arXiv:2006.04439v4 [cs.LG] 14 Dec 2020

Liquid Time-constant Networks

Ramin Hasani,[1,3*] **Mathias Lechner,**[2*] **Alexander Amini,**[1] **Daniela Rus,**[1] **Radu Grosu**[3]
[1] Massachusetts Institute of Technology (MIT)
[2] Institute of Science and Technology Austria (IST Austria)
[3] Technische Universitaẗ Wien (TU Wien)
rhasani@mit.edu, mathias.lechner@ist.ac.at, amini@mit.edu, rus@csail.mit.edu, radu.grosu@tuwien.ac.at

Abstract

We introduce a new class of time-continuous recurrent neural network models. Instead of declaring a learning system's dynamics by implicit nonlinearities, we construct networks of linear first-order dynamical systems modulated via nonlinear interlinked gates. The resulting models represent dynamical systems with varying (i.e., *liquid*) time-constants coupled to their hidden state, with outputs being computed by numerical differential equation solvers. These neural networks exhibit stable and bounded behavior, yield superior expressivity within the family of neural ordinary differential equations, and give rise to improved performance on time-series prediction tasks. To demonstrate these properties, we first take a theoretical approach to find bounds over their dynamics, and compute their expressive power by the *trajectory length* measure in a latent trajectory space. We then conduct a series of time-series prediction experiments to manifest the approximation capability of Liquid Time-Constant Networks (LTCs) compared to classical and modern RNNs.[1]

1 Introduction

Recurrent neural networks with continuous-time hidden states determined by ordinary differential equations (ODEs), are effective algorithms for modeling time series data that are ubiquitously used in medical, industrial and business settings. The state of a neural ODE, $\mathbf{x}(t) \in \mathbb{R}^D$, is defined by the solution of this equation (Chen et al. 2018): $d\mathbf{x}(t)/dt = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)$, with a neural network f parametrized by θ. One can then compute the state using a numerical ODE solver, and train the network by performing reverse-mode automatic differentiation (Rumelhart, Hinton, and Williams 1986), either by gradient descent through the solver (Lechner et al. 2019), or by considering the solver as a black-box (Chen et al. 2018; Dupont, Doucet, and Teh 2019; Gholami, Keutzer, and Biros 2019) and apply the *adjoint method* (Pontryagin 2018). The open questions are: how expressive are neural ODEs in their current formalism, and can we improve their structure to enable richer representation learning and expressiveness?

Rather than defining the derivatives of the hidden-state directly by a neural network f, one can determine a more stable continuous-time recurrent neural network (CT-RNN) by the following equation (Funahashi and Nakamura 1993): $\frac{d\mathbf{x}(t)}{dt} = -\frac{\mathbf{x}(t)}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)$, in which the term $-\frac{\mathbf{x}(t)}{\tau}$ assists the autonomous system to reach an equilibrium state with a time-constant τ. $\mathbf{x}(t)$ is the hidden state, $\mathbf{I}(t)$ is the input, t represents time, and f is parametrized by θ.

We propose an alternative formulation: let the hidden state flow of a network be declared by a system of linear ODEs of the form: $d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t)$, and let $\mathbf{S}(t) \in \mathbb{R}^M$ represent the following nonlinearity determined by $\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$, with parameters θ and A. Then, by plugging in $\mathbf{S}$ into the hidden states equation, we get:

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A. \tag{1}$$

Eq. 1 manifests a novel time-continuous RNN instance with several features and benefits:

Liquid time-constant. A neural network f not only determines the derivative of the hidden state $\mathbf{x}(t)$, but also serves as an input-dependent varying time-constant ($\tau_{sys} = \frac{\tau}{1+\tau f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)}$) for the learning system (Time constant is a parameter characterizing the speed and the coupling sensitivity of an ODE).This property enables single elements of the hidden state to identify specialized dynamical systems for input features arriving at each time-point. We refer to these models as *liquid time-constant* recurrent neural networks (LTCs). LTCs can be implemented by an arbitrary choice of ODE solvers. In Section 2, we introduce a practical fixed-step ODE solver that simultaneously enjoys the stability of the implicit Euler and the computational efficiency of the explicit Euler methods.

Reverse-mode automatic differentiation of LTCs. LTCs realize differentiable computational graphs. Similar to neural ODEs, they can be trained by variform of gradient-based optimization algorithms. We settle to trade memory for numerical precision during a backward-pass by using a vanilla backpropagation through-time algorithm to optimize LTCs instead of an adjoint-based optimization method (Pontryagin 2018). In Section 3, we motivate this choice thoroughly.

*Authors with equal contributions

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Code and data are available at: https://github.com/raminmh/liquid_time_constant_networks

Bounded dynamics - stability. In Section 4, we show that the state and the time-constant of LTCs are bounded to a finite range. This property assures the stability of the output dynamics and is desirable when inputs to the system relentlessly increase.

Superior expressivity. In Section 5, we theoretically and quantitatively analyze the approximation capability of LTCs. We take a functional analysis approach to show the universality of LTCs. We then delve deeper into measuring their expressivity compared to other time-continuous models. We perform this by measuring the *trajectory length* of activations of networks in a latent trajectory representation. Trajectory length was introduced as a measure of expressivity of feed-forward deep neural networks (Raghu et al. 2017). We extend these criteria to the family of continuous-time recurrent models.

Time-series modeling. In Section 6, we conduct a series of eleven time-series prediction experiments and compare the performance of modern RNNs to the time-continuous models. We observe improved performance on a majority of cases achieved by LTCs.

Why this specific formulation? There are two primary justifications for the choice of this particular representation:

I) LTC model is loosely related to the computational models of neural dynamics in small species, put together with synaptic transmission mechanisms (Hasani et al. 2020). The dynamics of non-spiking neurons' potential, $\mathbf{v}(t)$, can be written as a system of linear ODEs of the form (Lapicque 1907; Koch and Segev 1998): $d\mathbf{v}/dt = -g_l\mathbf{v}(t) + \mathbf{S}(t)$, where $\mathbf{S}$ is the sum of all synaptic inputs to the cell from presynaptic sources, and g_l is a leakage conductance.

All synaptic currents to the cell can be approximated in steady-state by the following nonlinearity (Koch and Segev 1998; Wicks, Roehrig, and Rankin 1996): $\mathbf{S}(t) = f(\mathbf{v}(t), \mathbf{I}(t)), (A - \mathbf{v}(t))$, where $f(.)$ is a sigmoidal nonlinearity depending on the state of all neurons, $\mathbf{v}(t)$ which are presynaptic to the current cell, and external inputs to the cell, $I(t)$. By plugging in these two equations, we obtain an equation similar to Eq. 1. LTCs are inspired by this foundation.

II) Eq. 1 might resemble that of the famous Dynamic Causal Models (DCMs) (Friston, Harrison, and Penny 2003) with a Bilinear dynamical system approximation (Penny, Ghahramani, and Friston 2005). DCMs are formulated by taking a second-order approximation (Bilinear) of the dynamical system $d\mathbf{x}/dt = F(\mathbf{x}(t), \mathbf{I}(t), \theta)$, that would result in the following format (Friston, Harrison, and Penny 2003): $d\mathbf{x}/dt = (A + \mathbf{I}(t)B)\mathbf{x}(t) + C\mathbf{I}(t)$ with $A = \frac{dF}{d\mathbf{x}}$, $B = \frac{dF^2}{d\mathbf{x}(t)d\mathbf{I}(t)}$, $C = \frac{dF}{d\mathbf{I}(t)}$. DCM and bilinear dynamical systems have shown promise in learning to capture complex fMRI time-series signals. LTCs are introduced as variants of continuous-time (CT) models that are loosely inspired by biology, show great expressivity, stability, and performance in modeling time series.

2 LTCs forward-pass by a fused ODE solvers

Solving Eq. 1 analytically, is non-trivial due to the nonlinearity of the LTC semantics. The state of the system of ODEs, however, at any time point T, can be computed by a numerical ODE solver that simulates the system starting from a trajectory $x(0)$, to $x(T)$. An ODE solver breaks down the continuous simulation interval $[0, T]$ to a temporal discretization, $[t_0, t_1, \ldots t_n]$. As a result, a solver's step involves only the update of the neuronal states from t_i to t_{i+1}.

Algorithm 1 LTC update by fused ODE Solver

Parameters: $\theta = \{\tau^{(N\times 1)}$ = time-constant, $\gamma^{(M\times N)}$ = weights, $\gamma_r^{(N\times N)}$ = recurrent weights, $\mu^{(N\times 1)}$ = biases$\}$, $A^{(N\times 1)}$ = bias vector, L = Number of unfolding steps, Δt = step size, N = Number of neurons,
Inputs: M-dimensional Input $\mathbf{I}(t)$ of length T, $\mathbf{x}(0)$
Output: Next LTC neural state $\mathbf{x}_{t+\Delta t}$
Function: FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, Δt, θ)
$\mathbf{x}(t+\Delta t)^{(N\times T)} = \frac{\mathbf{x}(t) + \Delta t f(\mathbf{x}(t), \mathbf{I}(t), t, \theta) \odot A}{1 + \Delta t\left(1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right)}$
▷ $f(.)$, and all divisions are applied element-wise.
▷ $\odot$ is the Hadamard product.
end Function
$\mathbf{x}_{t+\Delta t} = \mathbf{x}(t)$
for $i = 1 \ldots L$ **do**
  $\mathbf{x}_{t+\Delta t}$ = FusedStep($\mathbf{x}(t)$, $\mathbf{I}(t)$, Δt, θ)
end for
return $\mathbf{x}_{t+\Delta t}$

LTCs' ODE realizes a system of stiff equations (Press et al. 2007). This type of ODE requires an exponential number of discretization steps when simulated with a Runge-Kutta (RK) based integrator. Consequently, ODE solvers based on RK, such as Dormand–Prince (default in torchdiffeq (Chen et al. 2018)), are not suitable for LTCs. Therefore, We design a new ODE solver that fuses the explicit and the implicit Euler methods (Press et al. 2007). This choice of discretization method results in achieving stability for an implicit update equation. To this end, the *Fused Solver* numerically unrolls a given dynamical system of the form $dx/dt = f(x)$ by:

$$x(t_{i+1}) = x(t_i) + \Delta t f(x(t_i), x(t_{i+1})). \quad (2)$$

In particular, we replace only the $x(t_i)$ that occur linearly in f by $x(t_{i+1})$. As a result, Eq 2 can be solved for $x(t_{i+1})$, symbolically. Applying the Fused solver to the LTC representation, and solving it for $\mathbf{x}(t + \Delta t)$, we get:

$$\mathbf{x}(t+\Delta t) = \frac{\mathbf{x}(t) \; + \; \Delta t f(\mathbf{x}(t), \mathbf{I}(t), t, \theta) A}{1 + \Delta t\big(1/\tau + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\big)}. \quad (3)$$

Eq. 3 computes one update state for an LTC network. Correspondingly, Algorithm 1 shows how to implement an LTC network, given a parameter space θ. f is assumed to have an arbitrary activation function (e.q. for a $tanh$ nonlinearity $f = \tanh(\gamma_r \mathbf{x} + \gamma \mathbf{I} + \mu)$). The computational complexity of the algorithm for an input sequence of length T is $O(L \times T)$, where L is the number of discretization steps. Intuitively, a dense version of an LTC network with N neurons, and a dense version of a long short-term memory (LSTM) (Hochreiter and Schmidhuber 1997) network with N cells, would be of the same complexity.

Algorithm 2 Training LTC by BPTT

Inputs: Dataset of traces $[I(t), y(t)]$ of length T, RNN-cell $= f(I, x)$
Parameter: Loss func $L(\theta)$, initial param θ_0, learning rate α, Output w $= W_{out}$, and bias $= b_{out}$
for $i = 1 \ldots$ number of training steps **do**
  $(I_b, y_b) =$ Sample training batch, $\quad x := x_{t_0} \sim p(x_{t_0})$
  for $j = 1 \ldots T$ **do**
    $x = f(I(t), x), \quad \hat{y}(t) = W_{out}.x + b_{out}, \quad L_{total} = \sum_{j=1}^{T} L(y_j(t), \hat{y}_j(t)), \quad \nabla L(\theta) = \frac{\partial L_{tot}}{\partial \theta}$
    $\theta = \theta - \alpha \nabla L(\theta)$
  end for
end for
return θ

Table 1: Complexity of the vanilla BPTT compared to the adjoint method, for a single layer neural network f

	Vanilla BPTT	**Adjoint**
Time	$O(L \times T \times 2)$	$O((L_f + L_b) \times T)$
Memory	$O(L \times T)$	**O(1)**
Depth	$O(L)$	$O(L_b)$
FWD acc	High	High
BWD acc	**High**	Low

Note: L = number of discretization steps, L_f = L during forward-pass. L_b = L during backward-pass. T = length of sequence, Depth = computational graph depth.

3 Training LTC networks by BPTT

Neural ODEs were suggested to be trained by a constant memory cost for each layer in a neural network f by applying the adjoint sensitivity method to perform reverse-mode automatic differentiation (Chen et al. 2018). The adjoint method, however, comes with numerical errors when running in reverse mode. This phenomenon happens because the adjoint method forgets the forward-time computational trajectories, which was repeatedly denoted by the community (Gholami, Keutzer, and Biros 2019; Zhuang et al. 2020).

On the contrary, direct backpropagation through time (BPTT) trades memory for accurate recovery of the forward-pass during the reverse mode integration (Zhuang et al. 2020). Thus, we set out to design a vanilla BPTT algorithm to maintain a highly accurate backward-pass integration through the solver. For this purpose, a given ODE solver's output (a vector of neural states), can be recursively folded to build an RNN and then apply the learning algorithm described in Algorithm 2 to train the system. Algorithm 2 uses a vanilla stochastic gradient descent (SGD). One can substitute this with a more performant variant of the SGD, such as Adam (Kingma and Ba 2014), which we use in our experiments.

Complexity. Table 1 summarizes the complexity of our vanilla BPTT algorithm compared to an adjoint method. We achieve a high degree of accuracy on both forward and backward integration trajectories, with similar computational complexity, at large memory costs.

4 Bounds on τ and neural state of LTCs

LTCs are represented by an ODE which varies its time-constant based on inputs. It is therefore important to see if

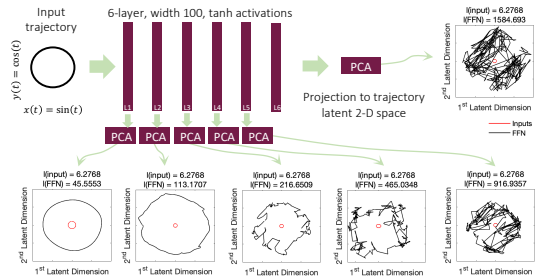

Figure 1: Trajectory's latent space becomes more complex as the input passes through hidden layers.

