Gated Recurrent Neural Networks with Weighted Time-Delay Feedback

N. Benjamin Erichson
ICSI and LBNL

Soon Hoe Lim
KTH and Nordita

Michael Mahoney
UC Berkeley, ICSI and LBNL

Abstract

In this paper, we present a novel approach to modeling long-term dependencies in sequential data by introducing a gated recurrent unit (GRU) with a weighted time-delay feedback mechanism. Our proposed model, named τ-GRU, is a discretized version of a continuous-time formulation of a recurrent unit, where the dynamics are governed by delay differential equations (DDEs). We prove the existence and uniqueness of solutions for the continuous-time model and show that the proposed feedback mechanism can significantly improve the modeling of long-term dependencies. Our empirical results indicate that τ-GRU outperforms state-of-the-art recurrent units and gated recurrent architectures on a range of tasks, achieving faster convergence and better generalization.

1 INTRODUCTION

Recurrent neural networks (RNNs) and their variants are flexible gradient-based methods specially designed to model sequential data. Models of this type can be viewed as dynamical systems whose temporal evolution is governed by a system of differential equations driven by an external input. Indeed, there is a long-standing tradition to formulate continuous-time variants of RNNs (Pineda, 1988). In this setting, the data are formulated in continuous-time, i.e., inputs are defined by the function $\mathbf{x} = \mathbf{x}(t) \in \mathbb{R}^p$ and targets are defined as $\mathbf{y} = \mathbf{y}(t) \in \mathbb{R}^q$, where t denotes continuous time. In this way, one can, for instance, employ a nonautonomous ordinary differential equation (ODE) to model the dynamics of the hidden states $\mathbf{h}(t) \in \mathbb{R}^d$:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), \mathbf{x}(t); \boldsymbol{\theta}).$$

Here, $f : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ is a function which is parameterized by a neural network (NN) with the learnable weights $\boldsymbol{\theta}$. A prototypical choice for f is the tanh recurrent unit:

$$f(\mathbf{h}(t), \mathbf{x}(t); \boldsymbol{\theta}) := \tanh(\mathbf{W}\mathbf{h}(t) + \mathbf{U}\mathbf{x}(t) + \mathbf{b}),$$

where $\mathbf{W} \in \mathbb{R}^{d\times d}$ denotes a hidden-to-hidden weight matrix, $\mathbf{U} \in \mathbb{R}^{d\times p}$ an input-to-hidden weight matrix, and $\mathbf{b}$ a bias term. With this continuous-time formulation in hand, one can then use tools from dynamical systems theory to study the dynamical behavior of the model as well as to motivate mechanisms that can prevent rapidly diverging or converging dynamics. For instance, Chang et al. (2018) proposed a parametrization of the hidden matrix as an antisymmetric matrix to ensure stable hidden state dynamics, and Erichson et al. (2020) relaxed this idea to improve model expressivity. More recently, Rusch et al. (2022) has proposed an RNN architecture based on a suitable time-discretization of a set of coupled multiscale ODEs.

In this work, we consider using input-driven nonlinear delay differential equations (DDEs) to model the dynamics of hidden states:

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), \mathbf{h}(t-\tau), \mathbf{x}(t); \boldsymbol{\theta}),$$

where τ is a constant that indicates the delay (i.e., time-lag). Here, the time derivative is described by a function $f : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ that explicitly depends on states from the past. Prior work (Lin et al., 1996) has shown that delay units can improve performance on long-term dependency problems (Pascanu et al., 2013), i.e., problems for which the desired model output depends on inputs presented at times far in the past.

In more detail, we propose a novel continuous-time nonlinear recurrent unit, given in Eq. (1), that is composed of two parts: (i) a component $u(\mathbf{h}(t), \mathbf{x}(t))$ that

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

τ-GRU

Continuous-time formulation of τ-GRU:

$$\frac{d\,\mathbf{h}(t)}{dt} = \underbrace{g(\mathbf{h}(t), \mathbf{x}(t))}_{\text{gating}} \left(\underbrace{u(\mathbf{h}(t),\, \mathbf{x}(t))}_{\text{instantaneous dynamics}} + \underbrace{a(\mathbf{h}(t), \mathbf{x}(t)) \odot z(\mathbf{h}(t-\tau),\, \mathbf{x}(t))}_{\text{weighted time-delayed feedback}} - \mathbf{h}(t) \right) \tag{1}$$

Discrete-time formulation of τ-GRU:

$$\mathbf{h}_{n+1} = (1 - \mathbf{g}_n) \odot \mathbf{h}_n + \mathbf{g}_n \odot (\mathbf{u}_n + \mathbf{a}_n \odot \mathbf{z}_n) \tag{2}$$

with

$$\mathbf{u}_n = u(\mathbf{h}_n, \mathbf{x}_n) := \tanh(\mathbf{W}_1 \mathbf{h}_n + \mathbf{U}_1 \mathbf{x}_n) \tag{3}$$

$$\mathbf{z}_n = z(\mathbf{h}_l, \mathbf{x}_n) := \tanh(\mathbf{W}_2 \mathbf{h}_l + \mathbf{U}_2 \mathbf{x}_n) \tag{4}$$

$$\mathbf{g}_n = g(\mathbf{h}_n, \mathbf{x}_n) := \text{sigmoid}(\mathbf{W}_3 \mathbf{h}_n + \mathbf{U}_3 \mathbf{x}_n) \tag{5}$$

$$\mathbf{a}_n = a(\mathbf{h}_n, \mathbf{x}_n) := \text{sigmoid}(\mathbf{W}_4 \mathbf{h}_n + \mathbf{U}_4 \mathbf{x}_n) \tag{6}$$

input	$\mathbf{x}$	$\mathbb{R}^p$
time index	t	$\mathbb{R}$
time delay	τ	$\mathbb{R}$
hidden state	$\mathbf{h}$	$\mathbb{R}^d$
hidden-to-hidden matrix	$\mathbf{W}_i$	$\mathbb{R}^{d\times d}$
input-to-hidden matrix	$\mathbf{U}_i$	$\mathbb{R}^{d\times p}$
decoder matrix	$\mathbf{V}$	$\mathbb{R}^{q\times d}$

$\mathbf{h}_n \approx \mathbf{h}(t_n)$, $t_n = n\Delta t$, $n = 0, 1, \ldots$
$l := n - \lfloor \tau / \Delta t \rfloor$

explicitly models instantaneous dynamics; and (ii) a component $z(\mathbf{h}(t-\tau), \mathbf{x}(t))$ that provides time-delayed feedback to account for non-instantaneous dynamics. The feedback also helps to propagate gradient information more efficiently, thus lessening the issue of vanishing gradients. In addition, we introduce $a(\mathbf{h}(t), \mathbf{x}(t))$ to weight the importance of the feedback component-wise, which helps to better model different time scales. By considering a suitable time-discretization scheme of this continuous-time setup, we obtain a gated recurrent unit (GRU), given in Eq. (2), which we call τ-GRU. The individual parts are described by Eq. (3)-(6), where $\mathbf{g}_n$ and $\mathbf{a}_n$ resemble commonly used gating functions.

While nonlinear RNNs have been widely used to model sequences for decades, there are also other more recently developed models. However, there are trade-offs that different sequence models strike. Our proposed RNN model works better than Transformers (Vaswani et al., 2017; Tay et al., 2022) and state space models (SSMs) (Gu et al., 2021b, 2020, 2021a) in the small data regime while being parameter efficient (see also App. H for more details) – *we emphasize that this is the regime that we focus on in the present work.* The computational cost of the model scales linearly with the sequence length, and it is typically faster than Transformers at inference time even for modest sequence lengths. However, it is still slow to optimize due to the inherently sequential nature of the computation, and are therefore hard to scale. Transformers and SSMs are easier to scale and excel in the big data regime, but typically need a much larger number of trainable parameters and tend to overfit in the small data regime. While Transformers do not have to face the sequential training issue of recurrent models (and can be trained in parallel), the computational and memory cost of Transformers scales quadratically with the sequence length, and can therefore be very expensive to deploy on long sequences.

Main Contributions. Our key contributions are:

- **Design:** We introduce a novel gated recurrent unit, which we call τ-GRU, that incorporates a weighted time-delay feedback mechanism to lessen the vanishing gradients issue. This model is motivated by nonlinear DDEs, and it is obtained by discretizing the continuous Eq. (1).
- **Theory:** We show that the continuous-time τ-GRU model has a unique well-defined solution (see Theorem 1). Moreover, we provide intuition and analysis to understand how the introduction of delays in τ-GRU can act as a buffer to help lessen the vanishing gradients problem, thus improving the ability to retain information far in the past. See Proposition 1 for a simplified setting and Proposition 3 in App. D for τ-GRU.
- **Experiments:** We provide empirical results to demonstrate the performance of τ-GRU on a variety of benchmark tasks. We show that τ-GRU converges faster during training and can achieve improved generalization performance. Moreover, we demonstrate that it provides favorable trade-offs between effectiveness in dealing with long-term dependencies and expressivity in the considered tasks. See Figure 1 for an illustration of this.

Our main focus here is on *shallow (single-layer) nonlinear RNNs*, in particular we provide a theoretically backed approach to improve upon existing RNN models, not so much about achieving state-of-the-art results among all the possible sequence models out there.

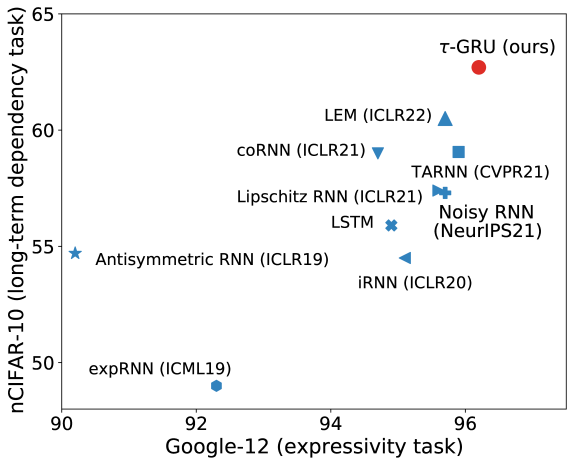


Figure 1: Test accuracy for nCIFAR (Chang et al., 2018) versus Google-12 (Warden, 2018). nCIFAR requires a recurrent unit with long-term dependency capabilities, while Google-12 requires a highly expressive unit. Our τ-GRU is able to improve performance on both tasks, relative to existing state-of-the-art alternatives, including LEM (Rusch et al., 2022).

2 RELATED WORK

In this section, we discuss recent RNN advances that have been shown to outperform classic architectures such as Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) and the GRUs (Cho et al., 2014). We also briefly discuss prior works on incorporating delays into NNs.

Unitary and orthogonal RNNs. The seminal work (Arjovsky et al., 2016) introduced a recurrent unit where the hidden matrix is constructed as the product of unitary matrices. This enforces that the eigenvalues lie on the unit circle. This, in turn, prevents vanishing and exploding gradients, thereby enabling the learning of long-term dependencies. However, such unitary RNNs suffer from limited expressivity, since the construction of the hidden matrix is restrictive (Azencot et al., 2021). Work by Wisdom et al. (2016) and Vorontsov et al. (2017) partially addressed this issue by considering the Cayley transform on skew-symmetric matrices; and work by L-Casado and M-Rubio (2019); Lezcano Casado (2019) leveraged skew-Hermitian matrices to parameterize the orthogonal group to improve expressiveness. The expressiveness of RNNs has been further improved by considering nonnormal hidden matrices (Kerg et al., 2019).

Continuous-time nonlinear RNNs. The work on Neural ODEs (Chen et al., 2018) and variants (Kidger et al., 2020; Queiruga et al., 2021; Xia et al., 2021; Hasani et al., 2022a) have motivated the formulation of several modern continuous-time RNNs, which are expressive and have good long-term memory. The work by Chang et al. (2018) used an antisymmetric matrix to parameterize the hidden-to-hidden matrix in order to obtain stable dynamics. In (Kag et al., 2020), a modified differential equation was considered, which allows one to update the hidden states based on the difference between predicted and previous states.

Rusch and Mishra (2021) demonstrated that long-term memory can be improved by modeling the hidden dynamics by a second-order system of ODEs, which models a coupled network of controlled forced and damped nonlinear oscillators. Another approach for improving long-term memory was motivated by a time-adaptive discretization of an ODE (Kag and Saligrama, 2021). The expressiveness of continuous-time RNNs has been further improved by introducing a suitable time-discretization of a set of multiscale ODEs (Rusch et al., 2022). Lastly, Lim et al. (2021) studied noise-injected RNNs which can be viewed as discretizations of stochastic differential equations driven by input data. In this case, the noise can help to stabilize the hidden dynamics during training and improve robustness.

Using delays in NNs. The idea of introducing delays into NNs goes back to (Waibel et al., 1989; Lang et al., 1990). Several works followed: Kim (1998) considered a time-delayed RNN model that is suitable for temporal correlations and prediction of chaotic and financial time series; and delays were also incorporated into the nonlinear autoregressive with exogenenous inputs (NARX) RNNs (Lin et al., 1996). More recently, Zhu et al. (2021) introduced delay terms in Neural ODEs and demonstrated their approximation capacities. In particular, this model is able to learn delayed dynamics where the trajectories in the lower-dimensional phase space could be mutually intersected, while the standard Neural ODEs (Chen et al., 2018) are not able to do so.

State-space models. Recently, state-space models (SSMs) (Gu et al., 2021a,b; Yu et al., 2024a) have emerged as alternatives to RNNs due to their strong performance on long-sequence tasks. SSMs have demonstrated state-of-the-art results across diverse applications in video, audio, and time-series processing, often surpassing traditional LSTM and Transformer architectures while offering significant speed and memory efficiency (Gu and Dao, 2023).

In contrast to RNNs, which rely on sequential processing and suffer from slow training, SSMs exploit linear time-invariant (LTI) dynamics, enabling parallel computation either in the time domain (Smith et al., 2022) or frequency domain (Yu et al., 2023, 2024b). Such parallelization significantly enhances computational efficiency and scalability for long sequences.

3 METHOD

In this section, we provide an introduction to DDEs; then, we motivate the formulation of our DDE-based models, in continuous and discrete time; and, finally, we propose a weighted time-delay feedback architecture.

