external input signal for the data update accumulation node, $\vec{a}_{du}[n]$, this gating operation will be accomplished through the element-wise multiplication, $\vec{g}_{cx}[n] \odot \vec{\xi}_{x_{du}}[n]$, the method used by all the other gates to control the fractional amount of their designated data signals.

The equations for accommodating the new control input gate, $\vec{g}_{cx}[n]$, as part of the LSTM cell design as well as the equation for the data update accumulation node, $\vec{a}_{du}[n]$, modified to take advantage of this new gate, are provided below:

$$\vec{\xi}_{x_{cx}}[n] = W_{x_{cx}}[n] * \vec{x}[n] \tag{191}$$
$$\vec{a}_{cx}[n] = \vec{\xi}_{x_{cx}}[n] + W_{s_{cx}}\vec{s}[n-1] + W_{v_{cx}}\vec{v}[n-1] + \vec{b}_{cx} \tag{192}$$
$$\vec{g}_{cx}[n] = G_c(\vec{a}_{cx}[n]) \tag{193}$$
$$\vec{\xi}_{x_{du}}[n] = W_{x_{du}}[n] * \vec{x}[n] \tag{194}$$
$$\vec{a}_{du}[n] = \vec{g}_{cx}[n] \odot \vec{\xi}_{x_{du}}[n] + W_{v_{du}}\vec{v}[n-1] + \vec{b}_{du} \tag{195}$$

where the additional parameters, needed to characterize the accumulation node, $\vec{a}_{cx}[n]$, of the new control input gate, $\vec{g}_{cx}[n]$, have the following interpretation:

- $W_{x_{cx}}[l] \in \mathbb{R}^{d_x \times d_x}$ — the matrices of weights (for $0 \le l \le L-1$) connecting the input signal, $\vec{x}[n+l]$, at the step with the index, $n+l$, to the "control input" accumulation node, $\vec{a}_{cx}[n]$, of the cell at the present step with the index, n
- $W_{s_{cx}} \in \mathbb{R}^{d_x \times d_s}$ — the matrix of weights connecting the state signal, $\vec{s}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control input" accumulation node, $\vec{a}_{cx}[n]$, of the cell at the present step with the index, n
- $W_{v_{cx}} \in \mathbb{R}^{d_x \times d_v}$ — the matrix of weights connecting the externally-accessible (observable) value signal, $\vec{v}[n-1]$, at the adjacent lower-indexed step with the index, $n-1$, to the "control input" accumulation node, $\vec{a}_{cx}[n]$, of the cell at the present step with the index, n
- $\vec{b}_{cx} \in \mathbb{R}^{d_x}$ — the vector of bias elements for the "control input" accumulation node, $\vec{a}_{cx}[n]$, of the cell at the present step with the index, n
- $W_{x_{du}}[l] \in \mathbb{R}^{d_s \times d_x}$ — the matrices of weights (for $0 \le l \le L-1$) connecting the input signal, $\vec{x}[n+l]$, at the step with the index, $n+l$, to the "data update" accumulation node, $\vec{a}_{du}[n]$, of the cell at the present step with the index, n

7.4. Augmented LSTM system equations ("forward pass")

We are now ready to assemble the equations for the Augmented LSTM system by enhancing the Vanilla LSTM network with the new functionality, presented earlier in this section — the recurrent projection layer, the non-causal input context windows, and the input gate:

$$\vec{\xi}_{x_{cu}}[n] = W_{x_{cu}}[n] * \vec{x}[n] \tag{196}$$
$$\vec{a}_{cu}[n] = \vec{\xi}_{x_{cu}}[n] + W_{s_{cu}}\vec{s}[n-1] + W_{v_{cu}}\vec{v}[n-1] + \vec{b}_{cu} \tag{197}$$
$$\vec{g}_{cu}[n] = G_c(\vec{a}_{cu}[n]) \tag{198}$$
$$\vec{\xi}_{x_{cs}}[n] = W_{x_{cs}}[n] * \vec{x}[n] \tag{199}$$
$$\vec{a}_{cs}[n] = \vec{\xi}_{x_{cs}}[n] + W_{s_{cs}}\vec{s}[n-1] + W_{v_{cs}}\vec{v}[n-1] + \vec{b}_{cs} \tag{200}$$
$$\vec{g}_{cs}[n] = G_c(\vec{a}_{cs}[n]) \tag{201}$$
$$\vec{\xi}_{x_{cr}}[n] = W_{x_{cr}}[n] * \vec{x}[n] \tag{202}$$
$$\vec{a}_{cr}[n] = \vec{\xi}_{x_{cr}}[n] + W_{s_{cr}}\vec{s}[n] + W_{v_{cr}}\vec{v}[n-1] + \vec{b}_{cr} \tag{203}$$
$$\vec{g}_{cr}[n] = G_c(\vec{a}_{cr}[n]) \tag{204}$$
$$\vec{\xi}_{x_{cx}}[n] = W_{x_{cx}}[n] * \vec{x}[n] \tag{205}$$
$$\vec{a}_{cx}[n] = \vec{\xi}_{x_{cx}}[n] + W_{s_{cx}}\vec{s}[n-1] + W_{v_{cx}}\vec{v}[n-1] + \vec{b}_{cx} \tag{206}$$
$$\vec{g}_{cx}[n] = G_c(\vec{a}_{cx}[n]) \tag{207}$$
$$\vec{\xi}_{x_{du}}[n] = W_{x_{du}}[n] * \vec{x}[n] \tag{208}$$
$$\vec{a}_{du}[n] = \vec{g}_{cx}[n] \odot \vec{\xi}_{x_{du}}[n] + W_{v_{du}}\vec{v}[n-1] + \vec{b}_{du} \tag{209}$$
$$\vec{u}[n] = G_d(\vec{a}_{du}[n]) \tag{210}$$
$$\vec{s}[n] = \vec{g}_{cs}[n] \odot \vec{s}[n-1] + \vec{g}_{cu}[n] \odot \vec{u}[n] \tag{211}$$
$$\vec{r}[n] = G_d(\vec{s}[n]) \tag{212}$$
$$\vec{q}[n] = \vec{g}_{cr}[n] \odot \vec{r}[n] \tag{213}$$
$$\vec{v}[n] = W_{q_{dr}}\vec{q}[n] \tag{214}$$