LTCs stay stable for unbounded arriving inputs (Hasani et al. 2019; Lechner et al. 2020b). In this section, we prove that the time-constant and the state of LTC neurons are bounded to a finite range, as described in Theorems 1 and 2, respectively.

Theorem 1. *Let x_i denote the state of a neuron i within an LTC network identified by Eq. 1, and let neuron i receive M incoming connections. Then, the time-constant of the neuron, τ_{sys_i}, is bounded to the following range:*

$$\tau_i/(1 + \tau_i W_i) \leq \tau_{sys_i} \leq \tau_i, \tag{4}$$

The proof is provided in Appendix. It is constructed based on bounded, monotonically increasing sigmoidal nonlinearity for neural network f and its replacement in the LTC network dynamics. A stable varying time-constant significantly enhances the expressivity of this form of time-continuous RNNs, as we discover more formally in Section 5.

Theorem 2. *Let x_i denote the state of a neuron i within an LTC, identified by Eq. 1, and let neuron i receive M incoming connections. Then, the hidden state of any neuron i, on a finite interval $Int \in [0, T]$, is bounded as follows:*

$$min(0, A_i^{min}) \leq x_i(t) \leq max(0, A_i^{max}), \tag{5}$$

The proof is given in Appendix. It is constructed based on the sign of the LTC's equation's compartments, and an approximation of the ODE model by an explicit Euler discretization. Theorem 2 illustrates a desired property of LTCs, namely *state stability* which guarantees that the outputs of LTCs never explode even if their inputs grow to infinity. Next we discuss the expressive power of LTCs compared to the family of time-continuous models, such as CT-RNNs and neural ordinary differential equations (Chen et al. 2018; Rubanova, Chen, and Duvenaud 2019).

5 On the expressive power of LTCs

Understanding how the structural properties of neural networks determine which functions they can compute is known as the expressivity problem. The very early attempts on measuring expressivity of neural nets include the theoretical studies based on functional analysis. They show that neural networks with three-layers can approximate any finite set of continuous mapping with any precision. This is known as the *universal approximation theorem* (Hornik, Stinchcombe, and White 1989; Funahashi 1989; Cybenko

Table 2: **Computational depth of models**

Activations	Computational Depth		
	Neural ODE	CT-RNN	**LTC**
tanh	0.56 ± 0.016	4.13 ± 2.19	9.19 ± 2.92
sigmoid	0.56 ± 0.00	5.33 ± 3.76	7.00 ± 5.36
ReLU	1.29 ± 0.10	4.31 ± 2.05	56.9 ± 9.03
Hard-tanh	0.61 ± 0.02	4.05 ± 2.17	81.01 ± 10.05

Note: # of tries = 100, input samples' $\Delta t = 0.01$, $T = 100$ sequence length. # of layers = 1, width = 100, $\sigma_w^2 = 2$, $\sigma_b^2 = 1$.

1989). Universality was extended to standard RNNs (Funahashi 1989) and even continuous-time RNNs (Funahashi and Nakamura 1993). By careful considerations, we can also show that LTCs are also universal approximators.

Theorem 3. *Let $\boldsymbol{x} \in \mathbb{R}^n$, $S \subset \mathbb{R}^n$ and $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ be an autonomous ODE with $F : S \to \mathbb{R}^n$ a C^1-mapping on S. Let D denote a compact subset of S and assume that the simulation of the system is bounded in the interval $I = [0,\ T]$. Then, for a positive ϵ, there exist an LTC network with N hidden units, n output units, and an output internal state $\boldsymbol{u}(t)$, described by Eq. 1, such that for any rollout $\{\boldsymbol{x}(t) | t \in I\}$ of the system with initial value $x(0) \in D$, and a proper network initialization,*

$$max_{t \in I} |\boldsymbol{x}(t) - \boldsymbol{u}(t)| < \epsilon \tag{6}$$

The main idea of the proof is to define an n-dimensional dynamical system and place it into a higher dimensional system. The second system is an LTC. The fundamental difference of the proof of LTC's universality to that of CT-RNNs (Funahashi and Nakamura 1993) lies in the distinction of the semantics of both systems where the LTC network contains a nonlinear input-dependent term in its time-constant module which makes parts of the proof non-trivial.

The universal approximation theorem broadly explores the expressive power of a neural network model. The theorem however, does not provide us with a foundational measure on where the separation is between different neural network architectures. Therefore, a more rigorous measure of expressivity is demanded to compare models, specifically those networks specialized in spatiotemporal data processing, such as LTCs. The advances made on defining measures for the expressivity of static deep learning models (Pascanu, Montufar, and Bengio 2013; Montufar et al. 2014; Eldan and Shamir 2016; Poole et al. 2016; Raghu et al. 2017) could presumably help measure the expressivity of time-continuous models, both theoretically and quantitatively, which we explore in the next section.

5.1 Measuring expressivity by trajectory length

A measure of expressivity has to take into account what degrees of complexity a learning system can compute, given the network's capacity (depth, width, type, and weights configuration). A unifying expressivity measure of static deep networks is the *trajectory length* introduced in (Raghu et al. 2017). In this context, one evaluates how a deep model transforms a given input trajectory (e.g., a circular 2-dimensional input) into a more complex pattern, progressively.

We can then perform principle component analysis (PCA) over the obtained network's activations. Subsequently, we measure the length of the output trajectory in a 2-dimensional latent space, to uncover its relative complexity (see Fig. 1). The trajectory length is defined as the *arc length* of a given trajectory $I(t)$, (e.g. a circle in 2D space) (Raghu et al. 2017): $l(I(t)) = \int_t \|dI(t)/dt\|\, dt$. By establishing a lower-bound for the growth of the trajectory length, one can set a barrier between networks of shallow and deep architectures, regardless of any assumptions on the network's weight configuration (Raghu et al. 2017), unlike many other measures of expressivity (Pascanu, Montufar, and Bengio 2013; Montufar et al. 2014; Serra, Tjandraatmadja, and Ramalingam 2017; Gabrié et al. 2018; Hanin and Rolnick 2018, 2019; Lee, Alvarez-Melis, and Jaakkola 2019). We set out to extend the trajectory-space analysis of static networks to time-continuous (TC) models, and to lower-bound the trajectory length to compare models' expressivity. To this end, we designed instances of Neural ODEs, CT-RNNs and LTCs with shared f. The networks were initialized by weights $\sim \mathcal{N}(0, \sigma_w^2/k)$, and biases $\sim \mathcal{N}(0, \sigma_b^2)$. We then perform forward-pass simulations by using different types of ODE solvers, for arbitrary weight profiles, while exposing the networks to a circular input trajectory $I(t) = \{I_1(t) = \sin(t), I_2(t) = \cos(t)\}$, for $t \in [0, 2\pi]$. By looking at the first two principle components (with an average variance-explained of over 80%) of hidden layers' activations, we observed consistently more complex trajectories for LTCs. Fig. 2 gives a glimpse of our empirical observations. All networks are implemented by the Dormand-Prince explicit Runge-Kutta(4,5) solver (Dormand and Prince 1980) with a variable step size. We had the following **observations**: **I)** Exponential growth of the trajectory length of Neural ODEs and CT-RNNs with Hard-tanh and ReLU activations (Fig. 2A) and unchanged shape of their latent space regardless of their weight profile. **II)** LTCs show a slower growth-rate of the trajectory length when designed by Hard-tanh and ReLU, with the compromise of realizing great levels of complexity (Fig. 2A, 2C and 2E). **III)** Apart from multi-layer time-continuous models built by Hard-tanh and ReLU activations, in all cases, we observed a longer and a more complex latent space behavior for the LTC networks (Fig. 2B to 2E). **IV)** Unlike static deep networks (Fig. 1), we witnessed that the trajectory length does not grow by depth in multi-layer continuous-time networks realized by tanh and sigmoid (Fig. 2D). **V)** conclusively, we observed that the trajectory length in TC models varies by a model's activations, weight and bias distributions variance, width and depth. We presented this more systematically in Fig. 3. **VI)** Trajectory length grows linearly with a network's width (Fig. 3B - Notice the logarithmic growth of the curves in the log-scale Y-axis). **VII)** The growth is considerably faster as the variance grows (Fig. 3C). **VIII)** Trajectory length is reluctant to the choice of ODE solver (Fig. 3A). **IX)** Activation functions diversify the complex patterns explored by the TC system, where ReLU and Hard-tanh networks demonstrate higher degrees of complexity for LTCs. A key reason is the presence of recurrent links between each layer's cells. **Definition of Computational Depth (L).** For one hidden layer of f in a time-continuous network, L is the average number of integration steps taken by the solver for each incoming input

sample. Note that for an f with n layers we define the total depth as $n \times L$. These observations have led us to formulate Lower bounds for the growth of the trajectory length of continuous-time networks.

Theorem 4. Trajectory Length growth Bounds for Neural ODEs and CT-RNNs. *Let* $dx/dt = f_{n,k}(\boldsymbol{x}(t), \boldsymbol{I}(t), \theta)$ *with* $\theta = \{W, b\}$, *represent a Neural ODE and* $\frac{d\boldsymbol{x}(t)}{dt} = -\frac{\boldsymbol{x}(t)}{\tau} + f_{n,k}(\boldsymbol{x}(t), \boldsymbol{I}(t), \theta)$ *with* $\theta = \{W, b, \tau\}$ *a CT-RNN.* f *is randomly weighted with*Hard-tanh *activations. Let* $\boldsymbol{I}(t)$ *be a 2D input trajectory, with its progressive points (i.e.* $I(t+\delta t)$*) having a perpendicular component to* $\boldsymbol{I}(t)$ *for all* δt*, with* L *= number of solver-steps. Then, by defining the projection of the first two principle components' scores of the hidden states over each other, as the 2D* latent trajectory space *of a layer d,* $z^{(d)}(\boldsymbol{I}(t)) = z^{(d)}(t)$*, for Neural ODE and CT-RNNs respectively, we have:*

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}}\right)^{d\times L} l(I(t)), \tag{7}$$

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{(\sigma_w - \sigma_b)\sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}}\right)^{d\times L} l(I(t)). \tag{8}$$

The proof is provided in Appendix. It follows similar steps as (Raghu et al. 2017) on the trajectory length bounds established for deep networks with piecewise linear activations, with careful considerations due to the continuous-time setup. The proof is constructed such that we formulate a recurrence between the norm of the hidden state gradient in layer $d+1$, $\left\|dz/dt^{(d+1)}\right\|$, in principle components domain, and the expectation of the norm of the right-hand-side of the differential equations of neural ODEs and CT-RNNs. We then roll back the recurrence to reach the inputs.

Note that to reduced the complexity of the problem, we only bounded the orthogonal components of the hidden state image $\left\|dz/dt_{\perp}^{(d+1)}\right\|$, and therefore we have the assumption on input $I(t)$, in the Theorem's statement (Raghu et al. 2017). Next, we find a lower-bound for the LTC networks.

Theorem 5. Growth Rate of LTC's Trajectory Length. *Let Eq. 1 determine an LTC with* $\theta = \{W, b, \tau, A\}$. *With the same conditions on* f *and* $I(t)$*, as in Theorem 4, we have:*

$$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}}\right)^{d\times L} \times \left(\sigma_w + \frac{\left\|z^{(d)}\right\|}{\min(\delta t, L)}\right)\right) l(I(t)). \tag{9}$$

The proof is provided in Appendix. A rough outline: we construct the recurrence between the norm of the hidden state gradients and the components of the right-hand-side of LTC separately which progressively build up the bound.

5.2 Discussion of the theoretical bounds

I) As expected, the bound for the Neural ODEs is very similar to that of an n layer static deep network with the exception of the exponential dependencies to the number of solver-steps, L. **II)** The bound for CT-RNNs suggests their shorter trajectory length compared to neural ODEs, according to the base of the exponent. This results consistently matches our experiments presented in Figs. 2 and 3. **III)** Fig. 2B and Fig. 3C show a faster-than-linear growth for LTC's trajectory length as a function of weight distribution variance. This is confirmed by LTC's lower bound shown in Eq. 9. **IV)** LTC's lower bound also depicts the linear growth of the trajectory length with the width, k, which validates the results presented in 3B. **V)** Given the computational depth of the models L in Table 2 for Hard-tanh activations, the computed lower bound for neural ODEs, CT-RNNs and LTCs justify a longer trajectory length of LTC networks in the experiments of Section 5. Next, we assess the expressive power of LTCs in a set of real-life time-series prediction tasks.

6 Experimental Evaluation

6.1 Time series predictions. We evaluated the performance of LTCs realized by the proposed Fused ODE solver against

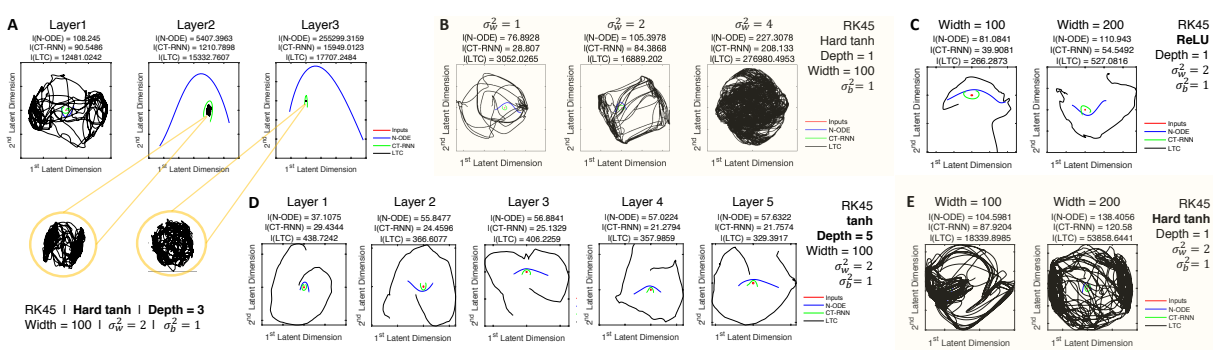

Figure 2: Trajectory length deformation A) in network layers with Hard-tanh activations, B) as a function of the weight distribution scaling factor, C) as a function of network width (ReLU), D) in network layers with logistic-sigmoid activations and E) as a function of width (Hard-tanh).