Notation. $\odot$ denotes Hadamard product, $|v|$ denotes vector norm for the vector v, $\|A\|$ denotes operator norm for the matrix A, σ and $\hat{\sigma}$ (or sigmoid) denote the tanh and sigmoid function, respectively; and $\lceil x \rceil$ and $\lfloor x \rfloor$ denote the ceiling and floor function in x.

3.1 Delay Differential Equations

DDEs are an important class of dynamical systems that arise in natural and engineering sciences (Smith, 2011; Erneux, 2009; Keane et al., 2017). In these systems, a feedback term is introduced to adjust the system non-instantaneously, resulting in delays in time. In mathematical terms, the derivative of the system state depends explicitly on the past value of the state variable. Here, we focus on DDEs with a single discrete delay

$$\dot{h} = F(t, h(t), h(t-\tau)), \tag{7}$$

with $\tau > 0$, where F is a continuous function. Due to the presence of the delay term, we need to *specify an initial function* which describes the behavior of the system prior to the initial time $t = 0$. For the DDE, it would be a function ϕ defined on $[-\tau, 0]$. Hence, a DDE numerical solver must save all the information needed to approximate delayed terms.

Instead of thinking the solution of the DDE as consisting of a sequence of values of h at increasing values of t, as one would do for ODEs, it is more fruitful to view it as a mapping of functions on the interval $[t-\tau, t]$ into functions on the interval $[t, t+\tau]$, i.e., as a sequence of functions defined over a set of contiguous time intervals of length τ. Since the state of the system at time $t \geq 0$ must contain all the information necessary to determine the solution for future times $s \geq t$, it should contain the initial condition ϕ.

More precisely, the DDE is a functional differential equation with the state space $C := C([-\tau, 0], \mathbb{R}^d)$. This state space is the Banach space of continuous functions from $[-\tau, 0]$ into $\mathbb{R}^d$, with the topology of uniform convergence. It is equipped with the norm $\|\phi\| := \sup\{|\phi(\theta)| : \theta \in [-\tau, 0]\}$. In contrast to the ODEs (with $\tau = 0$) whose state space is finite-dimensional, DDEs are generally infinite-dimensional dynamical systems. Various aspects of DDEs have been studied, including their solution properties (Hale and Lunel, 2013; Asl and Ulsoy, 2003), dynamics (Lepri et al., 1994; Baldi and Atiya, 1994) and stability (Marcus and Westervelt, 1989; Bélair, 1993; Liao et al., 2002; Yang et al., 2014; Park et al., 2019).

3.2 Continuous-Time τ-GRUs

The basic form of a time-delayed RNN is

$$\dot{\mathbf{h}} = \sigma(\mathbf{W}_1\mathbf{h}(t) + \mathbf{W}_2\mathbf{h}(t-\tau) + \mathbf{U}\mathbf{x}(t)) - \mathbf{h}(t), \tag{8}$$

for $t \geq 0$, and $\mathbf{h}(t) = 0$ for $t \in [-\tau, 0]$, with the output $\mathbf{y}(t) = \mathbf{V}\mathbf{h}(t)$. In this expression, $\mathbf{h} \in \mathbb{R}^d$ denotes the hidden states, $f : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^d$ is a nonlinear function, and $\sigma : \mathbb{R} \to (-1, 1)$ denotes the tanh activation function applied component-wise. The matrices $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d\times d}$, $\mathbf{U} \in \mathbb{R}^{d\times p}$ and $\mathbf{V} \in \mathbb{R}^{q\times d}$ are learnable parameters, and $\tau \geq 0$ denotes the discrete time-lag. For notational brevity, we omit the bias term here, assuming it is included in $\mathbf{W}_1$.

It is important to equip RNNs with mechanism to better represent a large number of scales, as discussed by Tallec and Ollivier (2018) and more recently by Rusch et al. (2022). Therefore, we follow (Tallec and Ollivier, 2018) and consider a time warping function $c : \mathbb{R}^d \to \mathbb{R}^d$ which we define to be a parametric function $c(t)$. Using the reasoning in (Tallec and Ollivier, 2018), and denoting $t_\tau := t - \tau$, we can formulate the following continuous-time delay recurrent unit

$$\dot{\mathbf{h}} = \frac{dc(t)}{dt}\left[\sigma(\mathbf{W}_1\mathbf{h}(t) + \mathbf{W}_2\mathbf{h}(t_\tau) + \mathbf{U}_1\mathbf{x}(t)) - \mathbf{h}(t)\right].$$

Now, we need a learnable function to model $\frac{dc(t)}{dt}$. A natural choice is to consider a standard gating function, which is a universal approximator, taking the form

$$\frac{dc(t)}{dt} = \hat{\sigma}(\mathbf{W}_3\mathbf{h}_t + \mathbf{U}_3\mathbf{x}_t) =: \mathbf{g}(t), \tag{9}$$

where $\mathbf{W}_3 \in \mathbb{R}^{d\times d}$ and $\mathbf{U}_3 \in \mathbb{R}^{d\times p}$ are learnable parameters, and $\hat{\sigma} : \mathbb{R} \to (0, 1)$ is the component-wise sigmoid function.

3.3 Discrete-Time τ-GRUs

To learn the weights of the recurrent unit, a numerical integration scheme can be used to discretize the continuous model. Specifically, we discretize the time as $t_n = n\Delta t$ for $n = -\lfloor \tau/\Delta t \rfloor, \ldots, -1, 0, 1, \ldots$, and approximate the solution $(\mathbf{h}(t))$ to Eq. (8) by the sequence $(\mathbf{h}_n = \mathbf{h}(t_n))$, given by $\mathbf{h}_n = 0$ for $n = -\lfloor \tau/\Delta t \rfloor, \ldots, 0$, and

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \int_{t_n}^{t_n+\Delta t} f(\mathbf{h}(s), \mathbf{h}(s-\tau), \mathbf{x}(s))\, ds \tag{10}$$

$$\approx \mathbf{h}_n + \texttt{scheme}[f, \mathbf{h}_n, \mathbf{h}_l, \Delta t], \tag{11}$$

for $n = 0, 1, \ldots$, where the subscript n denotes discrete time indices, $l := n - \lfloor \tau/\Delta t \rfloor$, and Δt represents the

time difference between a pair of consecutive elements in the input sequence. In addition, `scheme` refers to a numerical integration scheme whose application yields an approximate solution for the integral. Using the forward Euler scheme and choosing $\Delta t = 1$ gives:

$$\mathbf{h}_{n+1} = (1-\mathbf{g}_n)\odot\mathbf{h}_n + \mathbf{g}_n \odot \sigma(\mathbf{W}_1\mathbf{h}_n + \mathbf{W}_2\mathbf{h}_l + \mathbf{U}\mathbf{x}_n). \tag{12}$$

Note that this discretization corresponds to the leaky-integrator described by Jaeger et al. (2007) and we can in principle use other values of Δt. We choose forward Euler because it is computationally efficient. Higher-order integrator schemes require more function evaluations, while tending not to help to improve the performance (Queiruga et al., 2020).

It can be shown that (12) is a universal approximator of a large class of open dynamical systems with delay (see Theorem 4). However, the performance of this architecture is not able to outperform existing RNN architectures on a number of tasks. To improve the performance, we propose a modified model next.

3.4 Discrete-Time τ-GRUs with a Weighted Time-Delay Feedback Architecture

In this work, we propose to model the hidden dynamics using a mixture of a standard recurrent unit and a delay recurrent unit. To this end, we replace the σ in Eq. (12) by

$$\mathbf{u}_n + \mathbf{a}_n \odot \mathbf{z}_n, \tag{13}$$

so that we yield a new GRU that takes the form

$$\mathbf{h}_{n+1} = (1-\mathbf{g}_n)\odot\mathbf{h}_n + \mathbf{g}_n \odot (\mathbf{u}_n + \mathbf{a}_n \odot \mathbf{z}_n). \tag{14}$$

Here $\mathbf{u}_n$ describes the standard recurrent unit $\mathbf{u}_n = \tanh(\mathbf{W}_1\mathbf{h}_n + \mathbf{U}_1\mathbf{x}_n)$, and $\mathbf{z}_n$ describes the delay recurrent unit $\mathbf{z}_n = \tanh(\mathbf{W}_2\mathbf{h}_l + \mathbf{U}_2\mathbf{x}_n)$. Further, the gate $\mathbf{g}_n$ is a learnable vector-valued coefficient $\mathbf{g}_n = \text{sigmoid}(\mathbf{W}_3\mathbf{h}_n + \mathbf{U}_3\mathbf{x}_n)$. The weighting term $\mathbf{a}_n$ is also a vector-valued coefficient $\mathbf{a}_n = \text{sigmoid}(\mathbf{W}_4\mathbf{h}_n + \mathbf{U}_4\mathbf{x}_n)$, with the learnable parameters $\mathbf{W}_4 \in \mathbb{R}^{d\times d}$ and $\mathbf{U}_4 \in \mathbb{R}^{d\times p}$, that weights the importance of the time-delay feedback component-wise for the task on hand.

From the design point of view, Eq. (13) can be motivated by the sigmoidal coupling used in Hodgkin-Huxley type neural models (see Eq. (1)-(2) and Eq. (4) in (Campbell, 2007)). Importantly, Eq. (13) provides a flexible mechanism to allow the network to model a mixture of instantaneous term and a delay term for the nonlinearity, thereby increasing the expressivity of the model. Note that Eq. (12) also works well but could not outperform other RNN models due to fewer trainable parameters. Our empirical evaluations on a wide range of experiments confirm the advantages of having this mechanism.

4 THEORY

In this section, we define the notion of solution for DDEs and show that the continuous-time τ-GRU has a unique solution. Moreover, we provide intuition and analysis to understand how the delay mechanism can help improving long-term dependencies.

4.1 Existence and Uniqueness of Solution for Continuous-Time τ-GRUs

Since we must know $h(t+\theta)$, $\theta \in [-\tau, 0]$ in order to determine the solution of the DDE (7) for $s > t$, we call the state of the dynamical system at time t the element of C which we denote as h_t, defined as $h_t(\theta) := h(t+\theta)$ for $\theta \in [-\tau, 0]$. The trajectory of the solution can thus be viewed as the curve $t \to h_t$ in the state space C. In general, DDEs can be formulated as the following initial value problem (IVP) for the nonautonomous system (Hale and Lunel, 2013):

$$\dot{h}(t) = f(t, h_t), \ t \geq t_0, \tag{15}$$

where $h_{t_0} = \phi \in C$ for some initial time $t_0 \in \mathbb{R}$, and $f : \mathbb{R} \times C \to \mathbb{R}^d$ is a given continuous function. The above equation describes a general type of systems, including ODEs ($\tau = 0$) and DDEs of the form $\dot{h}(t) = g(t, h(t), h(t-\tau))$ for some continuous function g.

We say that a function h is a solution of Eq. (15) on $[t_0 - \tau, t_0 + A]$ if there exists $t_0 \in \mathbb{R}$ and $A > 0$ such that $h \in C([t_0 - \tau, t_0 + A), \mathbb{R}^d)$, $(t, h_t) \in \mathbb{R} \times C$ and $h(t)$ satisfies Eq. (15) for $t \in [t_0, t_0 + A)$. It can be shown that (see, for instance, Lemma 1.1 in (Hale and Lunel, 2013)) if $f(t, \phi)$ is continuous, then finding a solution of Eq. (15) is equivalent to solving the integral equation: $h_{t_0} = \phi$,

$$h(t) = \phi(0) + \int_{t_0}^{t} f(s, h_s)\ \text{ds}, \ \text{t} \geq \text{t}_0. \tag{16}$$

We now provide existence and uniqueness result for the continuous-time τ-GRU model, assuming that the input x is continuous in t. Defining the state $h_t \in C$ as $h_t(\theta) := h(t+\theta)$ for $\theta \in [-\tau, 0]$ as before, the DDE describing the τ-GRU model can be formulated as the following IVP:

$$\dot{h} = -h(t) + u(t, h(t)) + a(t, h(t)) \odot z(t, h_t), \ t \geq t_0, \tag{17}$$

and $h_{t_0} = \phi \in C$ for some initial time $t_0 \in \mathbb{R}$, with the dependence on $x(t)$ viewed as dependence on t. By applying Theorem 3.7 in (Smith, 2011), we obtain the following result. See App. B for a proof of this theorem.

Theorem 1 (Existence and uniqueness of solution for continuous-time τ-GRU)**.** *Let $t_0 \in \mathbb{R}$ and $\phi \in C$ be given. There exists a unique solution $h(t) = h(t, \phi)$ of*

Eq. (1), defined on $[t_0 - \tau, t_0 + A]$ for any $A > 0$. In particular, the solution exists for all $t \geq t_0$, and

$$\|h_t(\phi) - h_t(\psi)\| \leq \|\phi - \psi\| e^{K(t-t_0)},$$

for all $t \geq t_0$, where $K = 1 + \|W_1\| + \|W_2\| + \|W_4\|/4$.

Theorem 1 guarantees that the continuous-time τ-GRU, as a functional differential equation, has a well-defined unique solution that does not blow up in finite time.

4.2 The Delay Mechanism in τ-GRUs Can Help Improve Long-Term Dependencies

RNNs suffer from the problem of vanishing and exploding gradients, leading to the problem of long-term dependencies. While the gating mechanisms could mitigate the problem to some extent, the delays introduced in τ-GRUs can further help reduce the sensitivity to long-term dependencies.

To understand the reason for this, we consider how gradients are computed using the backpropagation through time (BPTT) algorithm (Pascanu et al., 2013). BPTT involves the two stages of unfolding the network in time and backpropagating the training error through the unfolded network. When τ-GRUs are unfolded in time, the delays in the hidden state will appear as jump-ahead connections (buffers) in the unfolded network. These buffers provide a shorter path for propagating gradient information, and therefore reducing the sensitivity of the network to long-term dependencies. Such intuition is also used to explain the behavior of the NARX RNNs in (Lin et al., 1996).

We use a simplified setting to make this intuition precise. See App. C for a proof of this proposition. We also provide results (bounds for the gradient norm) and discussions for τ-GRU (in App. D, see Proposition 3).