with the dimensions of the parameters adjusted to take into account the recurrent projection layer:
$W_{x_{cu}}[l] \in \mathbb{R}^{d_s \times d_x}$, $W_{s_{cu}} \in \mathbb{R}^{d_s \times d_s}$, $W_{v_{cu}} \in \mathbb{R}^{d_s \times d_v}$, $\vec{b}_{cu} \in \mathbb{R}^{d_s}$,
$W_{x_{cs}}[l] \in \mathbb{R}^{d_s \times d_x}$, $W_{s_{cs}} \in \mathbb{R}^{d_s \times d_s}$, $W_{v_{cs}} \in \mathbb{R}^{d_s \times d_v}$, $\vec{b}_{cs} \in \mathbb{R}^{d_s}$,
$W_{x_{cr}}[l] \in \mathbb{R}^{d_s \times d_x}$, $W_{s_{cr}} \in \mathbb{R}^{d_s \times d_s}$, $W_{v_{cr}} \in \mathbb{R}^{d_s \times d_v}$, $\vec{b}_{cr} \in \mathbb{R}^{d_s}$,
$W_{x_{cx}}[l] \in \mathbb{R}^{d_x \times d_x}$, $W_{s_{cx}} \in \mathbb{R}^{d_x \times d_s}$, $W_{v_{cx}} \in \mathbb{R}^{d_x \times d_v}$, $\vec{b}_{cx} \in \mathbb{R}^{d_x}$,
$W_{x_{du}}[l] \in \mathbb{R}^{d_s \times d_x}$, $W_{v_{du}} \in \mathbb{R}^{d_s \times d_v}$, $\vec{b}_{du} \in \mathbb{R}^{d_s}$, and $W_{q_{dr}} \in \mathbb{R}^{d_v \times d_s}$,
where $0 \le l \le L-1$ and $d_v \le d_s$.
The schematic diagram of the Augmented LSTM cell appears in Fig. 10.

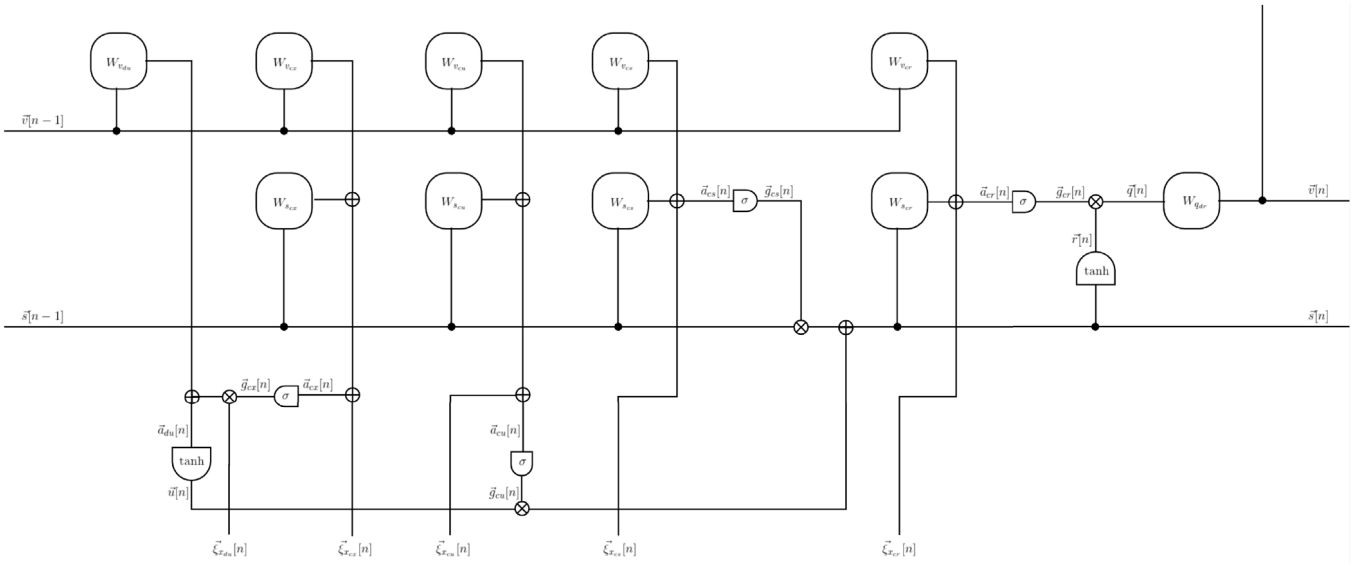
Combining all the matrix and vector parameters of the Augmented LSTM cell, described by Eqs. (196)–(214), into:

$$\Theta \equiv \left\{ \begin{array}{llll} W_{x_{cu}}[l], & W_{s_{cu}}, & W_{v_{cu}}, & \vec{b}_{cu}, \\ W_{x_{cs}}[l], & W_{s_{cs}}, & W_{v_{cs}}, & \vec{b}_{cs}, \\ W_{x_{cr}}[l], & W_{s_{cr}}, & W_{v_{cr}}, & \vec{b}_{cr}, \\ W_{x_{cx}}[l], & W_{s_{cx}}, & W_{v_{cx}}, & \vec{b}_{cx}, \\ W_{x_{du}}[l], & & W_{v_{du}}, & \vec{b}_{du}, \\ & & W_{q_{dr}} & \end{array} \right\} \tag{215}$$

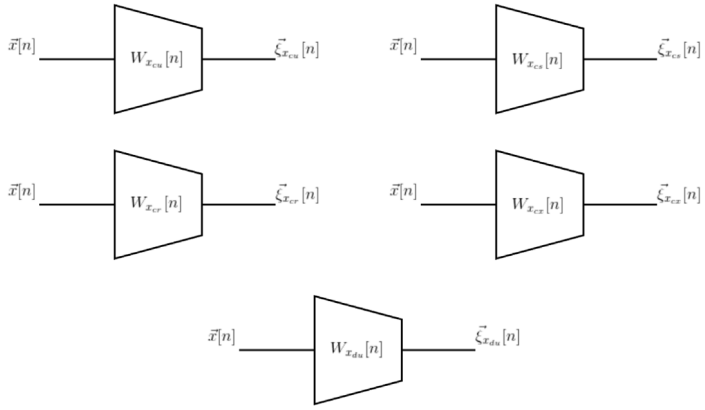
completes the definition of the inference phase (forward pass) of the Augmented LSTM cell.

7.5. Augmented LSTM system derivatives: Backward pass

The equations for training the Augmented LSTM cell, unrolled for K steps, using BPTT, are obtained by adopting the same method as was used for the Vanilla LSTM cell in Section 6.9. We rely on the same border and intermediate total partial derivatives, appearing in Eqs. (139)–(143), with the addition of the total



(a) Augmented LSTM network cell. The bias parameters, $\vec{b}$, have been omitted from the figure for brevity. They can be assumed to be included without the loss of generality by appending an additional element, always set to 1, to the input signal vector, $\vec{x}[n]$, and increasing the row dimensions of all corresponding weight matrices by 1. To minimize clutter in the diagram, the context windows are shown separately in Figure 10b below).



(b) The non-causal context windows, implemented as the convolutions (along the dimension of the sequence index, n) of the external input, $\vec{x}[n]$, with linear filters, whose coefficients are matrix-valued. The operations in this figure compute the quantities of the form $\vec{\xi}_x[n] = W_x[n] * \vec{x}[n]$, corresponding to their respective subscripts.

Fig. 10. Augmented LSTM network cell system schematics.

partial derivative of the objective function, E, with respect to the qualifier signal, $\vec{q}[n]$:

$$\vec{\beta}[n] \equiv \vec{\nabla}_{\vec{q}[n]} E = \frac{\partial E}{\partial \vec{q}[n]} \tag{216}$$

which is an intermediate total partial derivative that reflects the insertion of the projection layer into the cell's data path. Applying the chain rule to the Augmented LSTM cell, defined by Eqs. (196)–(214), and judiciously utilizing all of these border and intermediate total partial derivatives, yields the backward pass equations for the Augmented LSTM cell:

$$\vec{\chi}[n] = \left(\frac{\partial \vec{y}[n]}{\partial \vec{v}[n]}\right)^T \left(\frac{\partial E}{\partial \vec{y}[n]}\right) + \vec{f}_{\chi}[n+1] \tag{217}$$

$$\vec{\beta}[n] = \left(\frac{\partial \vec{v}[n]}{\partial \vec{q}[n]}\right)^T \left(\frac{\partial E}{\partial \vec{v}[n]}\right) = W_{q_{dr}}^T \vec{\chi}[n] \tag{218}$$

$$\vec{\rho}[n] = \left(\frac{\partial \vec{q}[n]}{\partial \vec{r}[n]}\right)^T \left(\frac{\partial E}{\partial \vec{q}[n]}\right) = (\vec{\nabla}_{\vec{q}[n]} E) \odot \vec{g}_{cr}[n] = \vec{\beta}[n] \odot \vec{g}_{cr}[n]$$
$$= W_{q_{dr}}^T \vec{\chi}[n] \odot \vec{g}_{cr}[n] \tag{219}$$