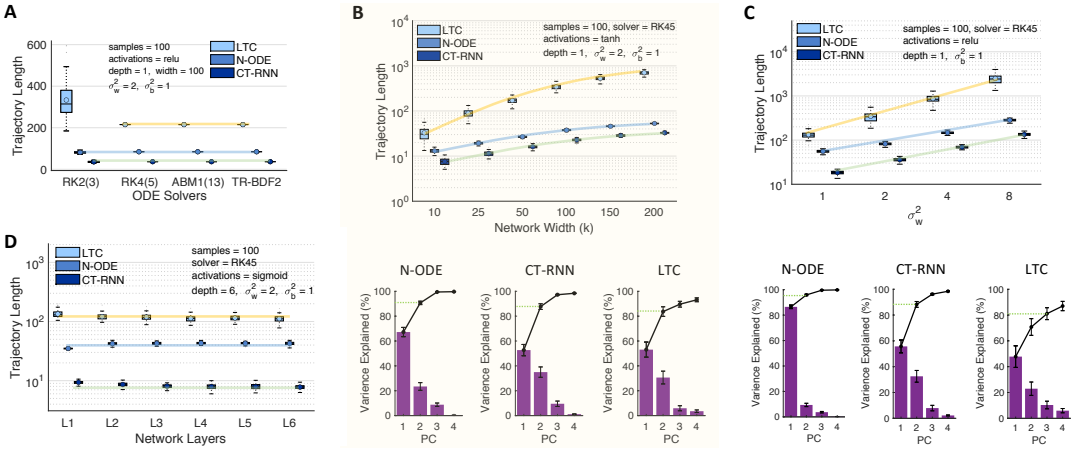

Figure 3: Dependencies of the trajectory length measure. A) trajectory length vs different solvers (variable-step solvers). RK2(3): Bogacki-Shampine Runge-Kutta (2,3) (Bogacki and Shampine 1989). RK4(5): Dormand-Prince explicit RK (4,5) (Dormand and Prince 1980). ABM1(13): Adams-Bashforth-Moulton (Shampine 1975). TR-BDF2: implicit RK solver with 1st stage trapezoidal rule and a 2nd stage backward differentiation (Hosea and Shampine 1996). B) Top: trajectory length vs network width. Bottom: Variance-explained of principle components (purple bars) and their cumulative values (solid black line). C) Trajectory length vs weights distribution variance. D) trajectory length vs layers. (More results in the supplements)

Table 3: **Time series prediction** Mean and standard deviation, n=5

Dataset	Metric	LSTM	CT-RNN	Neural ODE	CT-GRU	LTC (ours)
Gesture	(accuracy)	64.57% $\pm$ 0.59	59.01% $\pm$ 1.22	46.97% $\pm$ 3.03	68.31% $\pm$ 1.78	**69.55%** $\pm$ **1.13**
Occupancy	(accuracy)	93.18% $\pm$ 1.66	94.54% $\pm$ 0.54	90.15% $\pm$ 1.71	91.44% $\pm$ 1.67	**94.63%** $\pm$ **0.17**
Activity recognition	(accuracy)	95.85% $\pm$ 0.29	95.73% $\pm$ 0.47	**97.26%** $\pm$ 0.10	96.16% $\pm$ 0.39	95.67% $\pm$ 0.575
Sequential MNIST	(accuracy)	**98.41%** $\pm$ 0.12	96.73% $\pm$ 0.19	97.61% $\pm$ 0.14	98.27% $\pm$ 0.14	97.57% $\pm$ 0.18
Traffic	(squared error)	0.169 $\pm$ 0.004	0.224 $\pm$ 0.008	1.512 $\pm$ 0.179	0.389 $\pm$ 0.076	**0.099** $\pm$ 0.0095
Power	(squared-error)	0.628 $\pm$ 0.003	0.742 $\pm$ 0.005	1.254 $\pm$ 0.149	**0.586** $\pm$ 0.003	0.642 $\pm$ 0.021
Ozone	(F1-score)	0.284 $\pm$ 0.025	0.236 $\pm$ 0.011	0.168 $\pm$ 0.006	0.260 $\pm$ 0.024	**0.302** $\pm$ 0.0155

Table 4: Person activity, 1st setting - n=5

Algorithm	**Accuracy**
LSTM	83.59%$\pm$ 0.40
CT-RNN	81.54%$\pm$ 0.33
Latent ODE	76.48%$\pm$ 0.56
CT-GRU	85.27%$\pm$ 0.39
LTC (ours)	**85.48**%$\pm$ **0.40**

the state-of-the-art discretized RNNs, LSTMs (Hochreiter and Schmidhuber 1997), CT-RNNs (ODE-RNNs) (Funahashi and Nakamura 1993; Rubanova, Chen, and Duvenaud 2019), continuous-time gated recurrent units (CT-GRUs) (Mozer, Kazakov, and Lindsey 2017), and Neural ODEs constructed by a 4th order Runge-Kutta solver as suggested in (Chen et al. 2018), in a series of diverse real-life supervised learning tasks. The results are summarized in Table 3. The experimental setup are provided in Appendix. We observed between 5% to 70% performance improvement achieved by the LTCs compared to other RNN models in four out of seven experiments and comparable performance in the other three (see Table 3).

6.2 Person activity dataset. We use the "Human Activity" dataset described in (Rubanova, Chen, and Duvenaud 2019) in two distinct frameworks. The dataset consists of 6554 sequences of activity of humans (e.g. lying, walking, sitting), with a period of 211 ms. we designed two experimental frameworks to evaluate models' performance. In the *1st Setting*, the baselines are the models described before, and the input representations are unchanged (details in Appendix). LTCs outperform all models and in particular CT-RNNs and neural ODEs with a large margin as shown in Table 4. Note that the CT-RNN architecture is equivalent to the ODE-RNN described in (Rubanova, Chen, and Duvenaud 2019), with the difference of having a state damping factor τ.

In the *2nd Setting*, we carefully set up the experiment to match the modifications made by (Rubanova, Chen, and Duvenaud 2019) (See supplements), to obtain a fair comparison between LTCs and a more diverse set of RNN variants discussed in (Rubanova, Chen, and Duvenaud 2019). LTCs show superior performance with a high margin compared to other models. The results are summarized in Table 5).

6.3 Half-Cheetah kinematic modeling. We intended to evaluate how well continuous-time models can capture physical dynamics. To perform this, we collected 25 rollouts of a pre-trained controller for the HalfCheetah-v2 gym environment (Brockman et al. 2016), generated by the Mu-

Table 5: Person activity, 2nd setting

Algorithm	Accuracy
RNN Δ_t *	0.797± 0.003
RNN-Decay*	0.800± 0.010
RNN GRU-D*	0.806± 0.007
RNN-VAE*	0.343± 0.040
Latent ODE (D enc.)*	0.835± 0.010
ODE-RNN *	0.829 ± 0.016
Latent ODE(C enc.)*	0.846 ± 0.013
LTC (ours)	**0.882** ± **0.005**

Note: Accuracy for algorithms indicated by $*$, are taken directly from (Rubanova, Chen, and Duvenaud 2019). RNN Δ_t = classic RNN + input delays (Rubanova, Chen, and Duvenaud 2019). RNN-Decay = RNN with exponential decay on the hidden states (Mozer, Kazakov, and Lindsey 2017). GRU-D = gated recurrent unit + exponential decay + input imputation (Che et al. 2018). D-enc. = RNN encoder (Rubanova, Chen, and Duvenaud 2019). C-enc = ODE encoder (Rubanova, Chen, and Duvenaud 2019). n=5

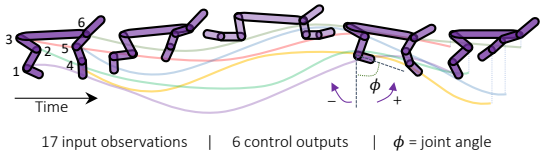

Figure 4: Half-cheetah physics simulation

JoCo physics engine (Todorov, Erez, and Tassa 2012). The task is then to fit the observation space time-series in an autoregressive fashion (Fig. 4). To increase the difficulty, we overwrite 5% of the actions by random actions. The test results are presented in Table 6, and root for the superiority of the performance of LTCs compared to other models.

7 Related Works

Time-continuous models. TC networks have become unprecedentedly popular. This is due to the manifestation of several benefits such as adaptive computations, better continuous time-series modeling, memory, and parameter efficiency (Chen et al. 2018). A large number of alternative approaches have tried to improve and stabilize the adjoint method (Gholami, Keutzer, and Biros 2019), use neural ODEs in specific contexts (Rubanova, Chen, and Duvenaud 2019; Lechner et al. 2019) and to characterize them better (Dupont, Doucet, and Teh 2019; Durkan et al. 2019; Jia and Benson 2019; Hanshu et al. 2020; Holl, Koltun, and Thuerey 2020; Quaglino et al. 2020). In this work, we investigated the expressive power of neural ODEs and proposed a new ODE model to improve their expressivity and performance.

Measures of expressivity. A large body of modern works tried to find answers to the questions such as why deeper networks and particular architectures perform well, and where is the boundary between the approximation capability of shallow networks and deep networks? In this context, (Montufar et al. 2014) and (Pascanu, Montufar, and Bengio 2013) suggested to count the number of linear regions of neural networks as a measure of expressivity, (Eldan and Shamir 2016) showed that there exists a class of radial functions that smaller networks fail to produce, and (Poole et al. 2016) studied the exponential expressivity of neural networks by transient chaos.

These methods are compelling; however, they are bound to particular weight configurations of a given network in order to lower-bound expressivity similar to (Serra, Tjandraatmadja, and Ramalingam 2017; Gabrié et al. 2018; Hanin and Rolnick 2018, 2019; Lee, Alvarez-Melis, and Jaakkola 2019). (Raghu et al. 2017) introduced an interrelated concept which quantifies the expressiveness of a given static network by trajectory length. We extended their expressivity analysis to time-continuous networks and provided lower-bound for the growth of the trajectory length, proclaiming the superior approximation capabilities of LTCs.

Table 6: Sequence modeling. Half-Cheetah dynamics n=5

Algorithm	MSE
LSTM	2.500± 0.140
CT-RNN	2.838± 0.112
Neural ODE	3.805 ± 0.313
CT-GRU	3.014± 0.134
LTC (ours)	**2.308**± **0.015**

8 Conclusions, Scope and Limitations

We investigated the use of a novel class of time-continuous neural network models obtained by a combination of linear ODE neurons and special nonlinear weight configurations. We showed that they could be implemented effectively by arbitrary variable and fixed step ODE solvers, and be trained by backpropagation through time. We demonstrated their bounded and stable dynamics, superior expressivity, and superseding performance in supervised learning time-series prediction tasks, compared to standard and modern deep learning models.

Long-term dependencies. Similar to many variants of time-continuous models, LTCs express the vanishing gradient phenomenon (Pascanu, Mikolov, and Bengio 2013; Lechner and Hasani 2020), when trained by gradient descent. Although the model shows promise on a variety of time-series prediction tasks, they would not be the obvious choice for learning long-term dependencies in their current format.

Choice of ODE solver. Performance of time-continuous models is heavily tided to their numerical implementation approach (Hasani 2020). While LTCs perform well with advanced variable-step solvers and the Fused fixed-step solver introduced here, their performance is majorly influenced when off-the-shelf explicit Euler methods are used.

Time and Memory. Neural ODEs are remarkably fast compared to more sophisticated models such as LTCs. Nonetheless, they lack expressivity. Our proposed model, in their current format, significantly enhances the expressive power of TC models at the expense of elevated time and memory complexity which must be investigated in the future.

Causality. Models described by time-continuous differential equation semantics inherently possess causal structures (Schölkopf 2019), especially models that are equipped with recurrent mechanisms to map past experiences to next-step predictions. Studying causality of performant recurrent models such as LTCs would be an exciting future research direction to take, as their semantics resemble *dynamic causal models* (Friston, Harrison, and Penny 2003) with a *bilinear dynamical system* approximation (Penny, Ghahramani, and Friston 2005). Accordingly, a natural application domain would be the control of robots in continuous-time observation and action spaces where causal structures such as LTCs can help improve reasoning (Lechner et al. 2020a).

Acknowledgments

R.H. and D.R. are partially supported by Boeing. R.H. and R.G. were partially supported by the Horizon-2020 ECSEL Project grant No. 783163 (iDev40). M.L. was supported in part by the Austrian Science Fund (FWF) under grant Z211-N23 (Wittgenstein Award). A.A. is supported by the National Science Foundation (NSF) Graduate Research Fellowship Program. This research work is partially drawn from the PhD dissertation of R.H.

References

Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; and Reyes-Ortiz, J. L. 2013. A public domain dataset for human activity recognition using smartphones. In *Esann*.

Bogacki, P.; and Shampine, L. F. 1989. A 3 (2) pair of Runge-Kutta formulas. *Applied Mathematics Letters* 2(4): 321–325.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* .

Candanedo, L. M.; and Feldheim, V. 2016. Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. *Energy and Buildings* 112: 28–39.

Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; and Liu, Y. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* 8(1): 1–12.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. K. 2018. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, 6571–6583.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2(4): 303–314.

Dormand, J. R.; and Prince, P. J. 1980. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics* 6(1): 19–26.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository. URL http://archive.ics.uci.edu/ml.

Dupont, E.; Doucet, A.; and Teh, Y. W. 2019. Augmented neural odes. In *Advances in Neural Information Processing Systems*, 3134–3144.

Durkan, C.; Bekasov, A.; Murray, I.; and Papamakarios, G. 2019. Neural spline flows. In *Advances in Neural Information Processing Systems*, 7509–7520.

Eldan, R.; and Shamir, O. 2016. The power of depth for feedforward neural networks. In *Conference on learning theory*, 907–940.

Friston, K. J.; Harrison, L.; and Penny, W. 2003. Dynamic causal modelling. *Neuroimage* 19(4): 1273–1302.

Funahashi, K.-I. 1989. On the approximate realization of continuous mappings by neural networks. *Neural networks* 2(3): 183–192.

Funahashi, K.-i.; and Nakamura, Y. 1993. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks* 6(6): 801–806.

Gabrié, M.; Manoel, A.; Luneau, C.; Macris, N.; Krzakala, F.; Zdeborová, L.; et al. 2018. Entropy and mutual information in models of deep neural networks. In *Advances in Neural Information Processing Systems*, 1821–1831.

Gholami, A.; Keutzer, K.; and Biros, G. 2019. Anode: Unconditionally accurate memory-efficient gradients for neural odes. *arXiv preprint arXiv:1902.10298* .

Hanin, B.; and Rolnick, D. 2018. How to start training: The effect of initialization and architecture. In *Advances in Neural Information Processing Systems*, 571–581.

Hanin, B.; and Rolnick, D. 2019. Complexity of linear regions in deep networks. *arXiv preprint arXiv:1901.09021* .

Hanshu, Y.; Jiawei, D.; Vincent, T.; and Jiashi, F. 2020. On Robustness of Neural Ordinary Differential Equations. In *International Conference on Learning Representations*.

Hasani, R. 2020. *Interpretable Recurrent Neural Networks in Continuous-time Control Environments*. PhD dissertation, Technische Universität Wien.

Hasani, R.; Amini, A.; Lechner, M.; Naser, F.; Grosu, R.; and Rus, D. 2019. Response characterization for auditing cell dynamics in long short-term memory networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

Hasani, R.; Lechner, M.; Amini, A.; Rus, D.; and Grosu, R. 2020. The natural lottery ticket winner: Reinforcement learning with ordinary neural circuits. In *Proceedings of the 2020 International Conference on Machine Learning. JMLR. org*.

Hirsch, M. W.; and Smale, S. 1973. *Differential equations, dynamical systems and linear algebra*. Academic Press college division.