Proposition 1. *Consider the linear time-delayed RNN, with hidden states described by the update equation:*

$$h_{n+1} = Ah_n + Bh_{n-m} + Cu_n, \ n = 0, 1, \ldots, \quad (18)$$

and $h_n = 0$ for $n = -m, -m+1, \ldots, 0$ with $m > 0$. Then, assuming that A and B commute, we have:

$$\frac{\partial h_{n+1}}{\partial u_i} = A^{n-i}C, \quad (19)$$

for $n = 0, 1, \ldots, m$, $i = 0, \ldots, n$, and

$$\frac{\partial h_{m+1+j}}{\partial u_i} = A^{m+j-i}C + \delta_{i,j-1}BC + 2\delta_{i,j-2}ABC + 3\delta_{i,j-3}A^2BC + \cdots + j\delta_{i,0}A^{j-1}BC,$$

for $j = 1, 2, \ldots, m+1$, $i = 0, 1, \ldots, m+j$, where $\delta_{i,j}$ denotes the Kronecker delta.

We remark that the commutativity assumption is not necessary. It is used here only to simplify the expression for the gradients. Analogous formula for the gradients can be derived without such assumption, at the cost of displaying more complicated formulae.

From Proposition 1, we see that the presence of the delay allows the model to place more emphasis on the gradients due to input information in the past (as can also be seen in Eq. (37) in the proof of the proposition, where additional terms dependent on B appear in the coefficients in front of the past inputs). In particular, if $\|A\| < 1$ and $B = 0$, then the gradients decay exponentially as i becomes large. Introducing the delay term ($B \neq 0$) dampens the exponential decay by perturbing the gradients of hidden states (dependent on the delay parameter m) with respect to the past inputs with nonzero values, thereby lessening the vanishing gradient issue.

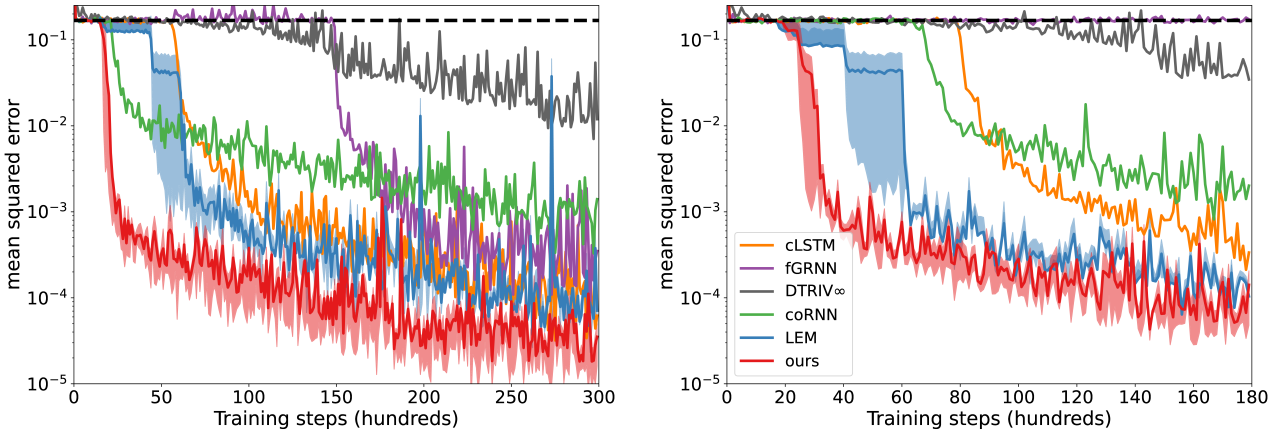
Similar qualitative conclusion can also be drawn for τ-GRU (see Proposition 3 and the discussions in App. D). Therefore, one expects that these networks would be able to more effectively deal with long-term dependencies than the counterpart models without delays.

5 EXPERIMENTAL RESULTS

In this section, we consider several tasks to demonstrate the performance of τ-GRU when compared to existing RNN models (see extra experiments in App. F). We use standard protocols for training, and validation sets for parameter tuning (see App. G for details and a study of sensitivity to random initialization).

The Adding Task. The adding task, proposed by Hochreiter and Schmidhuber (1997), tests a model's ability to learn long-term dependencies. The inputs are two stacked random vectors u and v of length N. Elements of u are drawn from $\mathcal{U}(0, 1)$, while v has two non-zero elements (both set to 1) at random locations $i \in 1, \ldots, \lfloor \frac{N}{2} \rfloor$ and $j \in \lceil \frac{N}{2} \rceil, \ldots, N$. The target value is $\sum(u \odot v)$, i.e., the sum of the two elements in u corresponding to the non-zero entries in v.

Following Rusch et al. (2022), we consider two challenging settings with very long input sequences $N = 2000, 5000$. Figure 2 shows results for our τ-GRU compared to several state-of-the-art RNN models designed for long-term dependency tasks, such as LEM (Rusch et al., 2022), coRNN (Rusch and Mishra, 2021), DTRIV∞(Lezcano Casado, 2019), fastGRNN(Kusupati et al., 2018), and LSTM with chrono initialization (Tallec and Ollivier, 2018). DTRIV∞ and fastGRNN perform poorly in both cases, while our τ-GRU converges faster and performs better.



(a) Sequence length $N = 2000$.

(b) Sequence length $N = 5000$.

Figure 2: Results for the adding task. We show the one standard deviation bands for LEM and our τ-GRU. On average, τ-GRU converges faster, and obtains a lower MSE on the adding task.

Human Activity Recognition: HAR-2. Next, we evaluate our model's performance on human activity recognition using the HAR dataset (Anguita et al., 2012), which includes accelerometer and gyroscope measurements from a Samsung Galaxy S3 smartphone tracking six activities performed by 30 volunteers aged 19-48. The sequences are divided into shorter segments of length $N = 128$, with the raw measurements summarized by 9 features per time step. The HAR-2 dataset (Kusupati et al., 2018) groups the activities into two categories. We use $7,352$ sequences for training, 900 for validation, and $2,947$ for testing.

Our τ-GRU outperforms traditional gated architectures on this task, as shown in Table 1. The most competitive model is coRNN (Rusch and Mishra, 2021), achieving 97.2% test accuracy with just 9k parameters. LEM (Rusch et al., 2022) achieves 97.1% with the same number of parameters as our τ-GRU.

Table 1: Results for HAR2 task.

Model	Test Acc. (%)	# units	# param
GRU (Kusupati et al., 2018)	93.6	75	19k
LSTM (Kag et al., 2020)	93.7	64	19k
FastRNN (Kusupati et al., 2018)	94.5	80	7k
FastGRNN (Kusupati et al., 2018)	95.6	80	7k
AsymRNN (Kag et al., 2020)	93.2	120	8k
iRNN (Kag et al., 2020)	96.4	64	4k
DIRNN (Zhang et al., 2021)	96.5	64	-
coRNN (Rusch and Mishra, 2021)	97.2	64	9k
LipschitzRNN	95.4	64	9k
LEM	97.1	64	19k
τ-GRU (ours)	**97.4**	64	19k

Sentiment Analysis: IMDB. Here, we test the expressiveness of our proposed model on the sentiment analysis task using the IMDB dataset (Maas et al., 2011). This dataset consists of 50k movie reviews, each labeled with a positive or negative sentiment, with an average length of 240 words. The dataset is evenly split into a training set and a test set, with 15% of the training data used for validation. After standard preprocessing, we embed the data using a pretrained GloVe model (Pennington et al., 2014), restricting the dictionary to 25k words. Our τ-GRU achieves higher test accuracy than LSTM, GRU, continuous-time coRNN (Rusch and Mishra, 2021), and LEM (Rusch et al., 2022), as shown in Table 2.

Table 2: Results for the IMDB task.

Model	Test Acc. (%)	# units	# param
LSTM (Campos et al., 2018)	86.8	128	220k
Skip LSTM (Campos et al., 2018)	86.6	128	220k
GRU (Campos et al., 2018)	86.2	128	164k
Skip GRU (Campos et al., 2018)	86.6	128	164k
ReLU GRU (Dey and Salem, 2017)	84.8	128	99k
coRNN (Rusch and Mishra, 2021)	87.4	128	46k
LEM	88.1	128	220k
τ-GRU (ours)	**88.7**	128	220k

Sequential Image Classification. We evaluate the long-term dependency learning capabilities of RNNs on four image classification datasets: sequential MNIST (sMNIST), permuted sMNIST (psMNIST), sequential CIFAR-10 (sCIFAR), and noise-padded CIFAR-10 (nCIFAR). For the sMNIST and psMNIST tasks, $N = 784$ pixels of each image are sequentially presented to the RNN, with psMNIST using a fixed random permutation. The sCIFAR task has a sequence length of $N = 1024$, with each element being a 3D vector representing the pixels for each color channel. The nCIFAR task uses a sequence of $N = 1000$ with 968 noisy elements of dimension 96.

τ-GRU outperforms other RNNs on the psMNIST,

Table 3: Test accuracies on sMNIST, psMNIST, sCIFAR, and nCIFAR.

Model	sMNIST	psMNIST	# units	# parms	sCIFAR	nCIFAR	# units	# parms
LSTM (Kag and Saligrama, 2021)	97.8	92.6	128	68k	59.7	11.6	128	69k / 117k
r-LSTM (Trinh et al., 2018)	98.4	95.2	-	100K	72.2	-	-	101k / -
chrono-LSTM (Rusch et al., 2022)	98.9	94.6	128	68k	-	55.9	128	- / 116k
Antisym. RNN (Chang et al., 2018)	98.0	95.8	128	10k	62.2	54.7	256	37k / 37k
Lipschitz RNN (Erichson et al., 2020)	99.4	96.3	128	34k	64.2	59.0	256	134k / 158k
expRNN (L-Casado and M-Rubio, 2019)	98.4	96.2	360	68k	-	49.0	128	- / 47k
iRNN (Kag et al., 2020)	98.1	95.6	128	8k	-	54.5	128	- / 12k
TARNN (Kag and Saligrama, 2021)	98.9	97.1	128	68k	-	59.1	128	- / 100K
Dilated GRU (Chang et al., 2017)	99.2	94.6	-	130k	-	-	-	- / -
coRNN (Rusch and Mishra, 2021)	99.3	96.6	128	34k	-	59.0	128	- / 46k
LEM (Rusch et al., 2022)	**99.5**	96.6	128	68k	-	60.5	128	- / 117k
Delay GRU (Eq. (12))	98.7	94.1	128	51k	56.1	53.7	128	52k / 75k
τ-GRU (ours)	99.4	97.3	128	68k	**74.9**	**62.7**	128	69k / 117k

sCIFAR, and nCIFAR tasks, demonstrating a notable advantage in the CIFAR tasks through the proposed weighted time-delay feedback mechanism. As shown in Figure 3, our model converges faster than other continuous-time models, such as LEM, coRNN, and LipschitzRNN, requiring significantly fewer epochs to reach peak performance.

Table 4: Results for the ENSO task.

Model	MSE ($\times 10^{-2}$)	# units	# parameters
Vanilla RNN	0.45	16	0.3k
LSTM	0.92	16	1.2k
GRU	0.53	16	0.9k
Lipschitz RNN	10.6	16	0.6k
coRNN	4.00	16	0.6k
LEM	0.31	16	1.2k
ablation ($\alpha = 0$)	0.31	16	0.6k
ablation ($\beta = 0$)	0.38	16	0.9k
τ-GRU (ours)	**0.17**	16	1.2k

Learning Climate Dynamics. We consider the task of learning the dynamics of the DDE model for El Niño Southern Oscillation (ENSO) of (Falkena et al., 2019) (see Eq. (46) there). It models the sea surface temperature T in the eastern Pacific Ocean as:

$$\dot{T} = T - T^3 - cT(t-\delta)(1 - \gamma T^2(t-\delta)), \ t > \delta, \quad (20)$$

where $\gamma < 1$, $c > 0$, with $T(t)$ satisfying $\dot{T} = T - T^3 - cT(0)(1 - \gamma T(0)^2)$ with $T(0) \sim \text{Unif}(0, 1)$ for $t \in [0, \delta]$. For data generation, we follow Falkena et al. (2019), and choose $c = 0.93$, $\gamma = 0.49$ and $\delta = 4.8$. We use the classical Runge-Kutta method (RK4) to numerically integrate the system from $t = 0$ to $t = 400$ using a step-size of 0.1. The training and testing samples are the time series (of length 2000) generated by the RK4 scheme on the interval $[200, 400]$ for different realizations of $T(0)$.

Table 4 shows that our model ($\alpha = \beta = 1$, $\tau = 20$) is more effective in learning the ENSO dynamics when compared to other models. We also see that the predictive performance deteriorates without using appropriate combination of standard and delay recurrent units (setting either α or β to zero).

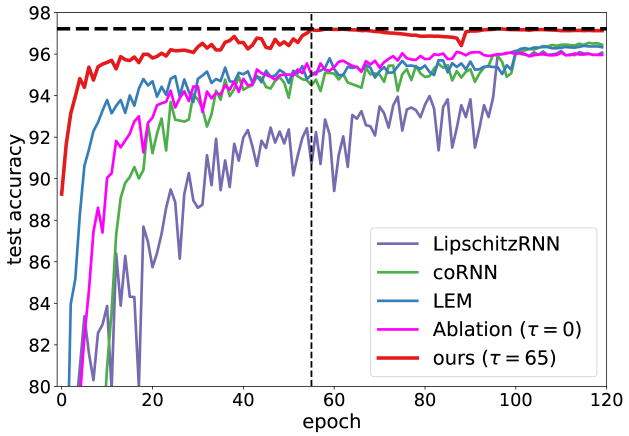


Figure 3: Test accuracy for psMNIST.

Frequency Classification. Next, we consider the frequency classification task introduced by Moreno-Pino et al. (2024). The dataset consists of N time series samples in 100 distinct frequency classes, one for each frequency f_j, linearly spaced between 1 and 2^{12}:

$$f_j = 1 + (j-1)\frac{2^{12} - 1}{99}, \quad j = 1, \dots, 100, \quad (21)$$

which are used to generate the length-L cosine signals $X_j(t_n) = \cos(2\pi f_j t_n) + \sigma \epsilon_j(t_n)$, where $t_n = n\Delta t$, $n = 0, 1, \dots, L-1$, $\sigma_j \geq 0$, and the $\epsilon_j(t_n)$ are i.i.d. standard Gaussians. We generated 1000 samples (i.e., $N = 10$) across the 100 distinct classes, with each of them sampled uniformly at 1000 time steps in the interval $[0, 1]$. We consider both the noise-free ($\sigma = 0$) and the more challenging noisy case ($\sigma = 0.1$).