$$\vec{\gamma}_{cr}[n] = \frac{\partial E}{\partial \vec{q}[n]} \frac{\partial \vec{q}[n]}{\partial \vec{g}_{cr}[n]} = (\vec{\nabla}_{\vec{q}[n]} E) \odot \vec{r}[n] = \vec{\beta}[n] \odot \vec{r}[n]$$
$$= W_{q_{dr}}^T \vec{\chi}[n] \odot \vec{r}[n] \tag{220}$$

$$\vec{\alpha}_{cr}[n] = \vec{\gamma}_{cr}[n] \odot \frac{\partial \vec{g}_{cr}[n]}{\partial \vec{a}_{cr}[n]} = \vec{\gamma}_{cr}[n] \odot \frac{dG_c(z)}{dz}\Big|_{z=\vec{a}_{cr}[n]}$$
$$= W_{q_{dr}}^T \vec{\chi}[n] \odot \vec{r}[n] \odot \frac{dG_c(z)}{dz}\Big|_{z=\vec{a}_{cr}[n]} \tag{221}$$

$$\vec{\psi}[n] = \vec{\rho}[n] \odot \frac{\partial \vec{r}[n]}{\partial \vec{s}[n]} + \frac{\partial \vec{a}_{cr}[n]}{\partial \vec{s}[n]} \vec{\alpha}_{cr}[n] + \vec{f}_{\psi}[n+1]$$
$$= \vec{\rho}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big|_{z=\vec{s}[n]} + W_{s_{cr}} \vec{\alpha}_{cr}[n] + \vec{f}_{\psi}[n+1]$$
$$= W_{q_{dr}}^T \vec{\chi}[n] \odot \vec{g}_{cr}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}}\Big|_{z=\vec{s}[n]} + W_{s_{cr}} \vec{\alpha}_{cr}[n]$$
$$+ \vec{f}_{\psi}[n+1] \tag{222}$$

$$\vec{\alpha}_{cs}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{g}_{cs}[n]} \odot \frac{\partial \vec{g}_{cs}[n]}{\partial \vec{a}_{cs}[n]}$$
$$= \vec{\psi}[n] \odot \vec{s}[n-1] \odot \frac{dG_c(\vec{z})}{d\vec{z}}\Big|_{z=\vec{a}_{cs}[n]} \tag{223}$$

$$\vec{\alpha}_{cu}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{g}_{cu}[n]} \odot \frac{\partial \vec{g}_{cu}[n]}{\partial \vec{a}_{cu}[n]} = \vec{\psi}[n] \odot \vec{u}[n] \odot \frac{dG_c(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{cu}[n]} \tag{224}$$

$$\vec{\alpha}_{du}[n] = \vec{\psi}[n] \odot \frac{\partial \vec{s}[n]}{\partial \vec{u}[n]} \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{du}[n]} = \vec{\psi}[n] \odot \vec{g}_{cu}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{du}[n]} \tag{225}$$

$$\vec{\gamma}_{cx}[n] = \vec{\alpha}_{du}[n] \frac{\partial \vec{u}[n]}{\partial \vec{g}_{cx}[n]} = \vec{\alpha}_{du}[n] \odot \vec{\xi}_{x_{du}}[n] \tag{226}$$

$$\vec{\alpha}_{cx}[n] = \vec{\gamma}_{cx}[n] \odot \frac{\partial \vec{g}_{cx}[n]}{\partial \vec{a}_{cx}[n]} = \vec{\gamma}_{cx}[n] \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cx}[n]} = \vec{\alpha}_{du}[n] \odot \vec{\xi}_{x_{du}}[n] \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cr}[n]} = \vec{\psi}[n] \odot \vec{g}_{cu}[n] \odot \vec{\xi}_{x_{du}}[n] \odot \frac{dG_d(\vec{z})}{d\vec{z}} \rfloor_{z=\vec{a}_{du}[n]} \odot \frac{dG_c(z)}{dz} \rfloor_{z=\vec{a}_{cr}[n]} \tag{227}$$

where:

$$\vec{f}_{\chi}[n+1] = W_{v_{cu}} \vec{\alpha}_{cu}[n+1] + W_{v_{cs}} \vec{\alpha}_{cs}[n+1] + W_{v_{cr}} \vec{\alpha}_{cr}[n+1] + W_{v_{cx}} \vec{\alpha}_{cx}[n+1] + W_{v_{du}} \vec{\alpha}_{du}[n+1] \tag{228}$$

$$\vec{f}_{\psi}[n+1] = W_{s_{cu}} \vec{\alpha}_{cu}[n+1] + W_{s_{cs}} \vec{\alpha}_{cs}[n+1] + W_{s_{cx}} \vec{\alpha}_{cx}[n+1] + \vec{g}_{cs}[n+1] \odot \vec{\psi}[n+1] \tag{229}$$