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

Holl, P.; Koltun, V.; and Thuerey, N. 2020. Learning to Control PDEs with Differentiable Physics. *arXiv preprint arXiv:2001.07457* .

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5): 359–366.

Hosea, M.; and Shampine, L. 1996. Analysis and implementation of TR-BDF2. *Applied Numerical Mathematics* 20(1-2): 21–37.

Jia, J.; and Benson, A. R. 2019. Neural jump stochastic differential equations. In *Advances in Neural Information Processing Systems*, 9843–9854.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Koch, C.; and Segev, K. 1998. *Methods in Neuronal Modeling - From Ions to Networks*. MIT press, second edition.

Lapicque, L. 1907. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej* 9: 620–635.

Lechner, M.; and Hasani, R. 2020. Learning Long-Term Dependencies in Irregularly-Sampled Time Series. *arXiv preprint arXiv:2006.04418* .

Lechner, M.; Hasani, R.; Amini, A.; Henzinger, T. A.; Rus, D.; and Grosu, R. 2020a. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence* 2(10): 642–652.

Lechner, M.; Hasani, R.; Rus, D.; and Grosu, R. 2020b. Gershgorin Loss Stabilizes the Recurrent Neural Network Compartment of an End-to-end Robot Learning Scheme. In *2020 International Conference on Robotics and Automation (ICRA)*. IEEE.

Lechner, M.; Hasani, R.; Zimmer, M.; Henzinger, T. A.; and Grosu, R. 2019. Designing worm-inspired neural networks for interpretable robotic control. In *2019 International Conference on Robotics and Automation (ICRA)*, 87–94. IEEE.

Lee, G.-H.; Alvarez-Melis, D.; and Jaakkola, T. S. 2019. Towards robust, locally linear deep networks. *arXiv preprint arXiv:1907.03207* .

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924–2932.

Mozer, M. C.; Kazakov, D.; and Lindsey, R. V. 2017. Discrete Event, Continuous Time RNNs. *arXiv preprint arXiv:1710.04110* .

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

Pascanu, R.; Montufar, G.; and Bengio, Y. 2013. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098* .

Penny, W.; Ghahramani, Z.; and Friston, K. 2005. Bilinear dynamical systems. *Philosophical Transactions of the Royal Society B: Biological Sciences* 360(1457): 983–993.

Pontryagin, L. S. 2018. *Mathematical theory of optimal processes*. Routledge.

Poole, B.; Lahiri, S.; Raghu, M.; Sohl-Dickstein, J.; and Ganguli, S. 2016. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, 3360–3368.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 2007. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 3 edition.

Quaglino, A.; Gallieri, M.; Masci, J.; and Koutník, J. 2020. SNODE: Spectral Discretization of Neural ODEs for System Identification. In *International Conference on Learning Representations*.

Raghu, M.; Poole, B.; Kleinberg, J.; Ganguli, S.; and Dickstein, J. S. 2017. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2847–2854. JMLR.

Rubanova, Y.; Chen, R. T.; and Duvenaud, D. 2019. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907* .

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *nature* 323(6088): 533–536.

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

Schölkopf, B. 2019. Causality for Machine Learning. *arXiv preprint arXiv:1911.10500* .

Serra, T.; Tjandraatmadja, C.; and Ramalingam, S. 2017. Bounding and counting linear regions of deep neural networks. *arXiv preprint arXiv:1711.02114* .

Shampine, L. F. 1975. Computer solution of ordinary differential equations. *The Initial Value Problem* .

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Tsagris, M.; Beneki, C.; and Hassani, H. 2014. On the folded normal distribution. *Mathematics* 2(1): 12–28.

Wagner, P. K.; Peres, S. M.; Madeo, R. C. B.; de Moraes Lima, C. A.; and de Almeida Freitas, F. 2014. Gesture unit segmentation using spatial-temporal information and machine learning. In *The Twenty-Seventh International Flairs Conference*.

Wicks, S. R.; Roehrig, C. J.; and Rankin, C. H. 1996. A dynamic network simulation of the nematode tap withdrawal circuit: predictions concerning synaptic function using behavioral criteria. *Journal of Neuroscience* 16(12): 4017–4031.

Zhang, K.; and Fan, W. 2008. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems* 14(3): 299–326.

Zhuang, J.; Dvornek, N.; Li, X.; Tatikonda, S.; Papademetris, X.; and Duncan, J. 2020. Adaptive Checkpoint Adjoint Method for Gradient Estimation in Neural ODE. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR 119.

Supplementary Materials

S1 Proof of Theorem 1

Proof. Assuming the neural network f in Eq. 1, possesses a bounded sigmoidal nonlinearity which is a monotonically increasing between 0 and 1. Then for each neuron i, we have:

$$0 < f(x_j(t), \gamma_{ij}, \mu_{ij}) < 1 \tag{S1}$$

By replacing the upper-bound of f in Eq. 1, and assuming a scaling weight matrix $W_i^{M\times 1}$, for each neuron i in f, we get:

$$\frac{dx_i}{dt} = -\Big[\frac{1}{\tau_i} + W_i\Big]x_i(t) + W_i A_i. \tag{S2}$$

The Equation simplifies to a linear ODE, of the form:

$$\frac{dx_i}{dt} = -\underbrace{\Big[\frac{1}{\tau_i} + W_i\Big]}_{\text{a}} x_i - \underbrace{W_i A_i}_{\text{b}}, \quad \rightarrow \quad \frac{dx_i}{dt} = -ax_i + b, \tag{S3}$$

with a solution of the form:

$$x_i(t) = k_1 e^{-at} + \frac{b}{a}. \tag{S4}$$

From this solution, we derive the lower bound of the system's time constant, $\tau_{sys_i}^{min}$:

$$\tau_{sys_i}^{min} = \frac{1}{a} = \frac{1}{1\ +\ \tau_i W_i}. \tag{S5}$$

By replacing the lower-bound of f in Eq. 1, the equation simplifies to an autonomous linear ODE as follows:

$$\frac{dx_i}{dt} = -\frac{1}{\tau_i}x_i(t). \tag{S6}$$

which gives us the upper-bound of the system's time-constant, $\tau_{sys_i}^{max}$:

$$\tau_{sys_i}^{max} = \tau_i \tag{S7}$$

□

S2 Proof of Theorem 2

Proof. Let us insert $M = max\{0, A_i^{max}\}$ as the neural state of neuron i, $x_i(t)$ into Equation 1:

$$\frac{dx_i}{dt} = -\Big[\frac{1}{\tau} + f(\mathbf{x}_j(t), t, \theta)\Big]M + f(\mathbf{x}_j(t), t, \theta)A_i. \tag{S8}$$

Now by expanding the brackets, we get

$$\frac{dx_i}{dt} = \underbrace{-\frac{1}{\tau}M}_{\leq 0} + \underbrace{-f(\mathbf{x}_j(t), t, \theta)M + f(\mathbf{x}_j(t), t, \theta)A_i}_{\leq 0}. \tag{S9}$$

The right-hand side of Eq. S9, is negative based on the conditions on M, positive weights, and the fact that $f(x_j)$ is also positive, Therefore, the left-hand-side must also be negative and if we perform an approximation on the derivative term, the following holds:

$$\frac{dx_i}{dt} \leq 0, \quad \frac{dx_i}{dt} \approx \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \leq 0, \tag{S10}$$

By substituting $x_i(t)$ with M, we get:

$$\frac{x(t+\Delta t) - M}{\Delta t} \leq 0 \ \rightarrow \ x(t+\Delta t) \leq M \tag{S11}$$

and therefore:

$$x_i(t) \leq max(0, A_i^{max}). \tag{S12}$$

Now if we replace $x_{(i)}$ by $m = min\{0, A_i^{min}\}$, and follow a similar methodology used for the upper bound, we can derive:

$$\frac{x(t+\Delta t) - m}{\Delta t} \leq 0 \ \rightarrow \ x(t+\Delta t) \leq m, \tag{S13}$$

and therefore:

$$x_i(t) \geq min(0, A_i^{min}). \tag{S14}$$

□

S3 Proof of Theorem 3

We prove that any given n-dimensional dynamical system for a finite simulation time can be approximated by the internal and output states of an LTC, with n-outputs, some hidden nodes, and a proper initial condition. We base our proof on the fundamental universal approximation theorem (Hornik, Stinchcombe, and White 1989) on feedforward neural networks (Funahashi 1989; Cybenko 1989; Hornik, Stinchcombe, and White 1989), recurrent neural networks (RNN) (Funahashi 1989; Schäfer and Zimmermann 2006) and continuous-time RNNs (Funahashi and Nakamura 1993). The fundamental difference of the proof of the universal approximation capability of LTCs compared to that of CT-RNNs lies in the distinction of the semantics of both ODE systems. LTC networks contain a nonlinear input-dependent term in their time-constant module, represented in Eq. 1, which alters the entire dynamical system from that of CT-RNNs. Therefore, careful considerations have to be adjusted while taking the same approach to that of CT-RNNs for proving their universality. We first revisit preliminary statements that are used in the proof and are about basic topics on dynamical systems.

THEOREM (The fundamental approximation theorem) (Funahashi 1989). Let $\mathbf{x} = (x_1, ..., x_n)$ be an n-dimensional Euclidean space $\mathbb{R}^n$. *Let $f(x)$ be a sigmoidal function (a non-constant, monotonically increasing and bounded continous function in $\mathbb{R}$). Let K be a compact subset of $\mathbb{R}^n$, and $f(x_1, ..., x_n)$ be a continuous function on K. Then, for an arbitrary $\epsilon > 0$, there exist an integer N, real constants c_i, $\theta_i (i = 1, ..., N)$ and $w_{ij} (i = 1, ..., N; j = 1, ..., n)$, such that*

$$\max_{x \in K} |g(x_1, ..., x_n) - \sum_{i=1}^{N} c_i f(\sum_{j=1}^{n} w_{ij} x_j - \theta_i)| < \epsilon \tag{S15}$$

holds.

This theorem illustrates that three-layer feedforward neural networks (Input-hidden layer-output), can approximate any continuous mapping $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ on a compact set.

THEOREM (Approximation of dynamical systems by continuous time recurrent neural networks) (Funahashi and Nakamura 1993). *Let $D \subset \mathbb{R}^n$ and $F : D \rightarrow \mathbb{R}^n$ be an autonomous ordinary differential equation and C^1-mapping, and let $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$ determine a dynamical system on D. Let K denote a compact subset of D and we consider the trajectories of the system on the interval $I = [0, \ T]$. Then, for an arbitrary positive ϵ, there exist an integer N and a recurrent neural network with N hidden units, n output units, and an output internal state $\boldsymbol{u}(t) = (U_1(t), ..., U_n(t))$, expressed as:*

$$\frac{du_i(t)}{dt} = -\frac{u_i(t)}{\tau_i} + \sum_{j=1}^{m} w_{ij} f(u_j(t)) + I_i(t), \tag{S16}$$

where τ_i is the time-constant, w_{ij} are the weights, $I_i(t)$ is the input, and f is a C^1-sigmoid function ($f(x) = 1/(1+exp(-x))$, such that for any trajectory $\{x(t); t \in I\}$ of the system with initial value $x(0) \in K$, and a proper initial condition of the network the statement below holds:

$$\max_{t \in I} |\boldsymbol{x}(t) - \boldsymbol{u}(t)| < \epsilon.$$

The theorem was proved for the case where the time-constants, τ, were kept constant for all hidden states, and the RNN was without inputs ($I_i(t) = 0$) (Funahashi and Nakamura 1993).

We now restate the necessary concepts from dynamical systems to be used in the proof. Where necessary, we adopt modifications and extensions to the Lemmas, for proving Theorem 1.

Lipschitz. The mapping $F : S \rightarrow \mathbb{R}^n$, where S is an open subset of $\mathbb{R}^n$, is called Lipschitz on S if there exist a constant L (Lipschitz constant), such that:

$$|F(x) - F(y)| \leq L|x - y|, \quad \forall x, y \in S. \tag{S17}$$

Locally Lipschitz. If every point of S has neighborhood S_0 in S, such that the restriction $F \mid S_0$ is Lipschitz, then F is locally Lipschitz.

Lemma 1. *Let a mapping $F : S \rightarrow \mathbb{R}^n$ be C^1. Then F is locally Lipschitz. Also, if $D \subset S$ is compact, then the restriction $F \mid D$ is Lipschitz. (Proof in (Hirsch and Smale 1973), chapter 8, section 3).*

Lemma 2. *Let $F : S \to \mathbb{R}^n$ be a C^1-mapping and $x_0 \in S$. There exists a positive a and a unique solution $x : (-a, a) \to S$ of the differential equation*

$$\dot{x} = F(x), \tag{S18}$$

which satisfies the initial condition $x(0) = x_0$. (Proof in (Hirsch and Smale 1973), chapter 8, section 2, Theorem 1.)

Lemma 3. *Let S be an open subset of $\mathbb{R}^n$ and $F : S \to \mathbb{R}^n$ be a C^1-mapping. On a maximal interval $J = (\alpha, \beta) \subset \mathbb{R}$, let x(t) be a solution. Then for any compact subset $D \subset S$, there exists some $t \in (\alpha, \beta)$, for which $x(t) \notin D$. (Proof in (Hirsch and Smale 1973), Chapter 8, section 5, Theorem).*

Lemma 4. *For an $F : \mathbb{R}^n \to \mathbb{R}^n$ which is a bound C^1-mapping, the differential equation*

$$\dot{x} = -\frac{x}{\tau} + F(x), \tag{S19}$$

where $\tau > 0$ has a unique solution on $[0, \infty)$. (Proof in (Funahashi and Nakamura 1993), Section 4, Lemma 4).

Lemma 5. *For an $F : \mathbb{R}^n \to \mathbb{R}^{+n}$ which is a bounded C^1-mapping, the differential equation*

$$\dot{x} = -(1/\tau + F(x))x + AF(x), \tag{S20}$$

in which τ is a positive constant, and A is constant coefficients bound to a range $[-\alpha, \beta]$ for $0 < \alpha < +\infty$, and $0 \leq \beta < +\infty$, has a unique solution on $[0, \infty)$.

Proof. Based on the assumptions, we can take a positive M, such that

$$0 \leq F_i(x) \leq M (\forall i = 1, ..., n) \tag{S21}$$

by looking at the solutions of the following differential equation:

$$\dot{x} = -(1/\tau + M)x + AM, \tag{S22}$$

we can show that

$$min\{|x_i(0)|, \frac{\tau(AM)}{1 + \tau M}\} \leq x_i(t) \leq max\{|x_i(0)|, \frac{\tau(AM)}{1 + \tau M}\}, \tag{S23}$$

if we set the output of the max to C_{max_i} and the output of the min to C_{min_i} and also set $C_1 = min\{C_{min_i}\}$ and $C_2 = max\{C_{max_i}\}$, then the solution $x(t)$ satisfies

$$\sqrt{n}C_1 \leq x(t) \leq \sqrt{n}C_2. \tag{S24}$$

Based on Lemma 2 and Lemma 3 a unique solution exists on the interval $[0, +\infty)$. □

Lemma 5 demonstrates that an LTC network defined by Eq. S20, has a unique solution on $[0, \infty)$, since the output function is bounded and is a C^1 mapping.