Table 5 summarizes the classification accuracy obtained for different models. RNNs show strong performance

on the noise-free dataset but degrade substantially on the noisy dataset. The SSM S4 model (Gu et al., 2021a) struggles to clearly discriminate frequency classes, showing a limited capability of capturing nonlinear frequency dependencies within linear dynamics. This limitation arises since linear time-invariant systems in SSMs function effectively as linear filters and thus primarily scale Fourier modes rather than distinguishing between them through nonlinear recurrence. In contrast, our τ-GRU outperforms all other methods, achieving perfect accuracy (100%) in the noise-free scenario and near-perfect (99.1%) accuracy in the noisy scenario. Furthermore, τ-GRU converges quickly, attaining 100% test accuracy within only 3 epochs in the noise-free scenario, while baseline methods typically require between 11 and 45 epochs to converge. Similarly, in the noisy setting, our model reaches 99% accuracy after merely 15 epochs, substantially faster than other methods.

Table 5: Accuracy on the frequency classification task.

Model	No noise	With noise
Tanh-RNN	97.1%	35.6%
LSTM	100.0%	39.4%
LSTM (w/o forget gate)	99.0%	19.4%
LEM	96.0%	54.1%
SSM-S4D (1 layer)	67.5%	66.4%
SSM-S4D (4 layers)	68.9%	67.2%
GRU (no delay, ablation)	95.0%	57.7%
GRU (with delay, ours)	**100.0%**	**99.1%**

Ablation Study using psMNIST. We use the psMNIST task to perform an ablation study. To do so, we consider the following model

$$\mathbf{h}_{t+1} = (1 - \mathbf{g}_t) \odot \mathbf{h}_t + \mathbf{g}_t \odot (\beta \cdot \mathbf{u}_t + \alpha \cdot \mathbf{a}_t \odot \mathbf{z}_t),$$

where $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are constants that can be used to control the effect of different components. We are interested in the cases where α and β are either 0 or 1, i.e., a component is switched off or on. Table 6 shows the results for different ablation configurations. By setting $\alpha = 0$ we yield a simple gated RNN. Second, for $\beta = 0$, we yield a τ-GRU without instantaneous dynamics. Third, we show how different values of τ affect the performance. Setting $\tau = 0$ leads to a τ-GRU without time-delay feedback. We also show that a model without the weighting function $\mathbf{a}_t$ is not able to achieve peak performance.

Figure 4 demonstrates the effect of τ. The performance of τ-GRU is increasing as a function of τ and peaking around $\tau = 65$. It can be seen that the performance in the range 50-150 is relatively constant for this task. Thus, the model is relatively insensitive as long as τ is sufficiently large, but not too large. The performance is starting to decrease for $\tau > 150$. Since τ takes discrete values, tuning is easier as compared to continuous tuning parameters, for instance, parameters used by LEM (Rusch et al., 2022), coRNN (Rusch and Mishra, 2021), or LipschitzRNN (Erichson et al., 2020).

Table 6: Ablation study on psMNIST.

Model	α	β	τ	$\mathbf{a}_t$	Accuracy (%)
ablation	0	1	-	yes	94.6
ablation	1	0	65	yes	94.9
ablation	1	1	0	yes	95.1
ablation	1	1	20	yes	96.4
ablation	1	1	65	no	96.8
τ-GRU (ours)	1	1	65	yes	**97.3**

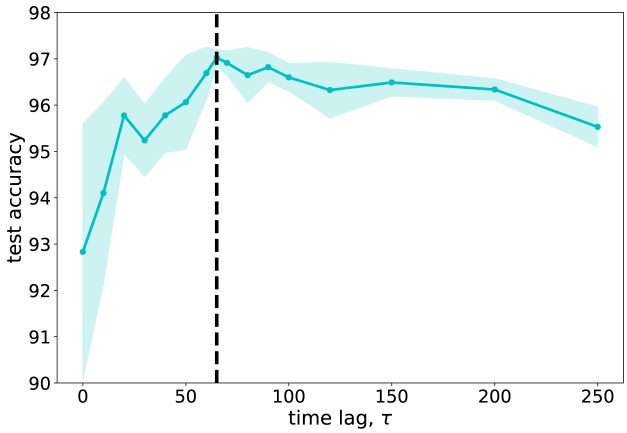


Figure 4: Sensitivity analysis of τ-GRU on psMNIST. The green envelope represent ± 1 s.d. around the mean.

6 CONCLUSION

Starting from a continuous-time formulation, we derive a discrete-time gated recurrent unit with delay, τ-GRU. We also provide intuition and theoretical analysis to understand how the proposed delay term can improve modeling of long-term dependencies. Importantly, we demonstrate the superior performance of τ-GRU in improving the long-range modeling capability of existing RNN models in a wide range of challenging tasks, from sequential image classification to learning nonlinear dynamics, given comparable number of trainable parameters. While there exist several other more sophisticated models such as the state-space models (SSMs) (Gu et al., 2021b,a), these models may not be optimal for tasks such as learning nonlinear dynamics in the *small data regime*, which is quite common in many scientific applications (see App. H).

One limitation of τ-GRU is that we consider only recurrent units with a single delay instead of multiple delays, limiting the potential of our architecture. We restrict to the single delay case to enable tractable analysis and experimentation. We shall leave the extension to include distributed delay mechanisms to future work.

Acknowledgments

NBE would like to acknowledge NSF, under Grant No. 2319621, and the U.S. Department of Energy, under Contract Number DE-AC02-05CH11231 and DE-AC02-05CH11231, for providing partial support of this work. SHL would like to acknowledge support from the Wallenberg Initiative on Networks and Quantum Information (WINQ), the Swedish Research Council (VR/2021-03648), and the resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725 (NAISS 2024/5-269).

References

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2012). Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International Workshop on Ambient Assisted Living*, pages 216–223. Springer.

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*, pages 1120–1128. PMLR.

Asl, F. M. and Ulsoy, A. G. (2003). Analysis of a system of linear delay differential equations. *J. Dyn. Sys., Meas., Control*, 125(2):215–223.

Azencot, O., Erichson, N. B., Ben-Chen, M., and Mahoney, M. W. (2021). A differential geometry perspective on orthogonal recurrent models. *arXiv preprint arXiv:2102.09589*.

Baldi, P. and Atiya, A. F. (1994). How delays affect neural dynamics and learning. *IEEE Transactions on Neural Networks*, 5(4):612–621.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945.

Bélair, J. (1993). Stability in a model of a delayed neural network. *Journal of Dynamics and Differential Equations*, 5(4):607–623.

Campbell, S. A. (2007). Time delays in neural systems. In *Handbook of Brain Connectivity*, pages 65–90. Springer.

Campos, V., Jou, B., Giró-i Nieto, X., Torres, J., and Chang, S.-F. (2018). Skip RNN: Learning to skip state updates in recurrent neural networks. In *International Conference on Learning Representations*.

Chang, B., Chen, M., Haber, E., and Chi, E. H. (2018). Antisymmetric RNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*.

Chang, S., Zhang, Y., Han, W., Yu, M., Guo, X., Tan, W., Cui, X., Witbrock, M., Hasegawa-Johnson, M. A., and Huang, T. S. (2017). Dilated recurrent neural networks. *Advances in neural information processing systems*, 30.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Dey, R. and Salem, F. M. (2017). Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1597–1600. IEEE.

Erichson, N. B., Azencot, O., Queiruga, A., Hodgkinson, L., and Mahoney, M. W. (2020). Lipschitz recurrent neural networks. In *International Conference on Learning Representations*.

Erneux, T. (2009). *Applied Delay Differential Equations*, volume 3. Springer.

Falkena, S. K., Quinn, C., Sieber, J., Frank, J., and Dijkstra, H. A. (2019). Derivation of delay equation climate models using the Mori-Zwanzig formalism. *Proceedings of the Royal Society A*, 475(2227):20190075.

Gu, A. and Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Ré, C. (2020). Hippo: Recurrent memory with optimal polynomial projections. *Advances in Neural Information Processing Systems*, 33:1474–1487.

Gu, A., Goel, K., and Ré, C. (2021a). Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. (2021b). Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in Neural Information Processing Systems*, 34:572–585.

Hale, J. K. and Lunel, S. M. V. (2013). *Introduction to Functional Differential Equations*, volume 99. Springer Science & Business Media.

Hasani, R., Lechner, M., Amini, A., Liebenwein, L., Ray, A., Tschaikowski, M., Teschl, G., and Rus, D. (2022a). Closed-form continuous-time neural networks. *Nature Machine Intelligence*, pages 1–12.

Hasani, R., Lechner, M., Wang, T.-H., Chahine, M., Amini, A., and Rus, D. (2022b). Liquid

structural state-space models. *arXiv preprint arXiv:2209.12951*.

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

Jaeger, H., Lukoševičius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, 20(3):335–352.

Kag, A. and Saligrama, V. (2021). Time adaptive recurrent neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15149–15158.

Kag, A., Zhang, Z., and Saligrama, V. (2020). RNNs incrementally evolving on an equilibrium manifold: A panacea for vanishing and exploding gradients? In *International Conference on Learning Representations*.

Keane, A., Krauskopf, B., and Postlethwaite, C. M. (2017). Climate models with delay differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(11):114309.

Kerg, G., Goyette, K., Touzel, M. P., Gidel, G., Vorontsov, E., Bengio, Y., and Lajoie, G. (2019). Non-normal recurrent neural network (nnRNN): Learning long time dependencies while improving expressivity with transient dynamics. In *Advances in Neural Information Processing Systems*, pages 13591–13601.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707.

Kim, S.-S. (1998). Time-delay recurrent neural network for temporal correlations and prediction. *Neurocomputing*, 20(1-3):253–263.

Kusupati, A., Singh, M., Bhatia, K., Kumar, A., Jain, P., and Varma, M. (2018). FastGRNN: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. *Advances in Neural Information Processing Systems*, 31.

L-Casado, M. and M-Rubio, D. (2019). Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *International Conference on Machine Learning*, pages 3794–3803.

Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43.

Lepri, S., Giacomelli, G., Politi, A., and Arecchi, F. (1994). High-dimensional chaos in delayed dynamical systems. *Physica D: Nonlinear Phenomena*, 70(3):235–249.

Lezcano Casado, M. (2019). Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32.

Liao, X., Chen, G., and Sanchez, E. N. (2002). Delay-dependent exponential stability analysis of delayed neural networks: an LMI approach. *Neural Networks*, 15(7):855–866.

Lim, S. H., Erichson, N. B., Hodgkinson, L., and Mahoney, M. W. (2021). Noisy recurrent neural networks. *Advances in Neural Information Processing Systems*, 34:5124–5137.

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

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150.

Mackey, M. C. and Glass, L. (1977). Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289.

Marcus, C. and Westervelt, R. (1989). Stability of analog neural networks with delay. *Physical Review A*, 39(1):347.

Moreno-Pino, F., Arroyo, Á., Waldon, H., Dong, X., and Cartea, Á. (2024). Rough transformers: Lightweight continuous-time sequence modelling with path signatures. *arXiv preprint arXiv:2405.20799*.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. (2023). Resurrecting recurrent neural networks for long sequences. *arXiv preprint arXiv:2303.06349*.

Park, J. H., Lee, T. H., Liu, Y., and Chen, J. (2019). *Dynamic Systems with Time Delays: Stability and Control*. Springer.

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

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pineda, F. J. (1988). Dynamics and architecture for neural computation. *Journal of Complexity*, 4(3):216–245.

Queiruga, A., Erichson, N. B., Hodgkinson, L., and Mahoney, M. W. (2021). Stateful ode-nets using basis function expansions. *Advances in Neural Information Processing Systems*, 34:21770–21781.

Queiruga, A. F., Erichson, N. B., Taylor, D., and Mahoney, M. W. (2020). Continuous-in-depth neural networks. *arXiv preprint arXiv:2008.02389*.

Rusch, T. K. and Mishra, S. (2021). Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*.

Rusch, T. K., Mishra, S., Erichson, N. B., and Mahoney, M. W. (2022). Long expressive memory for sequence modeling. In *International Conference on Learning Representations*.

Schäfer, A. M. and Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In *International Conference on Artificial Neural Networks*, pages 632–640.

Smith, H. L. (2011). *An Introduction to Delay Differential Equations with Applications to the Life Sciences*, volume 57. Springer New York.

Smith, J. T., Warrington, A., and Linderman, S. W. (2022). Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*.

Tallec, C. and Ollivier, Y. (2018). Can recurrent neural networks warp time? In *International Conference on Learning Representations*.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2022). Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

Voelker, A., Kajić, I., and Eliasmith, C. (2019). Legendre memory units: Continuous-time representation in recurrent neural networks. *Advances in Neural Information Processing Systems*, 32.

Vorontsov, E., Trabelsi, C., Kadoury, S., and Pal, C. (2017). On orthogonality and learning recurrent networks with long term dependencies. In *International Conference on Machine Learning*, pages 3570–3578. PMLR.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339.

Wang, S. and Xue, B. (2023). State-space models with layer-wise nonlinearity are universal approximators with exponential decaying memory. *arXiv preprint arXiv:2309.13414*.

Warden, P. (2018). Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. (2016). Full-capacity unitary recurrent neural networks. *Advances in Neural Information Processing Systems*, 29.

Xia, H., Suliafu, V., Ji, H., Nguyen, T., Bertozzi, A., Osher, S., and Wang, B. (2021). Heavy ball neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 34:18646–18659.

Yang, Z., Zhou, W., and Huang, T. (2014). Exponential input-to-state stability of recurrent neural networks with multiple time-varying delays. *Cognitive Neurodynamics*, 8(1):47–54.

Yu, A., Lyu, D., Lim, S. H., Mahoney, M. W., and Erichson, N. B. (2024a). Tuning frequency bias of state space models. *arXiv preprint arXiv:2410.02035*.

Yu, A., Mahoney, M. W., and Erichson, N. B. (2024b). Hope for a robust parameterization of long-memory state space models. *arXiv preprint arXiv:2405.13975*.

Yu, A., Nigmetov, A., Morozov, D., Mahoney, M. W., and Erichson, N. B. (2023). Robustifying state-space models for long sequences via approximate diagonalization. *arXiv preprint arXiv:2310.01698*.

Zhang, Z., Wu, G., Li, Y., Yue, Y., and Zhou, X. (2021). Deep incremental RNN for learning sequential data: A Lyapunov stable dynamical system. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 966–975. IEEE.