Referring once again to the definition of the Augmented LSTM cell in Eqs. (196)–(214), we obtain:

$$\frac{\partial E}{\partial W_{x_{cu}}[l]}[n] = \vec{\alpha}_{cu}[n] \vec{x}^T[n+l] \tag{230}$$

$$\frac{\partial E}{\partial W_{s_{cu}}}[n] = \vec{\alpha}_{cu}[n] \vec{s}^T[n-1] \tag{231}$$

$$\frac{\partial E}{\partial W_{v_{cu}}}[n] = \vec{\alpha}_{cu}[n] \vec{v}^T[n-1] \tag{232}$$

$$\frac{\partial E}{\partial \vec{b}_{cu}}[n] = \vec{\alpha}_{cu}[n] \tag{233}$$

$$\frac{\partial E}{\partial W_{x_{cs}}[l]}[n] = \vec{\alpha}_{cs}[n] \vec{x}^T[n+l] \tag{234}$$

$$\frac{\partial E}{\partial W_{s_{cs}}}[n] = \vec{\alpha}_{cs}[n] \vec{s}^T[n-1] \tag{235}$$

$$\frac{\partial E}{\partial W_{v_{cs}}}[n] = \vec{\alpha}_{cs}[n] \vec{v}^T[n-1] \tag{236}$$

$$\frac{\partial E}{\partial \vec{b}_{cs}}[n] = \vec{\alpha}_{cs}[n] \tag{237}$$

$$\frac{\partial E}{\partial W_{x_{cr}}[l]}[n] = \vec{\alpha}_{cr}[n] \vec{x}^T[n+l] \tag{238}$$

$$\frac{\partial E}{\partial W_{s_{cr}}}[n] = \vec{\alpha}_{cr}[n] \vec{s}^T[n] \tag{239}$$

$$\frac{\partial E}{\partial W_{v_{cr}}}[n] = \vec{\alpha}_{cr}[n] \vec{v}^T[n-1] \tag{240}$$

$$\frac{\partial E}{\partial \vec{b}_{cr}}[n] = \vec{\alpha}_{cr}[n] \tag{241}$$

$$\frac{\partial E}{\partial W_{x_{cx}}[l]}[n] = \vec{\alpha}_{cx}[n] \vec{x}^T[n+l] \tag{242}$$

$$\frac{\partial E}{\partial W_{s_{cx}}}[n] = \vec{\alpha}_{cx}[n] \vec{s}^T[n-1] \tag{243}$$

$$\frac{\partial E}{\partial W_{v_{cx}}}[n] = \vec{\alpha}_{cx}[n] \vec{v}^T[n-1] \tag{244}$$

$$\frac{\partial E}{\partial \vec{b}_{cx}}[n] = \vec{\alpha}_{cx}[n] \tag{245}$$

$$\frac{\partial E}{\partial W_{x_{du}}[l]}[n] = \vec{\alpha}_{du}[n] \vec{x}^T[n+l] \tag{246}$$

$$\frac{\partial E}{\partial W_{v_{du}}}[n] = \vec{\alpha}_{du}[n] \vec{v}^T[n-1] \tag{247}$$

$$\frac{\partial E}{\partial \vec{b}_{du}}[n] = \vec{\alpha}_{du}[n] \tag{248}$$

$$\frac{\partial E}{\partial W_{q_{dr}}}[n] = \vec{\chi}[n] \vec{v}^T[n] \tag{249}$$

where $0 \leq l \leq L-1$ and $0 \leq n \leq K-1$. Arranged to parallel the structure of Θ, defined in Eq. (215), the total partial derivative of the objective function, E, with respect to the model parameters, Θ, at the step with the index, n, is:

$$\frac{\partial E}{\partial \Theta}[n] = \left\{ \begin{array}{cccc} \frac{\partial E}{\partial W_{x_{cu}[l]}}[n], & \frac{\partial E}{\partial W_{s_{cu}}}[n], & \frac{\partial E}{\partial W_{v_{cu}}}[n], & \frac{\partial E}{\partial \vec{b}_{cu}}[n], \\ \frac{\partial E}{\partial W_{x_{cs}[l]}}[n], & \frac{\partial E}{\partial W_{s_{cs}}}[n], & \frac{\partial E}{\partial W_{v_{cs}}}[n], & \frac{\partial E}{\partial \vec{b}_{cs}}[n], \\ \frac{\partial E}{\partial W_{x_{cr}[l]}}[n], & \frac{\partial E}{\partial W_{s_{cr}}}[n], & \frac{\partial E}{\partial W_{v_{cr}}}[n], & \frac{\partial E}{\partial \vec{b}_{cr}}[n], \\ \frac{\partial E}{\partial W_{x_{cx}[l]}}[n] & \frac{\partial E}{\partial W_{s_{cx}}}[n] & \frac{\partial E}{\partial W_{v_{cx}}}[n] & \frac{\partial E}{\partial \vec{b}_{cx}}[n] \\ \frac{\partial E}{\partial W_{x_{du}[l]}}[n], & & \frac{\partial E}{\partial W_{v_{du}}}[n], & \frac{\partial E}{\partial \vec{b}_{du}}[n] \\ & & \frac{\partial E}{\partial W_{q_{dr}}}[n] & \end{array} \right\}. \tag{250}$$

Finally, $\frac{dE}{d\Theta}$, the total derivative of the objective function, E, with respect to the model parameters, Θ, for the entire unrolled sequence is computed by Eq. (172). Aggregated over a batch of segments, $\frac{dE}{d\Theta}$ is plugged in to the Gradient Descent training algorithm for learning the model parameters, Θ.