Lemma 6. *Let two continuous mapping $F, \tilde{F} : S \to \mathbb{R}^n$ be Lipschitz, and L be a Lipschitz constant of F. if $\forall x \in S$,*

$$|F(\boldsymbol{x}) - \tilde{F}(\boldsymbol{x})| < \epsilon, \tag{S25}$$

holds, if $\boldsymbol{x}(t)$ and $\boldsymbol{y}(t)$ are solutions to

$$\dot{\boldsymbol{x}} = F(\boldsymbol{x}), \tag{S26}$$

$$\dot{\boldsymbol{y}} = \tilde{F}(\boldsymbol{x}), \tag{S27}$$

on some interval J, such that $x(t_0) = y(t_0)$, then

$$|\boldsymbol{x}(t) - \boldsymbol{y}(t)| \leq \frac{\epsilon}{L}(e^{L|t-t_0|} - 1). \tag{S28}$$

(Proof in (Hirsch and Smale 1973), chapter 15, section 1, Theorem 3).

S3.1 Proof of the Theorem:

Proof. Using the above definitions and lemmas, we prove that LTCs are universal approximators.

Part 1. We choose an η which is in range $(0, min\{\epsilon, \lambda\})$, for $\epsilon > 0$, and λ the distance between $\tilde{D}$ and boundary δS of S. D_η is set:

$$D_\eta = \{\mathbf{x} \in \mathbb{R}^n; \exists z \in \tilde{D}, |\mathbf{x} - \mathbf{z}| \leq \eta\}. \tag{S29}$$

D_η stands for a compact subset of S, because $\tilde{D}$ is compact. Thus, F is Lipschitz on D_η by Lemma 1. Let L_F be the Lipschitz constant of $F|D_\eta$, then, we can choose an $\epsilon_l > 0$, such that

$$\epsilon_l < \frac{\eta L_F}{2(e^{L_F T - 1)}}. \tag{S30}$$

Based on the universal approximation theorem, there is an integer N, and an $n \times N$ matrix A, and an $N \times n$ matrix C and an N-dimensional vector μ such that

$$max|F(\mathbf{x}) - Af(\gamma\mathbf{x} + \mu)| < \frac{\epsilon_l}{2}. \tag{S31}$$

We define a C^1-mapping $\tilde{F} : \mathbb{R}^n \to \mathbb{R}^n$ as:

$$\tilde{F}(\mathbf{x}) = -(1/\tau + W_l f(\gamma\mathbf{x} + \mu))\mathbf{x} + W_l f(\gamma\mathbf{x} + \mu)A, \tag{S32}$$

with parameters matching that of Eq. 1 with $W_l = W$.

We set system's time-constant, τ_{sys} as:

$$\tau_{sys} = \frac{1}{\tau/1 + \tau W_l f(\gamma x + \mu)}. \tag{S33}$$

We chose a large τ_{sys}, conditioned with the following:

$$(a)\ \ \forall x \in D_\eta;\ \ |\frac{x}{\tau_{sys}}| < \frac{\epsilon_l}{2} \tag{S34}$$

$$(b)\ \ |\frac{\mu}{\tau_{sys}}| < \frac{\eta L_{\tilde{G}}}{2(e^{L_{\tilde{G}}T} - 1)} \text{ and } |\frac{1}{\tau_{sys}}| < \frac{L_{\tilde{G}}}{2}, \tag{S35}$$

where $L_{\tilde{G}}/2$ is a lipschitz constant for the mapping $W_l f : \mathbb{R}^{n+N} \to \mathbb{R}^{n+N}$ which we will determine later. To satisfy conditions (a) and (b), $\tau W_l << 1$ should hold true.

Then by Eq. S31 and S32, we can prove:

$$\underset{x \in D_\eta}{max}|F(\mathbf{x}) - \tilde{F}(\mathbf{x})| < \epsilon_l \tag{S36}$$

Let's set $\mathbf{x}(t)$ and $\tilde{\mathbf{x}}t)$ with initial state $x(0) = \tilde{x}(0) = x_0 \in D$, as the solutions of equations below:

$$\dot{\mathbf{x}} = F(\mathbf{x}), \tag{S37}$$

$$\dot{\tilde{\mathbf{x}}} = \tilde{F}(\mathbf{x}). \tag{S38}$$

Based on Lemma 6 for any $t \in I$,

$$|\mathbf{x}(t) - \tilde{\mathbf{x}}(t)| \leq \frac{\epsilon_l}{L_F}(e^{L_F t} - 1) \tag{S39}$$

$$\leq \frac{\epsilon_l}{L_F}(e^{L_F T} - 1). \tag{S40}$$

Thus, based on the conditions on ϵ,

$$\underset{t \in I}{max}|\mathbf{x}(t) - \tilde{\mathbf{x}}(t)| < \frac{\eta}{2}. \tag{S41}$$

Part 2. Let's Consider the following dynamical system defined by $\tilde{F}$ in Part 1:

$$\dot{\tilde{\mathbf{x}}} = -\frac{1}{\tau_{sys}}\tilde{\mathbf{x}} + W_l f(\gamma\tilde{\mathbf{x}} + \mu)A. \tag{S42}$$

Suppose we set $\tilde{\mathbf{y}} = \gamma\tilde{\mathbf{x}} + \mu$; then:

$$\dot{\tilde{\mathbf{y}}} = \gamma\dot{\tilde{\mathbf{x}}} = -\frac{1}{\tau_{sys}}\tilde{\mathbf{y}} + Ef(\tilde{\mathbf{y}}) + \frac{\mu}{\tau_{sys}}, \tag{S43}$$

where $E = \gamma W_l A$, an $N \times N$ matrix. We define

$$\tilde{\mathbf{z}} = (\tilde{x}_1, ..., \tilde{x}_n, \tilde{y}_1, ..., \tilde{y}_n), \tag{S44}$$

and we set a mapping $\tilde{G} : \mathbb{R}^{n+N} \to \mathbb{R}^{n+N}$ as:

$$\tilde{G}(\tilde{\mathbf{z}}) = -\frac{1}{\tau_{sys}}\tilde{\mathbf{z}} + Wf(\tilde{\mathbf{z}}) + \frac{\mu_1}{\tau_{sys}}, \tag{S45}$$

where;

$$W^{(n+N)\times(n+N)} = \begin{pmatrix} 0 & A \\ 0 & E \end{pmatrix}, \tag{S46}$$

$$\mu_1^{n+N} = \begin{pmatrix} 0 \\ \mu \end{pmatrix}. \tag{S47}$$

Now using Lemma 2, we can show that solutions of the following dynamical system:

$$\dot{\tilde{\mathbf{z}}} = \tilde{G}(\tilde{\mathbf{z}}), \quad \tilde{y}(0) = \gamma\tilde{x}(0) + \mu, \tag{S48}$$

are equivalent to the solutions of the Eq. S42.

Let's define a new dynamical system $G : \mathbb{R}^{n+N} \to \mathbb{R}^{n+N}$ as follows:

$$G(\mathbf{z}) = -\frac{1}{\tau_{sys}}\mathbf{z} + Wf(\mathbf{z}), \tag{S49}$$

where $\mathbf{z} = (x_1, ..., x_n, y_1, ..., y_n)$. Then the dynamical system below

$$\dot{\mathbf{z}} = -\frac{1}{\tau_{sys}}\mathbf{z} + Wf(\mathbf{z}), \tag{S50}$$

can be realized by an LTC, if we set $\mathbf{h}(t) = (h_1(t), ..., h_N(t))$ as the hidden states, and $\mathbf{u}(t) = (U_1(t), ..., U_n(t))$ as the output states of the system. Since $\tilde{G}$ and G are both C^1-mapping and $f'(\mathbf{x})$ is bound, therefore, the mapping $\tilde{\mathbf{z}} \mapsto Wf(\tilde{\mathbf{z}})$ is Lipschitz on $\mathbb{R}^{n+N}$, with a Lipschitz constant $L_{\tilde{G}}/2$. As $L_{\tilde{G}}/2$ is lipschitz constant for $-\tilde{z}/\tau_{sys}$ by condition (b) on τ_{sys}, $L_{\tilde{G}}$ is a Lipschitz constant of $\tilde{G}$.

From Eq. S45, Eq. S49, and condition (b) of τ_{sys}, we can derive the following:

$$|\tilde{G}(\mathbf{z}) - G(\mathbf{z})| = |\frac{\mu}{\tau_{sys}}| < \frac{\eta L_{\tilde{G}}}{2(e^{L_{\tilde{G}}T} - 1)}. \tag{S51}$$

Accordingly, we can set $\tilde{\mathbf{z}}(t)$ and $\mathbf{z}(t)$, solutions of the dynamical systems:

$$\dot{\tilde{\mathbf{z}}} = \tilde{G}(\mathbf{z}), \quad \begin{cases} \tilde{x}(0) = x_0 \in D \\ \tilde{y}(0) = \gamma x_0 + \mu \end{cases} \tag{S52}$$

$$\dot{\mathbf{z}} = G(\mathbf{z}), \quad \begin{cases} u(0) = x_0 \in D \\ \tilde{h}(0) = \gamma x_0 + \mu \end{cases} \tag{S53}$$

By Lemma 6, we achieve

$$\underset{t\in I}{max}|\tilde{\mathbf{z}}(t) - \mathbf{z}(t)| < \frac{\eta}{2}, \tag{S54}$$

and therefore we have:

$$\underset{t\in I}{max}|\tilde{\mathbf{x}}(t) - \mathbf{u}(t)| < \frac{\eta}{2}, \tag{S55}$$

Part3. Now by using Eq. S41 and Eq. S55, for a positive ϵ, we can design an LTC with internal dynamical state $\mathbf{z}(t)$, with τ_{sys} and W. For x(t) satisfying $\dot{\mathbf{x}} = F(\mathbf{x})$, if we initialize the network by $u(0) = x(0)$ and $h(0) = \gamma x(0) + \mu$, we obtain:

$$\underset{t\in I}{max}|\mathbf{x}(t) - \mathbf{u}(t)| < \frac{\eta}{2} + \frac{\eta}{2} = \eta < \epsilon. \tag{S56}$$

□

REMARKS. LTCs allow the elements of the hidden layer to have recurrent connections to each other. However, it assumes a feed-forward connection stream from hidden nodes to output units. We assumed no inputs to the system and principally showed that the hidden nodes' together with output units, could approximate any finite trajectory of an autonomous dynamical system.

S4 Proof of Theorem 4

In this section, we describe our mathematical notions and revisit concepts that are required to state the proof. The main statements of our theoretical results about the expressive power of time-continuous neural networks are chiefly built over the expressivity measure, *trajectory length*, introduced for static deep neural networks in (Raghu et al. 2017). It is therefore intuitive to follow similar steps with careful considerations, due to the continuous nature of the models.

S4.1 Notations

Neural network architecture – We determine a neural network architecture by $f_{n,k}(x(t), I(t), \theta)d$, with n layers (depth),width k and total number of neurons, $N = n \times k$.

Neural state, x(t) – For a layer d of a network f, $x^{(d)}(t)$ represent the neural state of the layer and is a matrix of the size $k \times m$, with m being the size of the input time series.

Inputs, I(t) – is a 2-dimensional matrix containing a 2-D trajectory for $t \in [0, t_{max}]$.

Network parameters, θ – include weights matrices for each layer d of the form $W^{(d)} \sim \mathcal{N}(0, \sigma_w^2/k)$ and bias vectors as $b^{(d)} \sim \mathcal{N}(0, \sigma_b^2)$. For CT-RNNs the vector parameter $\tau^{(d)}$ is also sampled from $\sim \mathcal{N}(0, \sigma_b^2)$

Perpendicular and parallel components – For given vectors x and y we can decompose each vector in respect to one another as $y = y_{\|} + y_{\perp}$. That is, $y_{\|}$ stands for component of y parallel to x and $y_{\perp}$ is the perpendicular component in respect to x.

Weight matrix decomposition – (Raghu et al. 2017) showed that for given non-zero vectors x and y, and a full rank matrix W, one can write a matrix decomposition for W in respect to x and y as follows: $W = {}^{\|}W_{\|} + {}^{\|}W_{\perp} + {}^{\perp}W_{\|} + {}^{\perp}W_{\perp}$, such that, ${}^{\|}W_{\perp}\, x = 0$, ${}^{\perp}W_{\perp}\, x = 0$, $y^T\, {}^{\perp}W_{\|} = 0$ and $y^T\, {}^{\perp}W_{\perp} = 0$. In this notation, the decomposition superscript on left is in respect to y and the subscript on right is in respect to x. It has also been observed that $W_{\perp}$ in respect to x can be obtained by: $W_{\perp} = W - W_{\|}$ (Raghu et al. 2017).

Lemma 7. Independence of Projections (Raghu et al. 2017). *Given a matrix W with iid entries drawn form $\mathcal{N}(0, \sigma^2)$, then its decomposition matrices $W_{\perp}$ and $W_{\|}$ in respect to x, are independent random variables.*

Proof in (Raghu et al. 2017), Appendix, Lemma 2.

Lemma 8. Norm of Gaussian Vector (Raghu et al. 2017). *The norm of a Gaussian vector $X \in \mathbb{R}^k$, with its entries sampled iid $\sim \mathcal{N}(0, \sigma^2)$ is given by:*

$$\mathbb{E}[\|X\|] = \sigma\sqrt{2}\frac{\Gamma((k+1)/2)}{\Gamma(k/2)}. \tag{S57}$$

Proof in (Raghu et al. 2017), Appendix, Lemma 3.

Lemma 9. Norm of Projections (Raghu et al. 2017). *for a $W^{k\times k}$ with conditions of Lemma 8, and two vectors, x and y, then the following holds for $x_{\perp}$ being a non-zero vector, perpendicular to x:*

$$\mathbb{E}[\|{}^{\perp}W_{\perp}\, x_{\perp}\|] = \|x_{\perp}\|\, \sigma\sqrt{2}\frac{\Gamma((k)/2)}{\Gamma((k-1)/2)} \geq \|x_{\perp}\|\, \sigma\sqrt{2}(\frac{k}{2} - \frac{3}{4})^{1/2}. \tag{S58}$$

It has also been shown in (Raghu et al. 2017): "that if $1_{\mathcal{A}}$ is an identity matrix with non-zero diagonal entry i iff $i \in \mathcal{A} \subset [k]$ and $|\mathcal{A}| > 2$, then:

$$\mathbb{E}[\|1_{\mathcal{A}}\, {}^{\perp}W_{\perp}\, x_{\perp}\|] = \|x_{\perp}\|\, \sigma\sqrt{2}\frac{\Gamma(|\mathcal{A}|/2)}{\Gamma((|\mathcal{A}|-1)/2)} \geq \|x_{\perp}\|\, \sigma\sqrt{2}(\frac{|\mathcal{A}|}{2} - \frac{3}{4})^{1/2}. \text{"} \tag{S59}$$

Proof in (Raghu et al. 2017), Appendix, Lemma 4.