Zhu, Q., Guo, Y., and Lin, W. (2021). Neural delay differential equations. *arXiv preprint arXiv:2102.10801*.

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes. Please refer to Theorem 1 and Proposition 1-2 for the statements of our main theoretical results. See also other results in the appendix.]
   (b) Complete proofs of all theoretical results. [Yes. All proofs are provided in the appendix.]
   (c) Clear explanations of any assumptions. [Yes. Discussions of the assumptions are provided in the main text and the appendix.]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable. As with other RNN papers, we provide the best results after running experiments several times.]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Not applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

This Appendix is organized as follows. In Section A, we provide illustrations to demonstrate the differences between input-driven ODE and DDE dynamics in the scalar setting. In Section B, we provide the proof of Theorem 1. In Section C, we provide proof of Proposition 1. In Section D, we provide gradient bounds for τ-GRU. In Section E, we provide a universal approximation result for a general class of time-delayed RNNs. In Section F-G, we provide additional experimental results and details. In Section H, we make some remarks on comparing our model with state-space models (SSMs).

We are using the following notation throughout this appendix: $\odot$ denotes Hadamard product, $|v|$ (or $\|v\|$) denotes vector norm for the vector v, $\|A\|$ denotes operator norm for the matrix A, $\|A\|_\infty$ denotes the matrix norm induced by ∞-norm (i.e., maximum absolute row sum of the matrix A), σ and $\hat{\sigma}$ (or sigmoid) denote the tanh and sigmoid function, respectively, and $\lceil x \rceil$ and $\lfloor x \rfloor$ denote the ceiling function and floor function in x, respectively. In addition, $\delta_{i,j}$ denotes the Kronecker delta, i.e., $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$.

A Illustrations of Differences Between ODE and DDE Dynamics

Of particular interest to us are the differences between ODEs and DDEs that are driven by an input. To illustrate the differences in the context of RNNs in terms of how they map the input signal into an output signal, we consider the simple examples:

(a) DDE based RNNs with the hidden states $h \in \mathbb{R}$ satisfying $\dot{h} = -h(t-\tau) + u(t)$, with $\tau = 0.5$ and $\tau = 1$, and $h(t) = 0$ for $t \in [-\tau, 0]$, and

(b) an ODE based RNN with the hidden states $h \in \mathbb{R}$ satisfying $\dot{h} = -h(t) + u(t)$,

where $u(t) = \cos(t)$ is the driving input signal.

Figure 5 shows the difference between the dynamics of the hidden state driven by the input signal for (a) and (b). We see that, when compared to the ODE based RNN, the introduced delay causes time lagging in the response of the RNNs to the input signal. The responses are also amplified. In particular, using $\tau = 0.5$ makes the response of the RNN closely matches the input signal. In other words, simply fine tuning the delay parameter τ in the scalar RNN model is sufficient to replicate the dynamics of the input signal.

To further illustrate the differences, we consider the following examples of RNN with a nonlinear activation:

(c) DDE based RNNs with the hidden states $h \in \mathbb{R}$ satisfying $\dot{h} = -h + \tanh(-h(t-\tau) + s(t))$, with $\tau > 0$, and $h(t) = 0$ for $t \in [-\tau, 0]$, and

(d) an ODE based RNN with the hidden states $h \in \mathbb{R}$ satisfying $\dot{h} = -h + \tanh(-h(t) + s(t))$,

where the driving input signal s is taken to be the truncated Weierstrass function:

$$s(t) = \sum_{n=0}^{3} a^{-n} \cos(b^n \cdot \omega t), \tag{22}$$

where $a = 3$, $b = 4$ and $\omega = 2$.

Figure 6 shows the difference between the input-driven dynamics of the hidden states for (c) and (d). We see that, when compared to the ODE based RNN, the introduced delay causes time lagging in the response of the RNNs to the input signal. Even though the response of both RNNs does not match the input signal precisely (since we consider RNNs with one-dimensional hidden states here and therefore their expressivity is limited), we see that using $\tau = 0.5$ produces a response that tracks the seasonality of the input signal better than the ODE based RNN.

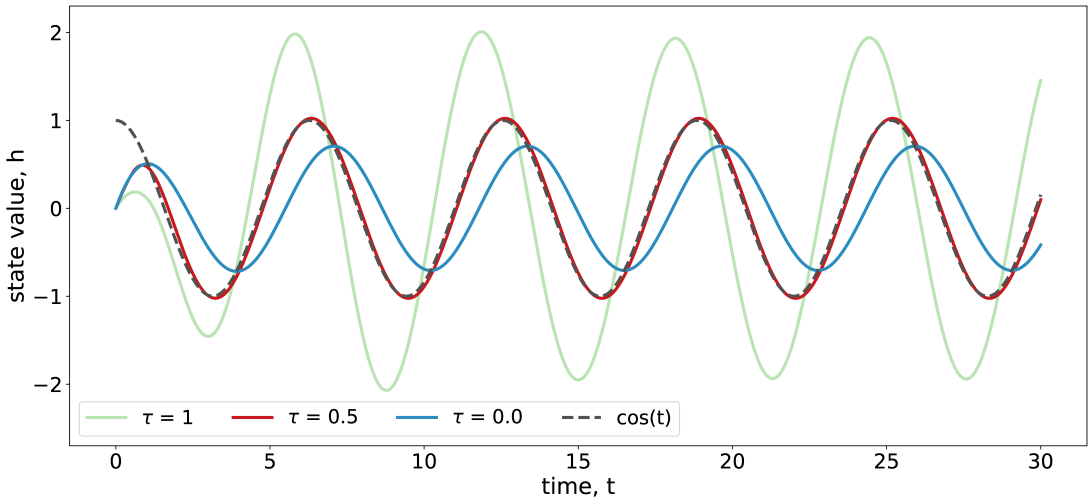


Figure 5: Hidden state dynamics of the DDE based RNNs with $\tau = 0.5$ and $\tau = 1$, and the ODE based RNN ($\tau = 0$). All RNNs are driven by the same cosine input signal.

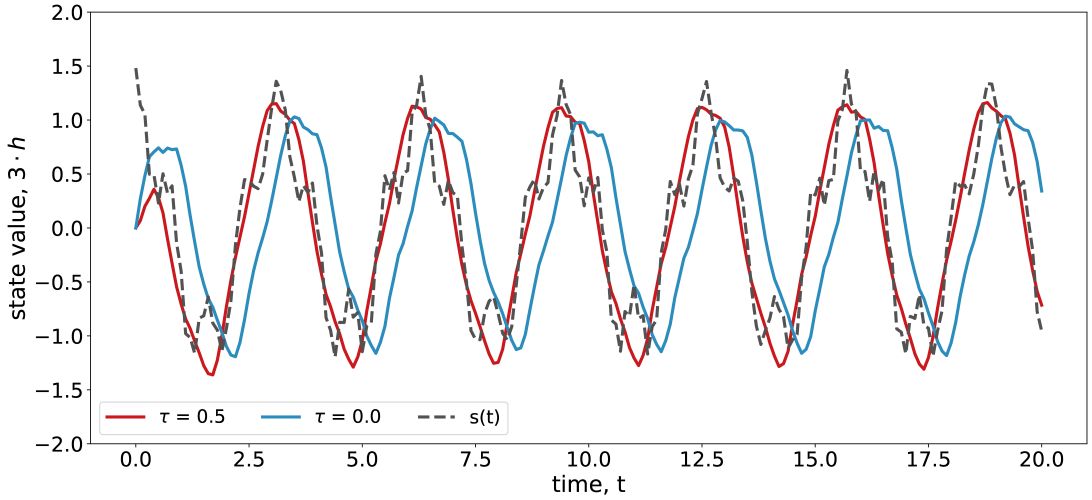


Figure 6: Hidden state dynamics of the DDE based RNN with $\tau = 0.5$ and the ODE based RNN ($\tau = 0$). All RNNs are driven by the same input signal $s(t)$.

B Proof of Theorem 1

In this section, we provide a proof of Theorem 1. To start, note that one can view the solution of the DDE as a mapping of functions on the interval $[t-\tau, t]$ into functions on the interval $[t, t+\tau]$, i.e., as a sequence of functions defined over a set of contiguous time intervals of length τ. This perspective makes it more straightforward to prove existence and uniqueness theorems analogous to those for ODEs than by directly viewing DDEs as an evolution over the state space $\mathbb{R}^d$.

The following theorem, adapted from Theorem 3.7 in (Smith, 2011), provides sufficient conditions for existence and uniqueness of solution through a point $(t_0, \phi) \in \mathbb{R} \times C$ for the IVP (15). Recall that $C := C([-\tau, 0], \mathbb{R}^d)$, the Banach space of continuous functions from $[-\tau, 0]$ into $\mathbb{R}^d$ with the topology of uniform convergence. It is equipped with the norm $\|\phi\| := \sup\{|\phi(\theta)| : \theta \in [-\tau, 0]\}$.

Theorem 2 (Adapted from Theorem 3.7 in (Smith, 2011))**.** *Let $t_0 \in \mathbb{R}$ and $\phi \in C$ be given. Assume that f is continuous and satisfies the Lipschitz condition on each bounded subset of $\mathbb{R} \times C$, i.e., for all $a, b \in \mathbb{R}$, there exists a constant $K > 0$ such that*

$$|f(t, \phi) - f(t, \psi)| \leq K\|\phi - \psi\|, \ t \in [a, b], \ \|\phi\|, \|\psi\| \leq M, \tag{23}$$

with K possibly dependent on a, b, M.

There exists $A > 0$, depending only on M such that if $\phi \in C$ satisfies $\|\phi\| \leq M$, then there exists a unique solution $h(t) = h(t, \phi)$ of Eq. (15), defined on $[t_0 - \tau, t_0 + A]$. Moreover, if K is the Lipschitz constant for f corresponding to $[t_0, t_0 + A]$ and M, then

$$\max_{\eta \in [t_0 - \tau, t_0 + A]} |h(\eta, \phi) - h(\eta, \psi)| \leq \|\phi - \psi\| e^{KA}, \ \|\phi\|, \|\psi\| \leq M. \tag{24}$$

We now provide existence and uniqueness result for the continuous-time τ-GRU model, assuming that the input x is continuous in t. As before, we define the state $h_t \in C$ as:

$$h_t(\theta) := h(t + \theta), \ -\tau \leq \theta \leq 0. \tag{25}$$

Then the DDE defining the model can be formulated as the following IVP for the nonautonomous system:

$$\dot{h}(t) = -h(t) + u(t, h(t)) + a(t, h(t)) \odot z(t, h_t), \ t \geq t_0, \tag{26}$$

and $h_{t_0} = \phi \in C$ for some initial time $t_0 \in \mathbb{R}$, with the dependence on $x(t)$ casted as dependence on t for notational convenience.

Now we restate Theorem 1 from the main text and provide the proof.

Theorem 3 (Existence and uniqueness of solution for continuous-time τ-GRU)**.** *Let $t_0 \in \mathbb{R}$ and $\phi \in C$ be given. There exists a unique solution $h(t) = h(t, \phi)$ of Eq. (17), defined on $[t_0 - \tau, t_0 + A]$ for any $A > 0$. In particular, the solution exists for all $t \geq t_0$, and*

$$\|h_t(\phi) - h_t(\psi)\| \leq \|\phi - \psi\| e^{K(t - t_0)}, \tag{27}$$

for all $t \geq t_0$, where $K = 1 + \|W_1\| + \|W_2\| + \|W_4\|/4$.

Proof. We shall apply Theorem 2. To verify the Lipschitz condition: for any $\phi, \psi \in C$,

$$\begin{aligned}
&|(u(t, \phi) + a(t, \phi) \odot z(t, \phi) - \phi) - (u(t, \psi) + a(t, \psi) \odot z(t, \psi) - \psi)| \\
&\quad \leq |u(t, \phi) - u(t, \psi)| + |\phi - \psi| + |(a(t, \phi) - a(t, \psi)) \odot z(t, \phi)| + |a(t, \psi) \odot (z(t, \phi) - z(t, \psi))| && (28) \\
&\quad \leq \|W_1\| \cdot |\phi - \psi| + |\phi - \psi| + \frac{1}{4}\|W_4\| \cdot |\phi - \psi| + \|W_2\| \cdot |\phi - \psi| && (29) \\
&\quad =: K|\phi - \psi|, && (30)
\end{aligned}$$

where we have used the fact that the tanh and sigmoid are Lipschitz continuous, both bounded by one in absolute value, and they have positive derivatives of magnitude no larger than one and 1/4 respectively in the last inequality above.

Therefore, we see that the right hand side function satisfies a global Lipschitz condition and so the result follows from Theorem 2. □

C Proof of Proposition 1

In this section, we restate Proposition 1, and provide its proof and some remarks.

Proposition 2. *Consider the linear time-delayed RNN whose hidden states are described by the update equation:*

$$h_{n+1} = Ah_n + Bh_{n-m} + Cu_n, \ n = 0, 1, \dots, \tag{31}$$

and $h_n = 0$ for $n = -m, -m+1, \dots, 0$ with $m > 0$. Then, assuming that A and B commute, we have:

$$\frac{\partial h_{n+1}}{\partial u_i} = A^{n-i}C, \tag{32}$$

for $n = 0, 1, \dots, m$, $i = 0, \dots, n$, and

$$\begin{aligned}\frac{\partial h_{m+1+j}}{\partial u_i} &= A^{m+j-i}C + \delta_{i,j-1}BC + 2\delta_{i,j-2}ABC \\ &\quad + 3\delta_{i,j-3}A^2BC + \cdots + j\delta_{i,0}A^{j-1}BC,\end{aligned} \tag{33}$$

for $j = 1, 2, \dots, m+1$, $i = 0, 1, \dots, m+j$, where $\delta_{i,j}$ denotes the Kronecker delta.

Proof. Note that by definition $h_i = 0$ for $i = -m, -m+1, \dots, 0$, and, upon iterating the recursion (31), one obtains:

$$h_{n+1} = A^nCu_0 + A^{n-1}Cu_1 + \cdots + ACu_{n-1} + Cu_n, \tag{34}$$

for $n = 0, 1, \dots, m$.