8. Conclusions and future work

In this paper, we presented the fundamentals of the RNN and the LSTM network using a principled approach. Starting with the differential equations encountered in many branches of science and engineering, we showed that the canonical formulation of the RNN can be obtained by sampling delay differential equations used to model processes in physics, life sciences, and neural networks. We proceeded to obtain the standard RNN formulation by appropriately choosing the parameters of the canonical RNN equations and applying stability considerations. We then formally explained RNN unrolling within the framework of approximating an IIR system by an FIR model and proved the sufficient conditions for its applicability to learning sequences. Next, we presented the training of the standard RNN using Back Propagation Through Time, segueing to the review of the vanishing and exploding gradients, the well-known numerical difficulties, associated with training the standard RNN by Gradient Descent. We subsequently addressed the shortcomings of the standard RNN by morphing the canonical RNN system into the more robust LSTM network through a series of extensions and embellishments. In addition to the logical construction of the Vanilla LSTM network from the canonical RNN, we included a self-contained overview of the Vanilla LSTM network, complete with the specifications of all principal entities as well as clear, descriptive, yet concise, presentations of the forward pass and, importantly, the backward pass, without skipping any steps. The main contribution up to this point has been our unique pedagogical approach for analyzing the RNN and Vanilla LSTM systems from the Signal Processing perspective, a formal derivation of the RNN unrolling procedure, and a thorough treatment using a descriptive and meaningful

notation, aimed at demystifying the underlying concepts. Moreover, as an unexpected benefit of our analysis, we identified two novel extensions to the Vanilla LSTM network: the convolutional non-causal input context windows and the external input gate. We then augmented the equations for the LSTM cell with these extensions (along with the recurrent projection layer, previously introduced by another researcher team). Candidate recommendations for future work include implementing the Augmented LSTM system within a high-performance computing environment and benchmarking its efficacy in multiple practical scenarios. The use of the Augmented LSTM could potentially benefit the language representation subsystems used in question answering and in automating customer support. For these applications, it will be important to evaluate the performance impact, attributed to the non-causal input context windows, as compared to the different baselines, such the Vanilla LSTM network, the bi-directional LSTM network, the Transformer, and other state-of-the-art models. Also of particular relevance to this use case will be to measure the effectiveness of the external input gate in helping to eliminate the non-essential content from the input sequences. Finally, adopting the Augmented LSTM network to other practical domains and publishing the results is respectfully encouraged.

CRediT authorship contribution statement

Alex Sherstinsky: Conceptualization, Methodology, Formal analysis, Investigation, Resources, Writing - original draft, Writing - review & editing.

Acknowledgments

The author thanks Eugene Mandel for long-time collaboration and engaging discussions, which were instrumental in clarifying the key concepts, as well as for his encouragement and support. Tremendous gratitude is expressed to Tom Minka for providing helpful critique and valuable comments and to Mike Singer for reviewing the proposition and correcting errors in the proof. Big thanks go to Varuna Jayasiri, Eduardo G. Ponferrada, Flavia Sparacino, and Janet Cahn for proofreading the manuscript.