Lemma 10. Norm and Translation (Raghu et al. 2017). *For X being a zero-mean multivariate Gaussian and having a diagonal covariance matrix, and μ a vector of constants, we have:*

$$\mathbb{E}[\|X - \mu\|] \geq \mathbb{E}[\|X\|]. \tag{S60}$$

Proof in (Raghu et al. 2017), Appendix, Lemma 5.

S4.2 Beginning of the proof of Theorem 4

We first establish the lower bound for Neural ODEs and then extend the results to that of CT-RNNs.

Proof. For a successive layer $d+1$ of a Neural ODE the gradient between the states at $t + \delta t$ and t, $x^{d+1}(t + \delta t)$ and $x^{d+1}(t)$ is determined by:

$$\frac{dx}{dt}^{(d+1)} = f(h^{(d)}), \qquad h^{(d)} = W^{(d)}x^{(d)} + b^{(d)}. \tag{S61}$$

Accordingly, for the latent representation (the first two principle components of the hidden state $x^{(d+1)}$), which is denoted by $z^{(d+1)}(t)$, this gradient can be determined by:

$$\frac{dz}{dt}^{(d+1)} = f(h^{(d)}), \qquad h^{(d)} = W^{(d)}z^{(d)} + b^{(d)} \tag{S62}$$

Let us continue with the zero bias case and discuss the non-zero bias case later.

We decompose $W^{(d)}$ in respect to the $z^{(d)}$, as $W^{(d)} = W_{\parallel}^{(d)} + W_{\perp}^{(d)}$. For this decomposition, the hidden state $h^{(d+1)} = W_{\parallel}^{(d)} z^{(d)}$ as the vertical components maps $z^{(d)}$ to zero.

We determine the set of indices for which the gradient state is not saturated as if f is defined by Hard-tanh activations:

$$\mathcal{A}_{W_{\parallel}^{(d)}} = \{i : i \in [k], |h_i^{(d+1)}| < 1\} \tag{S63}$$

As the decomposition components of $W^{(d)}$ are independent random variables, based on Lemma 9, we can build the expectation of the gradient state as follows:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}\right\|\right] = \mathbb{E}_{W_{\parallel}^{(d)}}\mathbb{E}_{W_{\perp}^{(d)}}\left[\left\|f(W^{(d)}z^{(d)})\right\|\right]. \tag{S64}$$

Now, if we condition on $W_{\parallel}^{(d)}$, we can replace the right-hand-side norm with the sum over the non-saturated indices, $\mathcal{A}_{W_{\parallel}^{(d)}}$ as follows:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}\right\|\right] = \mathbb{E}_{W_{\parallel}^{(d)}}\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left((W_{\perp}^{(d)})_i\, z^{(d)} + (W_{\parallel}^{(d)})_i\, z^{(d)}\right)^2\right)^{1/2}\right]. \tag{S65}$$

We need to derive a recurrence for the Eq. S65. To do this, we start a decomposition of the gradient state in respect to $z^{(d)}$ as $\frac{dz}{dt}^{(d)} = \frac{dz}{dt}_{\parallel}^{(d)} + \frac{dz}{dt}_{\perp}^{(d)}$.

Now, let $\frac{d\tilde{z}}{dt}^{(d+1)} = 1_{\mathcal{A}_{W_{\parallel}^{(d)}}} h^{(d+1)}$, be the latent gradient vector of all unsaturated units, and zeroed saturated units. Also we decompose the column space of the weight matrix in respect to $\tilde{z}^{(d+1)}$ as: $W^{(d)} = {}^{\perp}W^{(d)} + {}^{\parallel}W^{(d)}$.

Then by definition, we have the following expressions:

$$\frac{dz}{dt}_{\perp}^{(d+1)} = W^{(d)}z^{(d)}1_{\mathcal{A}} - \langle W^{(d)}z^{(d)}1_{\mathcal{A}}, \hat{z}^{(d+1)}\rangle\hat{z}^{(d+1)}, \quad \hat{:} = unit\ vector \tag{S66}$$

$${}^{\perp}W^{(d)}z^{(d)} = W^{(d)}z^{(d)} - \langle W^{(d)}z^{(d)}, \hat{\tilde{z}}^{(d+1)}\rangle\hat{\tilde{z}}^{(d+1)} \tag{S67}$$

Looking at Eq. S66 and Eq. S67, and based on the definitions provided, their right-hand-side are equal to each other for any $i \in \mathcal{A}$. Therefore, their left-hand-sides are equivalent as well. More precisely:

$$\frac{dz}{dt}_{\perp}^{(d+1)}.1_{\mathcal{A}} = {}^{\perp}W^{(d)}z^{(d)}.1_{\mathcal{A}}. \tag{S68}$$

The statement in Eq. S68 allows us to determine the following inequality, which builds up the first steps for the recurrence:

$$\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\| \geq \left\|\frac{dz}{dt}_{\perp}^{(d+1)}.1_{\mathcal{A}}\right\| \tag{S69}$$

Now let us return to Eq. S65, and plug in the following decompositions:

$$\frac{dz}{dt}^{(d)} = \frac{dz}{dt}_{\perp}^{(d)} + \frac{dz}{dt}_{\parallel}^{(d)} \tag{S70}$$

$$W_{\perp}{}^{(d)} = {}^{\parallel}W_{\perp}{}^{(d)} + {}^{\perp}W_{\perp}{}^{(d)} \qquad W_{\parallel}{}^{(d)} = {}^{\parallel}W_{\parallel}{}^{(d)} + {}^{\perp}W_{\parallel}{}^{(d)}, \tag{S71}$$

we have:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}\right\|\right] = \tag{S72}$$

$$\mathbb{E}_{W_{\parallel}^{(d)}}\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left(({}^{\parallel}W_{\perp}{}^{(d)} + {}^{\perp}W_{\perp}{}^{(d)})_i\, z_{\perp}^{(d)} + ({}^{\parallel}W_{\parallel}{}^{(d)} + {}^{\perp}W_{\parallel}{}^{(d)})_i\, z_{\parallel}^{(d)}\right)^2\right)^{1/2}\right] \tag{S73}$$

As stated in Theorem 4, we conditioned the input on its perpendicular components. Therefore, we write the recurrence of the states also for their perpendicular components by dropping the parallel components, ${}^{\parallel}W_{\perp}{}^{(d)}$ and ${}^{\parallel}W_{\parallel}{}^{(d)}$, and using Eq. S69 as follows:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}_{\perp}\right\|\right] \geq \mathbb{E}_{W_{\parallel}^{(d)}}\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left(({}^{\perp}W_{\perp}{}^{(d)})_i\, z_{\perp}^{(d)} + ({}^{\perp}W_{\parallel}{}^{(d)})_i\, z_{\parallel}^{(d)}\right)^2\right)^{1/2}\right] \tag{S74}$$

The term ${}^{\perp}W_{\parallel}{}^{(d)} z_{\parallel}^{(d)}$ is constant, as the inner expectation is conditioned on $W_{\parallel}^{(d)}$. Now by using Lemma 10, we can wirte:

$$\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left(({}^{\perp}W_{\perp}{}^{(d)})_i\, z_{\perp}^{(d)} + ({}^{\perp}W_{\parallel}{}^{(d)})_i\, z_{\parallel}^{(d)}\right)^2\right)^{1/2}\right] \geq \tag{S75}$$

$$\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left(({}^{\perp}W_{\perp}{}^{(d)})_i\, z_{\perp}^{(d)}\right)^2\right)^{1/2}\right] \tag{S76}$$

By applying Lemma 9 we get:

$$\mathbb{E}_{W_{\perp}^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_{\parallel}^{(d)}}}\left(({}^{\perp}W_{\perp}{}^{(d)})_i\, z_{\perp}^{(d)}\right)^2\right)^{1/2}\right] \geq \frac{\sigma_w}{\sqrt{k}}\sqrt{2}\frac{\sqrt{2|\mathcal{A}_{W_{\parallel}^{(d)}}|-3}}{2}\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right]. \tag{S77}$$

As we selected Hard-tanh activation functions with $p = \mathbb{P}(|h_i^{(d+1)}| < 1)$, and the condition $|\mathcal{A}_{W_{\parallel}^{(d)}}| \geq 2$ we have $\sqrt{2}\frac{\sqrt{2|\mathcal{A}_{W_{\parallel}^{(d)}}|-3}}{2} \geq \frac{1}{\sqrt{2}}\sqrt{|\mathcal{A}_{W_{\parallel}^{(d)}}|}$, and therefore we get:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}_{\perp}\right\|\right] \geq \frac{1}{\sqrt{2}}\left(\sum_{j=2}^{k}\binom{k}{j}p^j(1-p)^{k-j}\frac{\sigma_w}{\sqrt{k}}\sqrt{j}\right)\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right] \tag{S78}$$

Keep in mind that we are referring to $|\mathcal{A}_{W_{\parallel}^{(d)}}|$ as j. Now we need to bound the $\sqrt{j}$ term, by considering the binomial distribution represented by the sum. Consequently, we can rewrite the sum in Eq. S78 as follows:

$$\begin{aligned}\sum_{j=2}^{k}\binom{k}{j}p^j(1-p)^{k-j}\frac{\sigma_w}{\sqrt{k}}\sqrt{j} &= -\binom{k}{1}p^j(1-p)^{k-1}\frac{\sigma_w}{\sqrt{k}} + \sum_{j=2}^{k}\binom{k}{j}p^j(1-p)^{k-j}\frac{\sigma_w}{\sqrt{k}}\sqrt{j}\\ &= -\sigma_w\sqrt{k}p(1-p)^{k-1} + kp\frac{\sigma_w}{\sqrt{k}}\underbrace{\sum_{j=2}^{k}\frac{1}{\sqrt{j}}\binom{k-1}{j-1}p^{j-1}(1-p)^{k-j}}_{\text{XT}}\end{aligned}$$

and by utilizing Jensen's inequality with $1/\sqrt{x}$, we can simplify XT as follows as it is the expectation of the binomial distribution $(k-1, p)$ (Raghu et al. 2017):

$$\sum_{j=2}^{k}\frac{1}{\sqrt{j}}\binom{k-1}{j-1}p^{j-1}(1-p)^{k-j} \geq \frac{1}{\sqrt{\sum_{j=2}^{k} j\binom{k-1}{j-1}p^{j-1}(1-p)^{k-j}}} = \frac{1}{\sqrt{(k-1)p+1}}$$

and therefore:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\|\right] \geq \frac{1}{\sqrt{2}}\left(-\sigma_w\sqrt{k}p(1-p)^{k-1}+\sigma_w\frac{\sqrt{k}p}{\sqrt{(k-1)p+1}}\right)\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right] \tag{S79}$$

Now we need to find a range for p. (Raghu et al. 2017) showed that for Hard-tanh activations, given the fact that $h_i^{(d+1)}$ is a random variable with variance less than σ_w, for an input argument $|A| \sim \mathcal{N}(0, \sigma_w^2)$, we can lower bound $p = \mathbb{P}(|h_i^{(d+1)}| < 1)$, as follows:

$$p = \mathbb{P}(|h_i^{(d+1)}| < 1) \geq \mathbb{P}(|A| < 1) \geq \frac{1}{\sqrt{2\pi}\sigma_w}, \quad \forall\, \sigma_w \geq 1, \tag{S80}$$

and find an upper bound equal to $\frac{1}{\sigma_w}$ (Raghu et al. 2017). Therefore the equation becomes:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\|\right] \geq \frac{1}{\sqrt{2}}\left(-\sigma_w\sqrt{k}\frac{1}{\sigma_w}(1-\frac{1}{\sigma_w})^{k-1}+\sigma_w\frac{\sqrt{k}\frac{1}{\sqrt{2\pi}\sigma_w}}{\sqrt{(k-1)\frac{1}{\sqrt{2\pi}\sigma_w}+1}}\right)\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right] \tag{S81}$$

and with some simplifications:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\|\right] \geq \frac{1}{\sqrt{2}}\left(-\sqrt{k}(1-\frac{1}{\sigma_w})^{k-1}+(2\pi)^{-1/4}\frac{\sqrt{k\sigma_w}}{\sqrt{(k-1)+\sqrt{2\pi}\sigma_w}}\right)\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right] \tag{S82}$$

Now, we want to roll back Eq. S82 to arrive at the inputs. To do this, we replace the expectation term on the right-hand-side by:

$$\mathbb{E}\left[\left\|z_{\perp}^{(d)}\right\|\right] = \mathbb{E}\left[\left\|\int_t \frac{dz}{dt}_{\perp}^{(d)} dt\right\|\right] \tag{S83}$$

Proposition 1. *Let $f : \mathbb{R} \to S$, be an integratable function, on Banach space S. Then the following holds:*

$$\int_t \|f(t)\|\, dt \geq \left\|\int_t f(t)dt\right\|. \tag{S84}$$

Proof. let $x = \int_t f(t)dt \in S$, and $\Lambda \in S^*$ with $\|\Lambda\| = 1$. Then we have:

$$\Lambda x = \int_t \Lambda f(t)dt \leq \int_t \|\Lambda\|_{S^*} \|f(t)\|_S\, dt = \int_t \|f(t)\|\, dt. \tag{S85}$$

Now based on Hahn-Banach we have: $\|x\| \leq \int_t \|f(t)\|\, dt$. □

Based on Proposition 1 and Eq. S83 we have:

$$\mathbb{E}\left[\left\|\int_t \frac{dz}{dt}_{\perp}^{(d)} dt\right\|\right] \geq \mathbb{E}\left[\int_t \left\|\frac{dz}{dt}_{\perp}^{(d)}\right\| dt\right] = l(z_{\perp}^{(d)}(t)). \tag{S86}$$

Now by By recursively rolling out the the expression of Eq. S82 to arrive at input, $I(t)$ and denoting $c_1 = \frac{l(I_{\perp}(t))}{l(I(t))}$, we have:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\|\right] \geq \left(\frac{1}{\sqrt{2}}\left(-\sqrt{k}(1-\frac{1}{\sigma_w})^{k-1}+(2\pi)^{-1/4}\frac{\sqrt{k\sigma_w}}{\sqrt{(k-1)+\sqrt{2\pi}\sigma_w}}\right)\right)^d c_1 l(I(t)) \tag{S87}$$

Finally, the asymptotic form of the bound, and considering $c_1 \approx 1$ for input trajectories which are orthogonal to their successive time-points gives us:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}_{\perp}^{(d+1)}\right\|\right] \geq O\left(\frac{\sqrt{k\sigma_w}}{\sqrt{k+\sigma_w}}\right)^d \|I(t)\|. \tag{S88}$$

Eq. S88 shows the lower bound for every infinitesimal fraction of the length of the hidden state (in principle components state, z, for a neural ODE architecture. consequently, the overall trajectory length is bounded by:

$$\mathbb{E}\Big[l(z^{(d)}(t))\Big] \geq O\Big(\frac{\sqrt{k\sigma_w}}{\sqrt{k+\sigma_w}}\Big)^{d\times L} l(I(t)), \tag{S89}$$

with L being the number ODE steps. Finally we consider the non-zero bias case:

As stated in the Notations section, network parameters are set by $W^{(d)} \sim \mathcal{N}(0, \sigma_w^2/k)$ and bias vectors as $b^{(d)} \sim \mathcal{N}(0, \sigma_b^2)$. Therefore, the variance of the $h_i^{(d+1)}$ will be smaller than $\sigma_w^2 + \sigma_b^2$. Therefore we have (Raghu et al. 2017):

$$p = \mathbb{P}(|h_i^{(d+1)}| < 1) \geq \frac{1}{\sqrt{2\pi}\sqrt{\sigma_w^2+\sigma_b^2}} \tag{S90}$$

By replacing this into Eq. S79, and simplify further we get:

$$\mathbb{E}\Big[l(z^{(d)}(t))\Big] \geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t)), \tag{S91}$$

the main statement of Theorem 4 for Neural ODEs is obtained.