Now, applying the above formula for h_1, we obtain

$$\begin{aligned}h_{m+2} &= Ah_{m+1} + Bh_1 + Cu_{m+1} \\ &= (B + A^{m+1})Cu_0 + A^mCu_1 + \cdots + A^2Cu_{m-1} + ACu_m + Cu_{m+1}.\end{aligned} \tag{35}$$

Likewise, we obtain:

$$\begin{aligned}h_{m+3} &= Ah_{m+2} + Bh_2 + Cu_{m+2} \\ &= (BA + A^{m+2} + AB)Cu_0 + (B + A^{m+1})Cu_1 + A^mCu_2 + \cdots + ACu_{m+1} + Cu_{m+2} \\ &= (2AB + A^{m+2})Cu_0 + (B + A^{m+1})Cu_1 + A^mCu_2 + \cdots + ACu_{m+1} + Cu_{m+2},\end{aligned} \tag{36}$$

where we have used commutativity of A and B in the last line above.

Applying the above procedure repeatedly and using commutativity of A and B give:

$$h_{m+1+j} = (A^{m+j} + jA^{j-1}B)Cu_0 + (A^{m+j-1} + (j-1)A^{j-2}B)Cu_1 + \cdots + ACu_{m+j-1} + Cu_{m+j}, \tag{37}$$

for $j = 1, 2, \dots, m+1$.

The formula in the proposition then follows upon taking the derivative with respect to the u_i in the above formula for the hidden states h_k. □

We remark that one could also derive formula for the gradients $\frac{\partial h_{n+1+j}}{\partial u_i}$ for $n \geq 2m+1$ as well as those for our τ-GRU architecture analogously, albeit the resulting expression is quite complicated. In particular, the dependence on higher powers of B for the coefficients in front of the Kronecker deltas would appear in the formula for the former case (with much more complicated expressions without assuming commutativity of the matrices). However, we emphasize that the qualitative conclusion derived from the analysis remains the same: that introduction of delays places more emphasis on gradient information due to input elements in the past, so they act as buffers to propagate the gradient information more effectively than the counterpart models without delays.

D Gradient Bounds for τ-GRU

On the exploding and vanishing gradient problem. For simplicity of our discussion here, we consider the loss function:

$$\mathcal{E}_n = \frac{1}{2}\|y_n - \overline{y}_n\|^2, \tag{38}$$

where $n = 1, \dots, N$ and $\overline{y}_n$ denotes the underlying growth truth. The training of τ-GRU involves computing gradients of this loss function with respect to its underlying parameters $\theta \in \Theta = [W_{1,2,3,4}, U_{1,2,3,4}, V]$ at each iteration of the gradient descent algorithm. Using chain rule, we obtain (Pascanu et al., 2013):

$$\frac{\partial \mathcal{E}_n}{\partial \theta} = \sum_{k=1}^{n} \frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}, \tag{39}$$

where

$$\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta} = \frac{\partial \mathcal{E}_n}{\partial h_n}\frac{\partial h_n}{\partial h_k}\frac{\partial^+ h_k}{\partial \theta}, \tag{40}$$

with $\frac{\partial^+ h_k}{\partial \theta}$ denoting the "immediate" partial derivative of the state h_k with respect to θ, i.e., where h_{k-1} is taken as a constant with respect to θ (Pascanu et al., 2013).

The partial gradient $\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta}$ measures the contribution to the hidden state gradient at step n due to the information at step k. It can be shown that this gradient behaves as

$$\frac{\partial \mathcal{E}_n^{(k)}}{\partial \theta} \sim \gamma^{n-k}, \tag{41}$$

for some $\gamma > 0$ (Pascanu et al., 2013). If $\gamma > 1$, then the gradient grows exponentially in sequence length, for long-term dependencies where $k \ll n$, causing the exploding gradient problem. On the other hand, if $\gamma < 1$, then the gradient decays exponentially for $k \ll n$, causing the vanishing gradient problem. Therefore, we can investigate how τ-GRU deals with these problems by deriving bounds on the gradients. In particular, we are interested in the behavior of the gradients for long-term dependencies, i.e., $k \ll n$, and shall show that the delay mechanism in τ-GRU slows down the exponential decay rate, thereby reducing the sensivity to the vanishing gradient problem.

Recall that the update equations defining τ-GRU are given by $h_n = 0$ for $n = -m, -m+1, \dots, 0$,

$$h_n = (1 - g(A_{n-1})) \odot h_{n-1} + g(A_{n-1}) \odot [u(B_{n-1}) + a(C_{n-1}) \odot z(D_{n-m-1})], \tag{42}$$

for $n = 1, 2, \dots, N$, where $m := \lfloor \tau/\Delta t \rfloor \in \{1, 2, \dots, N-1\}$, $A_{n-1} = W_3 h_{n-1} + U_3 x_{n-1}$, $B_{n-1} = W_1 h_{n-1} + U_1 x_{n-1}$, $C_{n-1} = W_4 h_{n-1} + U_4 x_{n-1}$, and $D_{n-m-1} = W_2 h_{n-m-1} + U_2 x_{n-1}$.

In the sequel, we shall denote the ith component of a vector v as v^i and the (i, j) entry of a matrix A as A^{ij}.

We start with the following lemma.

Lemma 1. *For every i, we have $h_n^i = 0$, for $n = -m, -m+1, \dots, 0$, and $|h_n^i| \leq 2$, for $n = 1, 2, \dots, N$.*

Proof. The ith component of the hidden states of τ-GRU are given by: $h_n^i = 0$ for $n = -m, -m+1, \dots, 0$, and

$$h_n^i = (1 - g(A_{n-1}^i))h_{n-1}^i + g(A_{n-1}^i)[u(B_{n-1}^i) + a(C_{n-1}^i)z(D_{n-m-1}^i)], \tag{43}$$

for $n = 1, 2, \dots, N$.

Using the fact that $g(x), a(x) \in (0, 1)$ and $u(x), z(x) \in (-1, 1)$ for all x, we can bound the h_n^i as:

$$\begin{aligned} h_n^i &\leq (1 - g(A_{n-1}^i)) \max(h_{n-1}^i, 2) + g(A_{n-1}^i) \max(h_{n-1}^i, 2) \\ &\leq \max(h_{n-1}^i, 2), \end{aligned} \tag{44}$$

for all i and $n = 1, 2, \dots, N$.

Similarly, we have:

$$\begin{aligned} h_n^i &\geq (1 - g(A_{n-1}^i)) \min(-2, h_{n-1}^i) + g(A_{n-1}^i) \min(-2, h_{n-1}^i) \\ &\geq \min(-2, h_{n-1}^i), \end{aligned} \tag{45}$$

for all i and $n = 1, 2, \ldots, N$.

Thus,

$$\min(-2, h_{n-1}^i) \leq h_n^i \leq \max(h_{n-1}^i, 2), \tag{46}$$

for all i and $n = 1, 2, \ldots, N$.

Now, iterating over n and using $h_0^i = 0$ for all i, we obtain $-2 \leq h_n^i \leq 2$ for all i and $n = 1, 2, \ldots, N$.

□

We now provide the gradients bound for τ-GRU in the following proposition and proof.

Proposition 3. *Assume that there exists an $\epsilon > 0$ such that $\max_n g(A_{n-1}^i) \geq \epsilon$ and $\max_n a(C_{n-1}^i) \geq \epsilon$ for all i. Then*

$$\begin{aligned} \left\| \frac{\partial h_n}{\partial h_k} \right\|_\infty &\leq (1 + C - \epsilon)^{n-k} \\ &\quad + \|W_2\|_\infty \cdot \left((1 + C - \epsilon)^{n-k-2} \delta_{m,1} + \cdots + (1 + C - \epsilon) \delta_{m,n-k-2} + \delta_{m,n-k-1} \right), \end{aligned} \tag{47}$$

for $n = 1, \ldots, N$ and $k < n$, where $C = \|W_1\|_\infty + \|W_3\|_\infty + \frac{1}{4}\|W_4\|_\infty$.

Proof. Recall $h_n = 0$ for $n = -m, -m+1, \ldots, 0$, and

$$h_n = (1 - g(A_{n-1})) \odot h_{n-1} + g(A_{n-1}) \odot [u(B_{n-1}) + a(C_{n-1}) \odot z(D_{n-m-1})] =: F(h_{n-1}, h_{n-m-1}), \tag{48}$$

for $n = 1, 2, \ldots, N$.

Denote $q_{n,l} := \frac{\partial F}{\partial h_{n-l}}$, where $F := F(h_{n-1}, h_{n-l})$ for $l > 1$. The gradients $\frac{\partial h_n}{\partial h_k}$ can be computed recursively as follows.

$$p_n^{(1)} := \frac{\partial h_n}{\partial h_{n-1}}, \tag{49}$$

$$p_n^{(2)} := \frac{\partial h_n}{\partial h_{n-2}} = p_n^{(1)} p_{n-1}^{(1)} + q_{n,2} \delta_{m,1}, \tag{50}$$

$$p_n^{(3)} := \frac{\partial h_n}{\partial h_{n-3}} = p_n^{(1)} p_{n-1}^{(2)} + q_{n,3} \delta_{m,2}, \tag{51}$$

$$\vdots \tag{52}$$

$$p_n^{(n-k)} := \frac{\partial h_n}{\partial h_k} = p_n^{(1)} p_{n-1}^{(n-k-1)} + q_{n,n-k} \delta_{m,n-k-1}. \tag{53}$$

As $\|p_n^{(n-k)}\| \leq \|p_n^{(1)}\| \cdot \|p_{n-1}^{(n-k-1)}\| + \|q_{n,n-k}\| \delta_{m,n-k-1}$, it remains to upper bound the $p_n^{(1)}$ and $q_{n,n-k}$.

The ith component of the hidden states can be written as:

$$h_n^i = (1 - g(A_{n-1}^i)) h_{n-1}^i + g(A_{n-1}^i)[u(B_{n-1}^i) + a(C_{n-1}^i) z(D_{n-m-1}^i)], \tag{54}$$

where $A_{n-1}^i = W_3^{iq} h_{n-1}^q + U_3^{ir} x_{n-1}^r$, $B_{n-1}^i = W_1^{iq} h_{n-1}^q + U_1^{ir} x_{n-1}^r$, $C_{n-1}^i = W_4^{iq} h_{n-1}^q + U_4^{ir} x_{n-1}^r$, and $D_{n-m-1}^i = W_2^{iq} h_{n-m-1}^q + U_2^{ir} x_{n-1}^r$, using Einstein's summation notation for repeated indices.

Therefore, applying chain rule and using Einstein's summation for repeated indices in the following, we obtain:

$$\begin{aligned}\frac{\partial h_n^i}{\partial h_{n-1}^j} &= (1-g(A_{n-1}^i))\frac{\partial h_{n-1}^i}{\partial h_{n-1}^j} - \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^l}\frac{\partial A_{n-1}^l}{\partial h_{n-1}^j}h_{n-1}^i + g(A_{n-1}^i)\frac{\partial u(B_{n-1}^i)}{\partial B_{n-1}^l}\frac{\partial B_{n-1}^l}{\partial h_{n-1}^j} \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^l}\frac{\partial A_{n-1}^l}{\partial h_{n-1}^j}u(B_{n-1}^i) + g(A_{n-1}^i)\frac{\partial a(C_{n-1}^i)}{\partial C_{n-1}^l}\frac{\partial C_{n-1}^l}{\partial h_{n-1}^j}z(D_{n-m-1}^i) \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^l}\frac{\partial A_{n-1}^l}{\partial h_{n-1}^j}a(C_{n-1}^i)z(D_{n-m-1}^i). \end{aligned} \tag{55}$$

Noting that $\frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^l} = 0$, $\frac{\partial u(B_{n-1}^i)}{\partial B_{n-1}^l} = 0$ and $\frac{\partial a(C_{n-1}^i)}{\partial C_{n-1}^l} = 0$ for $i \neq l$, we have:

$$\begin{aligned}\frac{\partial h_n^i}{\partial h_{n-1}^j} &= (1-g(A_{n-1}^i))\frac{\partial h_{n-1}^i}{\partial h_{n-1}^j} - \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}\frac{\partial A_{n-1}^i}{\partial h_{n-1}^j}h_{n-1}^i + g(A_{n-1}^i)\frac{\partial u(B_{n-1}^i)}{\partial B_{n-1}^i}\frac{\partial B_{n-1}^i}{\partial h_{n-1}^j} \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}\frac{\partial A_{n-1}^i}{\partial h_{n-1}^j}u(B_{n-1}^i) + g(A_{n-1}^i)\frac{\partial a(C_{n-1}^i)}{\partial C_{n-1}^i}\frac{\partial C_{n-1}^i}{\partial h_{n-1}^j}z(D_{n-m-1}^i) \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}\frac{\partial A_{n-1}^i}{\partial h_{n-1}^j}a(C_{n-1}^i)z(D_{n-m-1}^i) \end{aligned} \tag{56}$$

$$\begin{aligned}&= (1-g(A_{n-1}^i))\delta_{i,j} - \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}W_3^{ij}h_{n-1}^i + g(A_{n-1}^i)\frac{\partial u(B_{n-1}^i)}{\partial B_{n-1}^i}W_1^{ij} \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}W_3^{ij}u(B_{n-1}^i) + g(A_{n-1}^i)\frac{\partial a(C_{n-1}^i)}{\partial C_{n-1}^i}W_4^{ij}z(D_{n-m-1}^i) \\ &\quad + \frac{\partial g(A_{n-1}^i)}{\partial A_{n-1}^i}W_3^{ij}a(C_{n-1}^i)z(D_{n-m-1}^i). \end{aligned} \tag{57}$$

Using the assumption that $\max_n g(A_{n-1}^i), \max_n a(C_{n-1}^i) \geq \epsilon$ for all i, the fact that $|z(x)|, |u(x)| \leq 1$, $g(x), a(x) \in (0,1)$, $g'(x), a'(x) \leq 1/4$, $u'(x) \leq 1$ for all $x \in \mathbb{R}$, and Lemma 1, we obtain:

$$\left|\frac{\partial h_n^i}{\partial h_{n-1}^j}\right| \leq (1-\epsilon)\delta_{i,j} + \frac{1}{4}|W_3^{ij}| \cdot |h_{n-1}^i| + |W_1^{ij}| + \frac{1}{4}|W_3^{ij}| + \frac{1}{4}|W_4^{ij}| + \frac{1}{4}|W_3^{ij}| \tag{58}$$

$$\leq (1-\epsilon)\delta_{i,j} + |W_1^{ij}| + |W_3^{ij}| + \frac{1}{4}|W_4^{ij}|. \tag{59}$$

Therefore,

$$\left\|\frac{\partial h_n}{\partial h_{n-1}}\right\|_\infty := \max_{i=1,\dots,d}\sum_{j=1}^d \left|\frac{\partial h_n^i}{\partial h_{n-1}^j}\right| \leq (1-\epsilon) + \|W_1\|_\infty + \|W_3\|_\infty + \frac{1}{4}\|W_4\|_\infty =: 1-\epsilon+C. \tag{60}$$

Likewise, we obtain, for $l > 1$:

$$\frac{\partial F^i}{\partial h_{n-l}^j} = g(A_{n-1}^i)a(C_{n-1}^i)\frac{\partial z(D_{n-l}^i)}{\partial D_{n-l}^i}\frac{\partial D_{n-l}^i}{\partial h_{n-l}^j} \tag{61}$$

$$= g(A_{n-1}^i)a(C_{n-1}^i)\frac{\partial z(D_{n-l}^i)}{\partial D_{n-l}^i}W_2^{ij}. \tag{62}$$

Using the fact that $|g(x)|, |a(x)| \leq 1$ and $|z'(x)| \leq 1$ for all $x \in \mathbb{R}$, we obtain:

$$\left|\frac{\partial F^i}{\partial h_{n-l}^j}\right| \leq |W_2^{ij}|, \tag{63}$$

for $l > 1$, and thus $\|q_{n,n-k}\|_\infty = \|\frac{\partial F}{\partial h_k}\|_\infty \leq \|W_2\|_\infty$ for $k > 1$.