References

[1] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[2] Henry W. Lin, Max Tegmark, Criticality in formal languages and statistical physics, Entropy 19 (7) (2017) 299.
[3] Terence D. Sanger, Optimal unsupervised learning in a single-layer linear feedforward neural network, Neural Netw. 2 (1989) 459–473.
[4] Alex Sherstinsky, M-Lattice: A System for Signal Synthesis and Processing Based on Reaction-Diffusion (Ph.D. thesis), Massachusetts Institute of Technology, 1994.
[5] Qianli Liao, Tomaso A. Poggio, Bridging the gaps between residual learning, recurrent neural networks and visual cortex, CoRR abs/1604.03640 (2016).
[6] Eldad Haber, Lars Ruthotto, Stable architectures for deep neural networks, Inverse Problems 34 (1) (2017) 014004.
[7] Yiping Lu, Aoxiao Zhong, Quanzheng Li, Bin Dong, Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations, in: Jennifer G. Dy, Andreas Krause (Eds.), ICML, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 3282–3291.
[8] Lars Ruthotto, Eldad Haber, Deep neural networks motivated by partial differential equations, CoRR abs/1804.04272, (2018).
[9] Alex Sherstinsky, Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network, 2018, arxiv:1808.03314, 39 pages, 10 figures, 66 references..
[10] Alex Sherstinsky, Deriving the recurrent neural network definition and rnn unrolling using signal processing, NeurIPS 2018, in: Critiquing and Correcting Trends in Machine Learning Workshop at Neural Information Processing Systems, vol. 31, 2018, Organizers: Benjamin Bloem-Reddy, Brooks Paige, Matt J. Kusner, Rich Caruana, Tom Rainforth, and Yee Whye Teh.
[11] Marco Ciccone, Marco Gallieri, Jonathan Masci, Christian Osendorfer, Faustino J. Gomez, Nais-net: Stable deep networks from non-autonomous differential equations, in: Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicoló Cesa-Bianchi, Roman Garnett (Eds.), NeurIPS, 2018, pp. 3029–3039.
[12] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, Elliot Holtham, Reversible architectures for arbitrarily deep residual neural networks, in: Sheila A. McIlraith, Kilian Q. Weinberger (Eds.), AAAI, AAAI Press, 2018, pp. 2811–2818.
[13] Bo Chang, Minmin Chen, Eldad Haber, Ed H. Chi, AntisymmetricRNN: A dynamical system view on recurrent neural networks. in: International Conference on Learning Representations, 2019.
[14] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, David Duvenaud, Neural ordinary differential equations, in: Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicoló Cesa-Bianchi, Roman Garnett (Eds.), NeurIPS, 2018, pp. 6572–6583.
[15] Yulia Rubanova, T.Q. Chen Ricky, David Duvenaud, Latent odes for irregularly-sampled time series, 2019, arxiv:1907.03907.
[16] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber, LSTM: A search space Odyssey, CoRR abs/1503.04069 (2015).
[17] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Netw. 18 (5–6) (2005) 602–610.
[18] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional LSTM networks. in: Proc. Int. Joint Conf. on Neural Networks IJCNN 2005, 2005.
[19] Alex Graves, Supervised Sequence Labelling with Recurrent Neural Networks (Ph.D. thesis), Technical University Munich, 2008.
[20] Ilya Sutskever, James Martens, Geoffrey E. Hinton, Generating text with recurrent neural networks, in: Lise Getoor, Tobias Scheffer (Eds.), ICML, Omnipress, 2011, pp. 1017–1024.
[21] Martin Sundermeyer, Ralf Schlüter, Hermann Ney, LSTM neural networks for language modeling. in: Interspeech, 2012, 194–197.
[22] Alex Graves, Generating sequences with recurrent neural networks, CoRR abs/1308.0850 (2013).
[23] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
[24] Hasim Sak, Andrew W. Senior, Françoise Beaufays, Long short-term memory recurrent neural network architectures for large scale acoustic modeling, in: Haizhou Li, Helen M. Meng, Bin Ma, Engsiong Chng, Lei Xie (Eds.), Interspeech, ISCA, 2014, pp. 338–342.
[25] Zachary Chase Lipton, A critical review of recurrent neural networks for sequence learning, CoRR abs/1506.00019 (2015).
[26] Andrej Karpathy, The unreasonable effectiveness of recurrent neural networks, 2015, http://karpathy.github.io/2015/05/21/rnn-effectiveness.
[27] Christopher Olah, Understanding LSTM networks, 2015, http://colah.github.io/posts/2015-08-Underst{and}ing-LSTMs.
[28] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, Rabab K. Ward, Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval, CoRR abs/1502.06922 (2015).
[29] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, László Lukács, Marina Ganea, Peter Young, Vivek Ramavajjala, Smart reply: Automated response suggestion for email, CoRR abs/1606.04870 (2016).
[30] Jie Zhou, Ying Cao, Xuguang Wang, Peng Li, Wei Xu, Deep recurrent models with fast-forward connections for neural machine translation, CoRR abs/1606.04199 (2016).
[31] Paul Renvoisé, Machine learning spotlight i: Investigating recurrent neural networks, 2017, https://recast.ai/blog/ml-spotlight-rnn/.
[32] Edwin Chen, Exploring LSTMs, 2017, http://blog.echen.me/2017/05/30/exploring-lstms.
[33] Arun Mallya, LSTM forward and backward pass, 2017, http://arunmallya.github.io/writeups/nn/lstm/index.html.
[34] Arun Mallya, Introduction to RNNs, 2017, http://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf.
[35] Arun Mallya, Some RNN variants, 2017, http://slazebni.cs.illinois.edu/spring17/lec03_rnn.pdf.
[36] Varuna Jayasiri, Vanilla LSTM with numpy, 2017, http://blog.varunajayasiri.com/numpy_lstm.html.
[37] Hojjat Salehinejad, Julianne Baarbe, Sharan Sankar, Joseph Barfett, Errol Colak, Shahrokh Valaee, Recent advances in recurrent neural networks, CoRR abs/1801.01078 (2018).
[38] S.H. Strogatz, Nonlinear Dynamics and Chaos, Westview Press, Cambridge, MA, 1994.
[39] Yu Wang, A new concept using LSTM neural networks for dynamic system identification, in: ACC, IEEE, 2017, pp. 5324–5329.
[40] S. Grossberg, Recurrent neural networks, Scholarpedia 8 (2) (2013) 1888, revision #138057.