Deriving the trajectory length lower-bound for CT-RNNs For a successive layer $d+1$ of a CT-RNN the gradient between the states at $t+\delta t$ and t, $x^{d+1}(t+\delta t)$ and $x^{d+1}(t)$ is determined by:

$$\frac{dx}{dt}^{(d+1)} = -w_\tau^{(d+1)} x^{(d+1)} + f(h^{(d)}), \qquad h^{(d)} = W^{(d)} x^{(d)} + b^{(d)}. \tag{S92}$$

With $W_\tau^{(d+1)}$ standing for the parameter vector $\frac{1}{\tau^{(d+1)}}$, which is conditioned to be strictly positive. Accordingly, for the latent representation (the first two principle components of the hidden state $x^{(d+1)}$), which is denoted by $z^{(d+1)}(t)$, this gradient can be determined by:

$$\frac{dz}{dt}^{(d+1)} = -W_\tau^{(d+1)} z^{(d+1)} + f(h^{(d)}), \qquad h^{(d)} = W^{(d)} z^{(d)} + b^{(d)} \tag{S93}$$

An explicit Euler discretization of this ODE gives us:

$$z^{(d+1)}(t+\delta t) = (1-\delta t W_\tau^{(d+1)}) z^{(d+1)} + \delta t f(h^{(d)}), \qquad h^{(d)} = W^{(d)} z^{(d)} + b^{(d)}. \tag{S94}$$

the same discretization model for Neural ODEs gives us:

$$z^{(d+1)}(t+\delta t) = z^{(d+1)} + \delta t f(h^{(d)}), \qquad h^{(d)} = W^{(d)} z^{(d)} + b^{(d)}. \tag{S95}$$

The difference between the two representations is only a $-\delta t W_\tau^{(d+1)}$ term before $z^{(d+1)}$, which consists of $W_\tau^{(d+1)}$ that is a strictly positive random variable sampled from a folded normal distribution $\mathcal{N}(|x|; \mu_Y, \sigma_Y)$, with mean $\mu_Y = \sigma\sqrt{\frac{2}{\pi}} e^{(-\mu^2/2\sigma^2)} - \mu(1-2\Phi(\frac{\mu}{\sigma}))$ and variance $\sigma_Y^2 = \mu^2 + \sigma^2 - \mu_Y^2$ (Tsagris, Beneki, and Hassani 2014). μ and σ are the mean and variance of the normal distribution over random variable x, and Φ is a normal cumulative distribution function. For a zero-mean normal distribution with variance of σ_b^2, we get:

$$\mathcal{N}(|W_\tau|; \sigma_b\sqrt{\frac{2}{\pi}},\ (1-\frac{2}{\pi})\sigma_b^2). \tag{S96}$$

Accordingly, we approximate the lower-bound for the CT-RNNs, with the simplified asymptotic form of:

$$\mathbb{E}\Big[l(z^{(d)}(t))\Big] \geq O\left(\frac{(\sigma_w-\sigma_b)\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t)), \tag{S97}$$

This gives of the statement of the theorem for CT-RNNs.

□

Proof of Theorem 5

Distribution of parameters of LTCs

The Weight matrix for each layer d of the form $W^{(d)} \sim \mathcal{N}(0, \sigma_w^2/k)$. The bias vectors as $b^{(d)} \sim \mathcal{N}(0, \sigma_b^2)$. The vector parameter $W_\tau^{(d+1)}$ is strictly positive and it is sampled from a folded normal distribution (Tsagris, Beneki, and Hassani 2014) $\mathcal{N}(|W_\tau|; \sigma_b\sqrt{\frac{2}{\pi}},\ (1-\frac{2}{\pi})\sigma_b^2)$. The parameter stands for the inverse of the time-constant of neurons, $\frac{1}{\tau^{(d+1)}}$ The parameter $A^{(d)}$ is a weight matrix sampled from $\sim \mathcal{N}(0, \sigma_w^2/k)$.

Proof. For a successive layer $d+1$ of an LTC network, the gradient between the states at $t+\delta t$ and t, $x^{d+1}(t+\delta t)$ and $x^{d+1}(t)$ is determined by:

$$\frac{dx}{dt}^{(d+1)} = -(w_\tau^{(d+1)} + f(h^{(d)}))x^{(d+1)} + A^{(d)} f(h^{(d)}), \qquad h^{(d)} = W^{(d)} x^{(d)} + b^{(d)}. \tag{S98}$$

Accordingly, for the latent representation (the first two principle components of the hidden state $x^{(d+1)}$), which is denoted by $z^{(d+1)}(t)$, this gradient can be determined by:

$$\frac{dz}{dt}^{(d+1)} = -(w_\tau^{(d+1)} + f(h^{(d)}))z^{(d+1)} + A^{(d)} f(h^{(d)}), \qquad h^{(d)} = W^{(d)} z^{(d)} + b^{(d)}. \tag{S99}$$

We first take the expectation of norms from both side of Eq. S99, while similar to Eq. S64 and based on Lemma 9, we decompose the expectation over parallel and orthogonal components of the weight matrix $W^{(d)}$ as follows:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}\right\|\right] = \mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|-(w_\tau^{(d+1)} + f(h^{(d)}))z^{(d+1)} + A^{(d)} f(h^{(d)})\Big\|\Big]. \tag{S100}$$

We can now derive the following inequality for the norms of difference versus difference of norms as follows:

$$\mathbb{E}_{W^{(d)}}\left[\left\|\frac{dz}{dt}^{(d+1)}\right\|\right] = \tag{S101}$$

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|A^{(d)} f(h^{(d)} - (w_\tau^{(d+1)} + f(h^{(d)}))z^{(d+1)})\Big\|\Big] \geq \tag{S102}$$

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|A^{(d)} f(h^{(d)})\Big\| - \Big\|(w_\tau^{(d+1)} + f(h^{(d)}))z^{(d+1)}\Big\|\Big] \geq \tag{S103}$$

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|A^{(d)} f(h^{(d)})\Big\|\Big] - \mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|(w_\tau^{(d+1)} + f(h^{(d)}))z^{(d+1)}\Big\|\Big]. \tag{S104}$$

Let us first focus on the **right expression** in Eq. S104. The norm can be split into the norm of products, as follows:

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|(w_\tau^{(d+1)} + f(h^{(d)}))\Big\|\ \Big\|z^{(d+1)}\Big\|\Big]. \tag{S105}$$

Now by conditioning the expectations by the following rule $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$, we get:

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\Big[\Big\|(w_\tau^{(d+1)} + f(h^{(d)}))\Big\|\Big]\mathbb{E}\Big[\Big\|z^{(d+1)}\Big\|\Big]. \tag{S106}$$

We determine the set of indices for which f is not saturated and we assume that it is defined by Hard-tanh activations:

$$\mathcal{A}_{W_\parallel^{(d)}} = \{i : i \in [k], |h_i^{(d+1)}| < 1\} \tag{S107}$$

Now, if we condition on $W_\parallel^{(d)}$, we can replace the first norm by the sum over the non-saturated indices, $\mathcal{A}_{W_\parallel^{(d)}}$ as follows:

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_\parallel^{(d)}}}\left((W_\perp^{(d)} + \tfrac{w_\tau^{(d+1)}}{|\mathcal{A}|})_i\, z^{(d)} + (W_\parallel^{(d)} + \tfrac{w_\tau^{(d+1)}}{|\mathcal{A}|})_i\, z^{(d)}\right)^2\right)^{1/2}\right]\mathbb{E}\Big[\Big\|z^{(d+1)}\Big\|\Big]. \tag{S108}$$

In Eq. S108, the term $\frac{w_\tau^{(d+1)}}{|\mathcal{A}|}$ determines the average effect of the time-constant weights in the computation of each state which is a constant addition. $|\mathcal{A}|$ is the number of non-saturated states. Now by taking similar steps, from Eq. S65 to Eq. S77, and by applying Lemma 9 to Eq. S108, we have:

$$\mathbb{E}_{W_\perp^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_\parallel^{(d)}}}\left((^\perp W_\perp{}^{(d)}+\frac{w_\tau^{(d+1)}}{|\mathcal{A}_{W_\parallel^{(d)}}|})_i\, z_\perp^{(d)}\right)^2\right)^{1/2}\right]\mathbb{E}_{W^{(d)}}\left[\left\|z^{(d+1)}\right\|\right]\geq$$
$$\sqrt{\frac{\sigma_w^2}{k}+\frac{\sigma_b^2}{|\mathcal{A}_{W_\parallel^{(d)}}|^2}}\sqrt{2}\frac{\sqrt{2|\mathcal{A}_{W_\parallel^{(d)}}|-3}}{2}\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right]\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]. \tag{S109}$$

As we selected Hard-tanh activation functions with $p=\mathbb{P}(|h_i^{(d+1)}|<1)$, and the condition $|\mathcal{A}_{W_\parallel^{(d)}}|\geq 2$ we have $\sqrt{2}\frac{\sqrt{2|\mathcal{A}_{W_\parallel^{(d)}}|-3}}{2}\geq\frac{1}{\sqrt{2}}\sqrt{|\mathcal{A}_{W_\parallel^{(d)}}|}$, and therefore we can simplify further:

$$\mathbb{E}_{W_\perp^{(d)}}\left[\left(\sum_{i\in\mathcal{A}_{W_\parallel^{(d)}}}\left((^\perp W_\perp{}^{(d)}+\frac{w_\tau^{(d+1)}}{|\mathcal{A}_{W_\parallel^{(d)}}|})_i\, z_\perp^{(d)}\right)^2\right)^{1/2}\right]\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]\geq$$
$$\frac{1}{\sqrt{2}}\sqrt{\frac{\sigma_w^2|\mathcal{A}_{W_\parallel^{(d)}}|}{k}+\underbrace{\frac{\sigma_b^2}{|\mathcal{A}_{W_\parallel^{(d)}}|}}_{<<1}}\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right]\,\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]. \tag{S110}$$

Finally, we have:

$$\mathbb{E}_{W_\perp^{(d)}}\left[\left\|(w_\tau^{(d+1)}+f(h^{(d)}))\right\|\right]\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]\geq\frac{1}{\sqrt{2}}\frac{\sigma_w}{\sqrt{k}}\sqrt{|\mathcal{A}_{W_\parallel^{(d)}}|}\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right]\,\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]. \tag{S111}$$

Now if we take the computational steps from Eq. S78 to S79, we obtain the following:

$$\mathbb{E}_{W_\perp^{(d)}}\left[\left\|(w_\tau^{(d+1)}+f(h^{(d)}))\right\|\right]\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]\geq$$
$$\frac{1}{\sqrt{2}}\left(-\sigma_w\sqrt{k}p(1-p)^{k-1}+\sigma_w\frac{\sqrt{k}p}{\sqrt{(k-1)p+1}}\right)\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right]\,\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]. \tag{S112}$$

As stated before, network parameters are set by $W^{(d)}\sim\mathcal{N}(0,\sigma_w^2/k)$ and bias vectors as $b^{(d)}\sim\mathcal{N}(0,\sigma_b^2)$. Therefore, the variance of the $h_i^{(d+1)}$ will be smaller than $\sigma_w^2+\sigma_b^2$. Therefore we have (Raghu et al. 2017):

$$p=\mathbb{P}(|h_i^{(d+1)}|<1)\geq\frac{1}{\sqrt{2\pi}\sqrt{\sigma_w^2+\sigma_b^2}} \tag{S113}$$

This will give us the following asymptotic bound for the right expression of Eq. S104 as follows:

$$\mathbb{E}_{W_\perp^{(d)}}\left[\left\|(w_\tau^{(d+1)}+f(h^{(d)}))\right\|\right]\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right]\geq$$
$$O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right]\,\mathbb{E}\left[\left\|z^{(d+1)}\right\|\right] \tag{S114}$$

Now let us work with the **Left expression** in Eq. S104:

$$\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\left[\left\|A^{(d)}f(h^{(d)})\right\|\right] \tag{S115}$$

As A serves as a constant, we can take it out of the norm and the expectations. The resulting expectation of the norm, precisely expresses a deep neural network f with Hard-tanh activations, for which (Raghu et al. 2017) showed that it can be bound as follows:

$$|A^{(d)}|\mathbb{E}_{W_\parallel^{(d)}}\mathbb{E}_{W_\perp^{(d)}}\left[\left\|f(h^{(d)})\right\|\right]\geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)|A^{(d)}|\mathbb{E}\left[\left\|z_\perp^{(d)}\right\|\right] \tag{S116}$$

And since $A \sim \mathcal{N}(0, \sigma_w^2)$, the bound can be computed as follows:

$$\mathbb{E}_{W_{\parallel}^{(d)}} \mathbb{E}_{W_{\perp}^{(d)}} \Big[\Big\| A^{(d)} f(h^{(d)}) \Big\| \Big] \geq O\Bigg(\frac{\sigma_w^2 \sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}} \Bigg) \mathbb{E}\Big[\Big\| z_{\perp}^{(d)} \Big\| \Big]. \tag{S117}$$

Therefore, for the perpendicular compartments of the gradient of the hidden state, we have:

$$\begin{aligned} \mathbb{E}_{W^{(d)}} \left[\left\| \frac{dz}{dt}_{\perp}^{(d+1)} \right\| \right] \geq O\Bigg(\frac{\sigma_w \sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}} \Bigg) \mathbb{E}\Big[\Big\| z_{\perp}^{(d)} \Big\| \Big] \, \mathbb{E}\Big[\Big\| z^{(d+1)} \Big\| \Big] + \\ O\Bigg(\frac{\sigma_w^2 \sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}} \Bigg) \mathbb{E}\Big[\Big\| z_{\perp}^{(d)} \Big\| \Big]. \end{aligned} \tag{S118}$$

If we simplify further and considering the fact that we are shaping the recurrence for every infinitesimal δt of the system's dynamics, we get the following asymptotic bound:

$$\mathbb{E}_{W^{(d)}} \left[\left\| \frac{dz}{dt}_{\perp}^{(d+1)} \right\| \right] \geq O\Bigg(\frac{\sigma_w \sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}} \Bigg) \mathbb{E}\Big[\Big\| z_{\perp}^{(d)} \Big\| \Big] \, \Big(\sigma_w + \frac{\| z^{(d+1)} \|}{\min(\delta t, L)} \Big). \tag{S119}$$

Now similar as before, by recursively unrolling the n layer neural network f to reach the input, denoting $c_1 = \frac{l(I_{\perp}(t))}{l(I(t))} \approx 1$, and establishing the bound for an input sequence of length T, for a layer d of a network we get:

$$\mathbb{E}\Bigg[l(z^{(d)}(t)) \Bigg] \geq O\Bigg(\Big(\frac{\sigma_w \sqrt{k}}{\sqrt{\sigma_w^2 + \sigma_b^2 + k\sqrt{\sigma_w^2 + \sigma_b^2}}} \Big)^{d \times L} \Big(\sigma_w + \frac{\| z^{(d)} \|}{\min(\delta t, L)} \Big) \Bigg) l(I(t)). \tag{S120}$$

Equation S120 gives us the statement of the theorem.