The upper bound in the proposition follows by using the above bounds for $\|p_n^{(1)}\|_\infty$ and $\|q_{n,n-k}\|_\infty$, and iterating the recursion $\|p_n^{(n-k)}\| \leq \|p_n^{(1)}\| \cdot \|p_{n-1}^{(n-k-1)}\| + \|q_{n,n-k}\|\delta_{m,n-k-1}$ over k.

□

From Proposition 3, we see that if $\epsilon > C$, then the gradient norm decays exponentially as k becomes large. However, the delay in τ-GRU introduces jump-ahead connections (buffers) to slow down the exponential decay. For instance, choosing $m = 1$ for the delay, we have $\left\|\frac{\partial h_n}{\partial h_k}\right\| \sim (1 + C - \epsilon)^{n-k-2}$ as $k \to \infty$ (instead of $\left\|\frac{\partial h_n}{\partial h_k}\right\| \sim (1 + C - \epsilon)^{n-k-1}$ as $k \to \infty$ in the case when no delay is introduced into the model). The larger the m is, the more effective the delay is able to slow down the exponential decay of the gradient norm. These qualitative conclusions can already be derived by studying the linear time-delayed RNN, which we consider in the main text for simplicity.

E Approximation Capability of Time-Delayed RNNs

RNNs (without delay) have been shown to be universal approximators of a large class of open dynamical systems (Schäfer and Zimmermann, 2006). Analogously, RNNs with delay can be shown to be universal approximators of open dynamical systems with delay.

Let $m > 0$ (time lag) and consider the state space models (which, in this section, we shall simply refer to as delayed RNNs) of the form:

$$\begin{aligned} s_{n+1} &= f(As_n + Bs_{n-m} + Cu_n + b), \\ r_n &= Ds_n, \end{aligned} \tag{64}$$

and dynamical systems of the form

$$\begin{aligned} x_{n+1} &= g(x_n, x_{n-m}, u_n), \\ o_n &= o(x_n), \end{aligned} \tag{65}$$

for $n = 0, 1, \ldots, N$. Here $u_n \in \mathbb{R}^{d_u}$ is the input, $o_n \in \mathbb{R}^{d_o}$ is the target output of the dynamical systems to be learned, $s_n \in \mathbb{R}^d$ is the hidden state of the learning model, $r_n \in \mathbb{R}^q$ is the model output, f is the tanh function applied component-wise, the maps g and o are Lipschitz continuous, and the matrices A, B, C, D and the vector b are learnable parameters. For simplicity, we take the initial functions to be $s_n = y_n = 0$ for $n = -m, -m+1, \ldots, 0$.

The following theorem shows that the delayed RNNs (64) are capable of approximating a large class of time-delay dynamical systems, of the form (65), to arbitrary accuracy.

Theorem 4. *Assume that there exists a constant $R > 0$ such that $\max(\|x_{n+1}\|, \|u_n\|) < R$ for $n = 0, 1, \ldots, N$. Then, for a given $\epsilon > 0$, there exists a delayed RNN of the form (64) such that the following holds for some d:*

$$\|r_n - o_n\| \leq \epsilon, \tag{66}$$

for $n = 0, 1, \ldots, N$.

Proof. The proof proceeds along the line of (Schäfer and Zimmermann, 2006; Rusch et al., 2022), using the universal approximation theorem (UAT) for feedforward neural network maps and with straightforward modification to deal with the extra delay variables s_{n-n} and x_{n-m} here. The proof proceeds in a similar manner as the one provided in Section E.4 in (Rusch et al., 2022). The goal is to construct hidden states, output state, weight matrices and bias vectors such that an output of the delayed RNN approximates the dynamical system (65).

Let $\epsilon > \epsilon^* > 0, R^* > R \gg 1$ be parameters to be defined later. Then, using the UAT for continuous functions with neural networks with the tanh activation function (Barron, 1993), we can obtain the following statements. Given an ϵ^*, there exist weight matrices W_1, W_2, W_3, V_1 and a bias vector b_1 of appropriate dimensions such that the neural network defined by $\mathcal{N}_1(h, \tilde{h}, u) := W_3 \tanh(W_1 h + W_2 \tilde{h} + V_1 u + b_1)$ approximates the underlying function g as follows:

$$\max_{\max(\|h\|, \|\tilde{h}\|, \|u\|) < R^*} \|g(h, \tilde{h}, u) - \mathcal{N}_1(h, \tilde{h}, u)\| \leq \epsilon^*. \tag{67}$$

Now, we define the dynamical system:

$$p_n = W_3 \tanh(W_1 p_{n-1} + W_2 p_{n-m-1} + V_1 u_{n-1} + b_1), \tag{68}$$

with $p_i = 0$ for $i = -m, -m+1, \ldots, 0$.

Then using the above approximation bound, we obtain, for $n = 1, \ldots, N+1$,

$$\begin{aligned}
\|x_n - p_n\| &= \|g(x_{n-1}, x_{n-m-1}, u_{n-1}) - p_n\| && (69)\\
&\leq \|g(x_{n-1}, x_{n-m-1}, u_{n-1}) - W_3 \tanh(W_1 p_{n-1} + W_2 p_{n-m-1} + V_1 u_{n-1} + b_1)\| && (70)\\
&\leq \|g(x_{n-1}, x_{n-m-1}, u_{n-1}) - g(p_{n-1}, p_{n-m-1}, u_{n-1})\| \\
&\quad + \|g(p_{n-1}, p_{n-m-1}, u_{n-1}) - W_3 \tanh(W_1 p_{n-1} + W_2 p_{n-m-1} + V_1 u_{n-1} + b_1)\| && (71)\\
&\leq Lip(g)(\|x_{n-1} - p_{n-1}\| + \|x_{n-m-1} - p_{n-m-1}\|) + \epsilon^*, && (72)
\end{aligned}$$

where $Lip(g)$ is the Lipschitz constant of g on the compact set $\{(h, \tilde{h}, u) : \|h\|, \|\tilde{h}\|, \|u\| < R^*\}$.

Iterating the above inequality over n leads to:

$$\|x_n - p_n\| \leq \epsilon^* C_1(n, m, Lip(g)), \tag{73}$$

for some constant $C_1 > 0$ that is dependent on $n, m, Lip(g)$.

Using the Lipschitz continuity of the output function o, we obtain:

$$\|o_n - o(p_n)\| \leq \epsilon^* C_2(n, m, Lip(g), Lip(o)), \tag{74}$$

for some constant C_2 that is dependent on $n, m, Lip(g), Lip(o)$, where $Lip(o)$ is the Lipschitz constant of o on the compact set $\{h : \|h\| < R^*\}$.

Next we use the UAT for neural networks again to obtain the following approximation result. Given an $\bar{\epsilon}$, there exists weight matrices W_4, W_5 and bias vector b_2 of appropriate dimensions such that the tanh neural network, $\mathcal{N}_2(h) := W_5 \tanh(W_4 h + b_2)$ approximates the underlying output function o as:

$$\max_{\|h\| < R^*} \|o(h) - \mathcal{N}_2(h)\| \leq \bar{\epsilon}. \tag{75}$$

Defining $\bar{o}_n = W_5 \tanh(W_4 p_n + b_2)$, we obtain, using the above approximation bound and the inequality (74):

$$\|o_n - \bar{o}_n\| = \|o_n - o(p_n)\| + \|o(p_n) - \bar{o}_n\| \leq \epsilon^* C_2(n, m, Lip(g), Lip(o)) + \bar{\epsilon}. \tag{76}$$

Now, let us denote:

$$\tilde{p}_n = \tanh(W_1 p_{n-1} + W_2 p_{n-m-1} + V_1 u_{n-1} + b_1), \tag{77}$$

so that $p_n = W_3 \tilde{p}_n$. With this notation, we have:

$$\bar{o}_n = W_5 \tanh(W_4 W_3 \tanh(W_1 W_3 \tilde{p}_{n-1} + W_2 W_3 \tilde{p}_{n-m-1} + V_1 u_{n-1} + b_1) + b_2). \tag{78}$$

Since the function $R(y) = W_5 \tanh(W_4 W_3 \tanh(W_1 W_3 y + W_2 W_3 \tilde{p}_{n-m-1} + V_1 u_{n-1} + b_1) + b_2)$ is Lipschitz continuous in y, we can apply the UAT again to obtain: for any $\tilde{\epsilon}$, there exists weight matrices W_6, W_7 and bias vector b_3 of appropriate dimensions such that

$$\max_{\|y\| < R^*} \|R(y) - W_7 \tanh(W_6 y + b_3)\| \leq \tilde{\epsilon}. \tag{79}$$

Denoting $\tilde{o}_n := W_7 \tanh(W_6 p_{n-1} + b_3)$ and using the above approximation bound, we obtain $\|\bar{o}_n - \tilde{o}_n\| \leq \tilde{\epsilon}$.

Finally, we collect all the ingredients above to construct a delayed RNN that can approximate the dynamical system (65). To this end, we define the hidden states (in an enlarged state space): $s_n := (\tilde{p}_n, \hat{p}_n)$, with $\tilde{p}_n$, $\hat{p}_n$ sharing the same dimension. These hidden states evolve according to the dynamical system:

$$s_n = \tanh\left(\begin{bmatrix} W_1 W_3 & 0 \\ W_6 W_3 & 0 \end{bmatrix} s_{n-1} + \begin{bmatrix} W_2 W_3 & 0 \\ 0 & 0 \end{bmatrix} s_{n-m-1} + \begin{bmatrix} V_1 u_{n-1} \\ 0 \end{bmatrix} + \begin{bmatrix} b_1 \\ 0 \end{bmatrix} \right). \tag{80}$$

Defining the output state as $r_n := [0, W_7] s_n$, with the s_n satisfying the above system, we arrive at a delayed RNN that approximates the dynamical system (65). In fact, we can verify that $r_n = \tilde{o}_n$. Setting $\bar{\epsilon} < \epsilon/2$ and $\epsilon^* < \epsilon/(2C_2(n, m, Lip(g), Lip(o)))$ give us the bound (66) in the theorem. □

F Additional Details and Experiments

In this section, we provide additional empirical results and details to demonstrate the advantages of τ-GRU when compared to other RNN architectures. As with many RNN papers, we report the performance for the single best model in our experiments. Rather than running all the baseline models ourselves, we report the results from the corresponding papers since we assume that the authors have done the best job tuning their respective model. In a few cases we train the baseline models and indicate the results by a "*".

F.1 Copy Task

Here we consider the copy task as an additional benchmark problem to assess the ability of τ-GRU to handle long-range dependencies. The copy task was originally proposed by Hochreiter and Schmidhuber (1997), and requires the model to memorize the initial 10 elements of the input sequence for the duration of T time steps. Then the model is tasked with outputting the initial elements as accuratelty as possible.

Here we consider a long sequence with $T = 1000$. The results are shown in Figure 7. It can be seen that our τ-GRU is able to solve the problem, whereas other models such as LEM or TARNN show inferior performance.

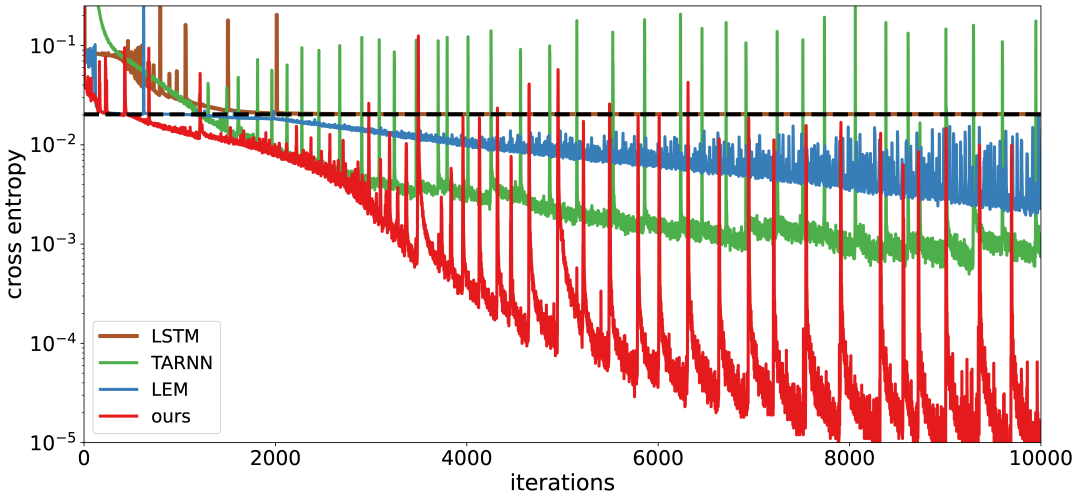


Figure 7: Cross-entropy as a function of iterations for the copy task for different recurrent models. Here we consider the copy task with a sequence lengths $T = 1000$.