[41] J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Natl. Acad. Sci. 81 (1984) 3088–3092.
[42] Stephen Grossberg, Nonlinear neural networks: Principles, mechanisms, and architectures, Neural Netw. 1 (1988) 17–61.
[43] Alex Sherstinsky, Rosalind W. Picard, M-lattice: from morphogenesis to image processing, IEEE Trans. Image Process. 5 (7) (1996) 1137–1149.
[44] Yuliya Kyrychko, Stephen Hogan, On the use of delay equations in engineering applications, 16 (2010) 943–960.
[45] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087.
[46] Alex Sherstinsky, Rosalind W. Picard, On stability and equilibria of the M-Lattice, IEEE Trans. Circuit Syst. I 45 (4) (1998) 408–415.
[47] Oleksii Ostroverkhyi, Neural Network Processing of Multidimensional Signals (Ph.D. thesis), Kharkiv National University of Radioelectronics, 2010.
[48] M.I. Jordan, Serial Order: A Parallel Distributed Processing Approach, Technical Report ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, 1986.
[49] F.J. Pineda, Generalization of backpropagation to recurrent neural networks, Phys. Rev. Lett. 59 (19) (1987) 2229–2232.
[50] Fernando L. Pineda, Generalization of backpropagation to recurrent and higher order neural networks, in: Dana Z. Anderson (Ed.), Neural Information Processing Systems, American Institute of Physics, New York, 1987, pp. 602–611.
[51] Barak A. Pearlmutter, Learning state space trajectories in recurrent neural networks, Neural Comput. 1 (2) (1989) 263–269.
[52] J.L. Elman, Finding structure in time, Cogn. Sci. 14 (1990) 179–211.
[53] Barak A. Pearlmutter, Dynamic Recurrent Neural Networks, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1990.
[54] B. de Vries, J.C. Principe, A theory for neural networks with time delays, in: R.P. Lippmann, J.E. Moody, D.S. Touretzky (Eds.), Advances in Neural Information Processing Systems 3, Morgan Kaufmann, 1991, pp. 162–168.
[55] Leon O. Chua, Lin Yang, Cellular neural networks: Theory, IEEE Trans. Circuits Syst. 35 (1988) 1257–1272.
[56] Leon O. Chua, Lin Yang, Cellular neural networks: Applications, IEEE Trans. Circuits Syst. 35 (1988) 1273–1290.
[57] Lloyd N. Trefethen, Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations. unpublished text, Cambridge, MA, 1996.
[58] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, Prentice-Hall, 1989.
[59] Oriol Vinyals, Beyond Deep Learning: Scalable Methods and Models for Learning (Ph.D. thesis), EECS Department, University of California, Berkeley, 2013.
[60] Alex Sherstinsky, Rosalind W. Picard, On the efficiency of the orthogonal least squares training method for radial basis function networks, IEEE Trans. Neural Netw. 7 (1) (1996) 195–200.
[61] Amar Gopal Bose, A Theory of Nonlinear Systems (Ph.D. thesis), Massachusetts Institute of Technology, 1956.
[62] Sepp Hochreiter, Bengio Yoshua, Frasconi Paolo, Jürgen Schmidhuber, Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, in: Kremer, Kolen (Eds.), A Field Guide to Dynamical Recurrent Neural Networks, IEEE Press, 2001.
[63] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training recurrent neural networks. in: International Conference on Machine Learning, 2013, pp. 1310–1318.
[64] P.J. Werbos, Generalization of backpropagation with application to a recurrent gas market model, Neural Netw. 1 (1988).
[65] P. Werbos, Backpropagation through time: what does it do and how to do it, in: Proceedings of IEEE, vol. 78, 1990, pp. 1550–1560.
[66] Ilya Sutskever, Training Recurrent Neural Networks (Ph.D. thesis), University of Toronto, 2012.
[67] Razvan Pascanu, On Recurrent and Deep Neural Networks (Ph.D. thesis), Université de Montréal, 2014.
[68] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning Internal Representations By Error Propagation, Technical Report DTIC Document, University of California, San Diego, 1985.
[69] D.E. Rumelhart, J.L. Mcclelland (Eds.), Parallel Distributed Processing, volume 1, MIT Press, 1986.
[70] M.L. Minsky, S.A. Papert, Perceptrons, second ed., MIT Press, Cambridge MA, 1990.
[71] Ronald J. Williams, David Zipser, A learning algorithm for continually running fully recurrent neural networks, Neural Comput. 1 (1989) 270–280.
[72] Mike Schuster, Kuldip K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (1997) 2673–2681.
[73] Omer Levy, Kenton Lee, Nicholas FitzGerald, Luke Zettlemoyer, Long short-term memory as a dynamically computed element-wise weighted sum, CoRR abs/1805.03716 (2018).
[74] Felix Gers, Long Short-Term Memory in Recurrent Neural Networks (Ph.D. thesis), École Polytechnique Fédérale de Lausanne, 2001.
[75] L.R. Rabiner, Techniques for designing finite-duration impulse-response digital filters, IEEE Trans. Commun. Technol. 19 (2) (1971) 188–195.
[76] J.H. McClellan, T.W. Parks, L.R. Rabiner, A computer program for designing optimum FIR linear phase digital filters, IEEE Trans. Audio Electroacoust. 21 (6) (1973) 506–526.
[77] Y. Yamamoto, B.D.O. Anderson, M. Nagahara, Y. Koyanagi, Optimizing FIR approximation for discrete-time IIR filters, IEEE Signal Process. Lett. 10 (9) (2003) 273–276.
[78] Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals, Recurrent neural network regularization, 2014, arxiv:1409.2329.