F.2 Speech Recognition: Google 12

Here, we consider the Google Speech Commands data set V2 (Warden, 2018) to demonstrate the performance of our model for speech recognition. The aim of this task is to learn a model that can classify a short audio sequence, which is sampled at a rate of 16 kHz from 1 second utterances of 2,618 speakers. We consider the Google 12-label task (Google12) which is composed of 10 keyword classes, and in addition one class that corresponds to 'silence', and a class corresponding to 'unknown' keywords. We adopt the standard train/validation/test set split for evaluating our model, and we use dropout, applied to the inputs, with rate 0.03 to reduce overfitting.

Table 7 presents the results for our τ-GRU and a number of competitive RNN architectures. We adopt the results for the competitive RNNs from (Rusch et al., 2022). Our proposed τ-GRU shows the best performance on this task, i.e., τ-GRU is able to outperform gated and continuous-time RNNs on this task that requires an expressive recurrent unit.

Table 7: Test accuracy results for Google12. Results indicated by * are produces by us, results indicated by + are from (Rusch et al., 2022).

Model	Accuracy (%)	# units	# param
tanh RNN (Rusch et al., 2022)+	73.4	128	27k
LSTM (Rusch et al., 2022)+	94.9	128	107k
GRU (Rusch et al., 2022)+	95.2	128	80k
AsymRNN (Chang et al., 2018)+	90.2	128	20k
expRNN L-Casado and M-Rubio (2019)+	92.3	128	19k
coRNN (Rusch and Mishra, 2021)+	94.7	128	44k
Fast GRNN (Kusupati et al., 2018)+	94.8	128	27k
LEM (Rusch et al., 2022)	95.7	128	107k
Lipschitz RNN (Erichson et al., 2020)*	95.6	128	34k
Noisy RNN (Lim et al., 2021)*	95.7	128	34k
iRNN (Kag et al., 2020)*	95.1	-	8.5k
TARNN (Kag and Saligrama, 2021)*	95.9	128	107k
ours	**96.2**	128	107k

F.3 Learning the Dynamics of Mackey-Glass System

Here, we consider the task of learning the Mackey-Glass equation, originally introduced in (Mackey and Glass, 1977) to model the variation in the relative quantity of mature cells in the blood:

$$\dot{x} = a\frac{x(t-\delta)}{1+x^n(t-\delta)} - bx(t),\ t \geq \delta, \tag{81}$$

where $\delta \geq 17$, $a, b, n > 0$, with x satisfying $\dot{x} = ax(0)/(1+x(0)^n) - bx$ for $t \in [0, \delta]$. It is a scalar equation with chaotic dynamics, with infinite-dimensional state space. Increasing the value of δ increases the dimension of the attractor.

For data generation, we choose $a = 0.2$, $b = 0.1$, $n = 10$, $\delta = 17$, $x(0) \sim \text{Unif}(0, 1)$, and use the classical Runge-Kutta method (RK4) to integrate the system numerically from $t = 0$ to $t = 1000$ with a step-size of 0.25. The training and testing samples are the time series (of length 2000) generated by the RK4 scheme on the interval $[500, 1000]$ for different realizations of $x(0)$. Figure 8 shows a realization of the trajectory produced by the Mackey-Glass system (and also the DDE based ENSO system considered in the main text).

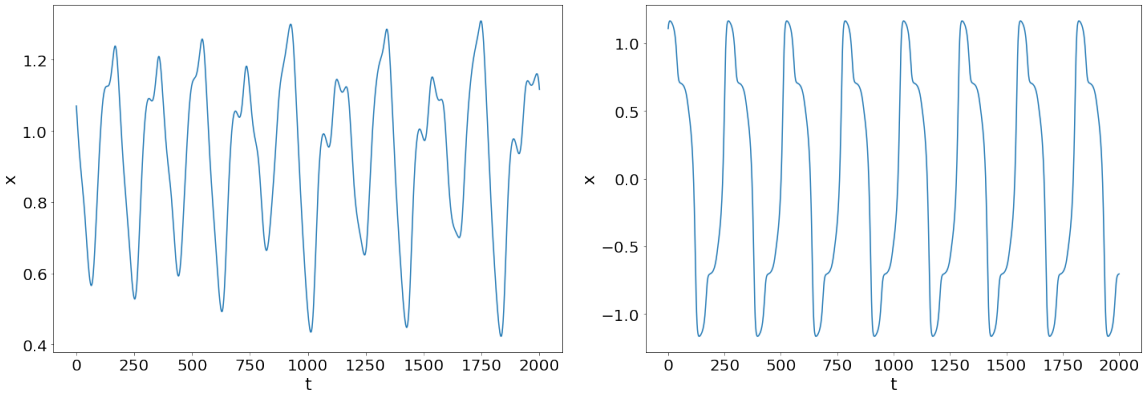


Figure 8: A realization of the Mackey-Glass dynamics (left) and the DDE based ENSO dynamics (right).

Table 8 shows that our τ-GRU model (with $\alpha = \beta = 1$ and using $\tau = 10$) is more effective in learning the Mackey-Glass system when compared to other RNN architectures. We also see that the predictive performance deteriorates without making full use of the combination of the standard recurrent unit and delay recurrent unit (setting either α or β to zero). Moreover, τ-GRU demonstrates improved performance when compared to the simple delay GRU model (Eq. (12)) and the counterpart model without using the gating. Similar observation also holds for the ENSO prediction task; see Table 9.

Table 8: Results for the Mackey-Glass system prediction.

Model	MSE ($\times 10^{-2}$)	# units	# parameters
Vanilla RNN	0.3903	16	0.321k
LSTM	0.6679	16	1.233k
GRU	0.4351	16	0.929k
Lipschitz RNN	8.9718	16	0.561k
coRNN	1.6835	16	0.561k
LEM	0.1430	16	1.233k
simple delay GRU (Eq. (12))	0.2772	16	0.897k
ablation (no gating)	0.2765	16	0.929k
ablation ($\alpha = 0$)	0.1553	16	0.625k
ablation ($\beta = 0$)	0.2976	16	0.929k
τ-GRU (ours)	**0.1358**	16	1.233k

Figure 9 shows that our model converges much faster than other RNN models during training. In particular, our model is able to achieve both lower training and testing error (as measured by the root mean square error (RMSE)) with fewer epochs, demonstrating the effectiveness of the delay mechanism in improving the performance on the problem of long-term dependencies. This is consistent with our analysis on how the gradient information is propagated through the delay buffers in the network (see Proposition 1), suggesting that the delay buffers can propagate gradient more efficiently. Similar behavior is also observed for the ENSO task; see Figure 10.

Table 9: Additional results for the ENSO model prediction.

Model	MSE ($\times 10^{-2}$)	# units	# parameters
simple delay GRU (Eq. (12))	0.2317	16	0.897k
ablation (no gating)	0.4289	16	0.929k
τ-GRU (ours)	**0.17**	16	1.2k

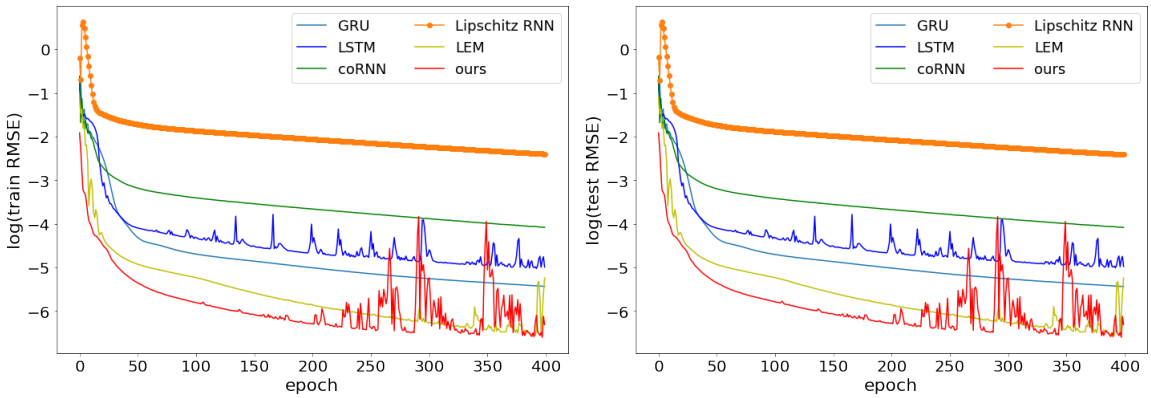


Figure 9: Train RMSE (left) and test RMSE (right) vs. epoch for the Mackey-Glass learning task.

G Tuning Parameters

To tune our τ-GRU, we use a non-exhaustive random search within the following plausible ranges for $\tau = 5, \ldots, 200$. We used Adam as our optimization algorithm for all of the experiments. For the synthetic data sets generated by the ENSO and Mackey-Glass system, we used learning rate of 0.01. For the other experiments we considered learning rates between 0.01 and 0.0005. We used dropout for the IMDB and Google12 task to avoid overfitting.

Table 10 is listing the tuning parameters for the different tasks that we considered in this work.

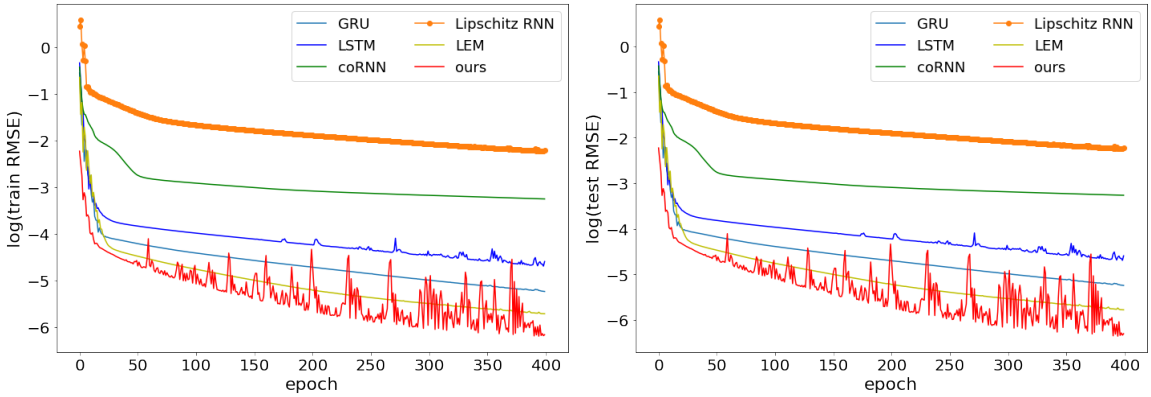


Figure 10: Train RMSE (left) and test RMSE (right) vs. epoch for the ENSO learning task.

Table 10: Summary of tuning parameters.

Name	d	lr	τ	dropout	epochs
Adding Task $N = 2000$	128	0.0026	900	-	200
Adding Task $N = 5000$	128	0.002	2000	-	200
IMDB	128	00012	1	0.04	30
HAR-2	128	0.00153	10	-	100
sMNIST	128	0.0018	50	-	60
psMNIST	128	0.0055	65	-	80
sCIFAR	128	0.0035	30	-	50
nCIFAR	128	0.0022	965	-	50
Google12	128	0.00089	5	0.03	60
ENSO	16	0.01	20	-	400
Mackey-Glass system	16	0.01	10	-	400

Sensitivity to Random Initialization. We evaluate our models for each tasks using 8 seeds. The maximum, minimum, average values, and standard deviations obtained for each task are tabulated in Table 11.

Table 11: Sensitivity to random initialization evaluated over 8 runs with different seeds.

Task	Maximum	Minimum	Average	standard dev.	standard error	d
IMDB	88.7	86.2	87.9	0.82	0.29	128
HAR-2	97.4	96.6	96.9	0.41	0.17	128
sMNIST	99.4	99.1	99.3	0.08	0.02	128
psMNIST	97.3	96.0	96.8	0.39	0.13	128
sCIFAR	74.9	72.65	73.54	0.90	0.41	128
nCIFAR	62.7	61.7	62.3	0.32	0.11	128
Google12	96.2	95.7	95.9	0.17	0.05	128

H τ-GRU is More Parameter-Efficient than SSMs in the Small Data Regime

The state-space models (SSMs) are a class of models which uses structured linear RNNs representations as layers in a deep learning pipeline (Gu et al., 2021b,a; Voelker et al., 2019). They originate from the HIPPO framework which offers an optimal solution to a natural online function approximation problem (Gu et al., 2020). While they have been shown to demonstrate state-of-the-art sequence modeling performance in several domains, these models may not be optimal for tasks in the small data regime, which is quite common in many scientific applications where the available data is often scarce. Therefore, developing lightweight yet effective models that can perform

well in this scenario is valuable.

To support this claim, we conduct an additional experiment to compare the performance of S4 model (Gu et al., 2021a) and LMU (Voelker et al., 2019) with our proposed model in the small data regime. We consider the Mackey-Glass system prediction problem in Section F.3, where the size of training set is 128 and the length of each sequence sample is 2000. Table 12 shows that τ-GRU significantly outperforms the other two models while being parameter efficient for this task despite having to deal with a relatively small training set consisting of long sequences.

Table 12: Comparison of the performance, in terms of test mean squared error (MSE), of τ-GRU with S4 and LMU on the Mackey-Glass system prediction problem.

Model	MSE	# Parameters
S4 (Gu et al., 2021a)	0.0097	1.2k
LMU (Voelker et al., 2019)	0.0064	1.2k
τ-GRU	**0.0014**	**1.2k**

Discussion. The above results show that there are domains/settings where nonlinear RNNs (i.e., RNNs using a nonlinear activation function in the hidden state update equation) are beneficial, while there are other domains/settings where SSMs are favorable. For instance, SSMs can outperform τ-GRU in sequential image classification, albeit using higher number of trainable parameters (Gu et al., 2021a). Also, SSMs have the tendency to require more data for training to perform well as compared to nonlinear RNNs. SSMs need to be stacked to model non-linear dynamics, and thus they are typically deep (Wang and Xue, 2023). A fair comparison between various forms of SSMs (Gu and Dao, 2023; Hasani et al., 2022b; Smith et al., 2022; Orvieto et al., 2023) and the RNNs we consider is challenging, because those SSMs are deep whereas the RNNs we consider are shallow.