□

S5 Experimental Setup - Section 6

Here, we describe the experimental setup for the tasks discussed in Tables 3, 4, 5, and 6.

For each experiment we performed a training-validation-test split of 75:10:15 ratio, with the exact ratios depending on the specific dataset. After each training epoch the validation metric was evaluated. We kept a backup of the network weights of the configuration that achieved the best validation metric over the whole training process. At the end of the training process, we restored the backed-up weights and evaluated the network on the test-set. We repeated this procedure for five times with different weight initializations and reported the mean and standard deviation in Tables 3, 4, 5, and 6. Hyper-parameters are shown in Table S1.

Each RNN consists of 32 hidden units. As each task requires a different number of output units, the output of the RNNs were fed through a learnable linear layer to project the output to the required dimension. Note that the objective of our experimental setup is not to build the best predictive models, but to empirically compare the expressive power and generalization abilities of various RNN models.

We implemented all RNN models in TensorFlow1.14. For the sake of reproducability, we have submitted all code and data along with our submission and will make them publicly available upon acceptance.

ODE solvers For simulating the differential equations we used an explicit Euler methods for CT-RNNs, a 4-th order Runge-Kutta method for the Neural ODE as suggested in (Chen et al. 2018), and our fused ODE solver for LTCs. All ODE solvers were fixed-step solvers. The time-step is set to 1/6 of the input sampling frequency, i.e., each RNN step consists of 6 ODE solver steps.

Hand Gesture Segmentation The experiment concerns the temporal segmentation of hand gestures. The dataset consists of seven recordings of individuals performing a sequence of hand gesticulations (Wagner et al. 2014). The input features at each time-step are comprised of 32 data points recorded from a motion detection sensor. The output, at each time step, represents one of the five possible hand gestures; rest position, preparation, stroke, hold, and retraction. The objective is to train a classifier to detect hand gestures from the motion data.

We cut each of the seven recordings into overlapping sub-sequences of exactly 32 time-steps. We randomly separated all sub-sequences into non-overlapping training (75%), validation (10%), and test (15%) sets. Input features were normalized to have zero mean and unit standard deviation. We used the categorical classification accuracy as the performance metric.

Room Occupancy The objective is to detect whether a room is occupied by observations recorded from five physical sensor streams, such as temperature, humidity, and CO2 concentration sensors (Candanedo and Feldheim 2016). Input data and binary labels are sampled in one-minute long intervals.

The original dataset consists of a pre-defined training and test set. We used the binary classification accuracy as the performance metric. We cut the sequences of each of the two sets into a training and test set of overlapping sub-sequences of exactly 32 time-steps. Note that no item from the test set was leaking into the training set during this process. Input features of all data were normalized by the mean and standard deviation of the training set, such that the training set has zero mean and unit standard deviation. We select 10% of the training set as the validation set.

Human Activity Recognition This task involves the recognition of human activities, such as walking, sitting, and standing, from inertial measurements of the user's smartphone (Anguita et al. 2013). Data consists of recordings from 30 volunteers performing activities form six possible categories. Input variables are filtered and are pre-processed to obtain a feature column of 561 items at each time step.

The output variable represents one of six activity categories at each time step. We employed the categorical classification accuracy as our performance metric. The original data is already split into a training and test set and preprocessed by temporal filters. The accelerometer and gyroscope sensor data were transformed into 561 features in total at each time step. We aligned the sequences of the training and test set into overlapping sub-sequences of exactly 32 time-steps. We select 10% of the training set as the validation set.

Sequential MNIST We also worked with MNIST. While the original MNIST is a computer vision classification problem, we transform the dataset into a sequence classification task. In particular, each sample is encoded as a 28-dimensional time-series of length 28. Moreover, we downscale all input feature to the range [0,1]. We exclude 10% of the training set and use it as our validation set.

Traffic Estimation The objective of this experiment is to predict the hourly westbound traffic volume at the US Interstate 94 highway between Minneapolis and St. Paul. Input features consist of weather data and date information such as local time and flags indicating the presence of weekends, national, or regional holidays. The output variable represents the hourly traffic volume.

The original data consists of hourly recordings between October 2012 and October 2018, provided by the Minnesota Department of Transportation and OpenWeatherMap. We selected the seven columns of the data as input features: 1. Flag indicating whether the current day is a holiday, 2. The temperature in Kelvin normalized by annual mean, 3. Amount of rainfall, 4. Amount of snowfall, 5. Cloud coverage in percent, 6. Flag indicating whether the current day is a weekday, and 7. time of the day pre-processed by a sine function to avoid the discontinuity at midnight. The output variable was normalized to have zero mean and unit standard deviation. We used the mean-squared-error as training loss and evaluation metric. We split the data into partially overlapping sequences lasting 32 hours. We randomly separated all sequences into non-overlapping training (75%), validation (10%), and test (15%) set.

Power We used the "Individual household electric power consumption Data Set" from the UCI machine learning repository (Dua and Graff 2017). Objective of this task is to predict the hourly active power consumption of a household. Input features are secondary measurement such as the reactive power draw and sub-meterings. Approximately 1.25% of all measurements are missing, which we overwrite by the most recent measurement of the same feature. We apply a feature-wise whitening normalization and split the dataset into non-overlapping sub-sequences of length 32 time-steps. The prediction variable (active power consumption) is also whitened. We use the squared-error as optimization loss and evaluation metric.

Ozone Day Prediction The objective of task is to forecast ozone days, i.e., days when the local ozone concentration exceeds a critical level. Input features consist of wind, weather, and solar radiation readings.

The original dataset "Ozone Level Detection Data Set" was taken from the UCI repository (Dua and Graff 2017) consists of daily data points collected by the Texas Commission on Environmental Quality (TCEQ). We split the 6-years period into overlapping sequences of 32 days. A day was labeled as ozone day if, for at least 8 hours, the exposure to ozone exceeded 80 parts per billion. Inputs consist of 73 features, including wind, temperature, and solar radiation data. The binary predictor variable has a prior of 6.31%, i.e., expresses a 1:15 imbalance. For the training procedure, we weighted the cross-entropy loss at each day, depending on the label. Labels representing an ozone day were assigned 15 times the weight of a non-ozone day. Moreover, we reported the F_1-score instead of standard accuracy (higher score is better).

In roughly 27% of all samples, some of the input features were missing. To not disrupt the continuity of the collected data, we set all missing features to zero. Note that such zeroing of some input features potentially negatively affects the performance of our RNN models compared to non-recurrent approaches and filtering out the missing data. Consequently, ensemble methods and model-based approaches, i.e., methods that leverage domain knowledge (Zhang and Fan 2008), can outperform the end-to-end RNNs studied in our experiment. We randomly split all sub-sequences into training (75%), validation (10%), and test (15%) set.

Person Activity - 1st Setting In this setting we used the "Human Activity" dataset described in (Rubanova, Chen, and Duvenaud 2019). However, as we use different random seeds for the training-validation-test splitting, and a different input representation, our results are not transferable directly to those obtained by (Rubanova, Chen, and Duvenaud 2019), in the current setting.

The dataset consists of 25 recordings of various physical activity of human participants, for instance, among others lying

down, walking, sitting on the ground. The participants were equipped with four different sensors, each sampling at a period of 211 ms.

Similar to (Rubanova, Chen, and Duvenaud 2019), we packed the 11 activity categories into 7 classes. No normalization is applied to the input features. The 25 sequences were split into partially overlapping sub-sequences of length 32 time-steps.

unlike Rubanova et al. (Rubanova, Chen, and Duvenaud 2019), we represented the input time-series as a 7-dimensional feature vector, where the first 4 entries specified the sensor ID and the last 3 entries the sensor values. Due to the high sampling frequency we discarded all timing information.

The results are reported in Table 4.

Person Activity - 2nd Setting We setup a second experimental setup based on the same dataset as the person activity task above. In contrast to the first setting, we made sure that the training and test sets are equivalent to (Rubanova, Chen, and Duvenaud 2019) in order to be able to directly compare results. However, we apply the same pre-processing as in our experiment before. In particular, represent the datasets as irregularly sampled in time and dimension using a padding and masking, which results in a 24-dimensional input vector. On the other hand, we discard all time information and feed the input data as described above in the form of a 7-dimensional vector. Note that the data is still the same, just represented in a different format.

Based on the training - test split of (Rubanova, Chen, and Duvenaud 2019) we select 10% of the training set as our validation set. Moreover, we train our model for 400 epochs and select the epoch checkpoint which achieved the best results on the validation set. This model is then selected to be tested on the test set provided by (Rubanova, Chen, and Duvenaud 2019). Results are reported in Table 5.

Half-Cheetah Kinematic modeling This task is inspired by the physics simulation experiment of Chen et al. (Rubanova, Chen, and Duvenaud 2019), which evaluated how well RNNs are suited to model kinematic dynamics. In our experiment, we collected 25 rollouts of a pre-trained controller for the HalfCheetah-v2 gym environment (Brockman et al. 2016). Each rollout is composed of a series of 1000 17-dimensional observation vectors generated by the MuJoCo physics engine (Todorov, Erez, and Tassa 2012). The task is then to fit the observation space time-series in an autoregressive fashion.To increase the difficulty, we overwrote 5% of the actions produced by the pre-trained controller by random actions. We split the data into training, test, and validation sets by a ratio of 2:2:1. Training loss and test metric were mean squared error (MSE). Results were reported in Table 6.

S6 Hyperparameters and Parameter counts - Tables 3, 4, and 6

Table S1: Hyperparameters used for the experimental evaluations

Parameter	**Value**	**Description**
Number of hidden units	32	
Minibatch size	16	
Learning rate	0.001 - 0.02	
ODE-solver step	1/6	relative to input sampling period
Optimizer	Adam (Kingma and Ba 2014)	
β_1	0.9	Parameter of Adam
β_2	0.999	Parameter of Adam
$\hat{\epsilon}$	1e-08	Parameter of Adam
BPTT length	32	Backpropagation through time length in time-steps
Validation evaluation interval	1	Every x-th epoch the validation metric will be evaluated
Training epochs	200	

Table S2: Number of parameters of various RNN model in relation to the RNN width k, the number of hidden layers n, and the number of decay slots m.

Model	**Parameter count (asymptotic)**	**Parameter count (exact)**
CT-RNN	$O(nk^2)$	$nk^2 + 2nk$
ODE-RNN	$O(nk^2)$	$nk^2 + nk$
LSTM	$O(nk^2)$	$4nk^2 + 4nk$
CT-GRU	$O(mk^2)$	$2mk^2 + 2mk + k^2 + k$
LTC	$O(nk^2)$	$4nk^2 + 3nk$

S7 Additional trajectory space representations:

Trajectory space representation for the results provided can be viewed at: https://www.dropbox.com/s/ly6my34mbvsfi6k/additional_LTC_neurIPS_2020.zip?dl=0

S8 Trajectory Length results

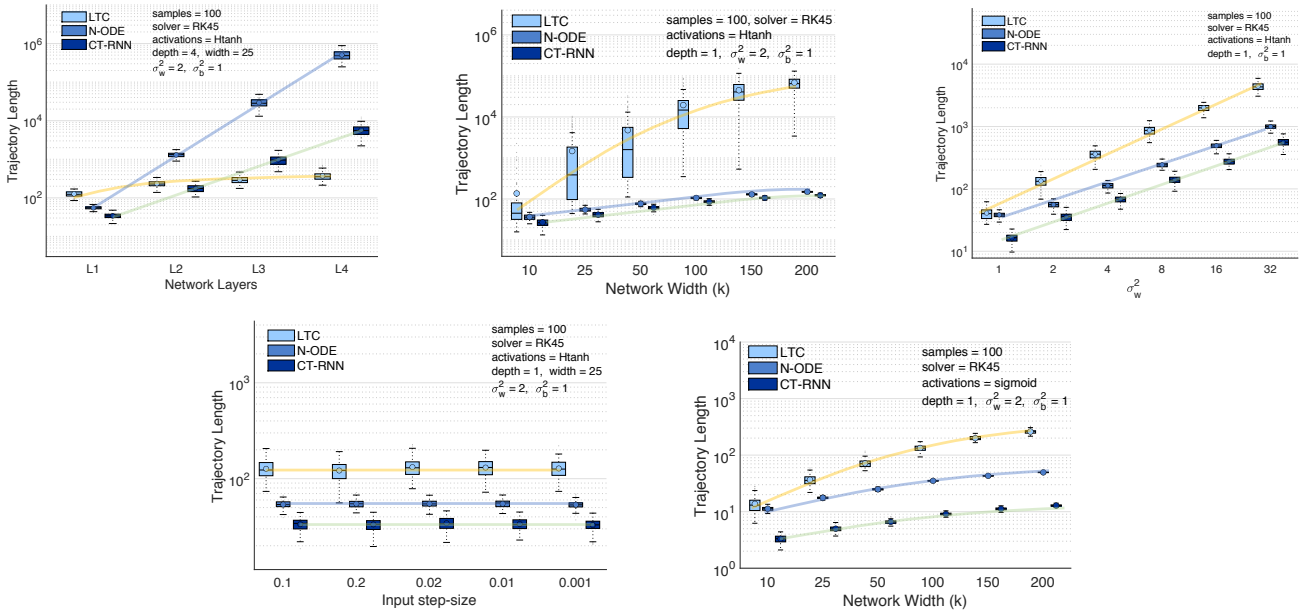

Figure S1: Additional trajectory length results.

S9 Code and Data availability

All code and data are publicly accessible at: https://github.com/raminmh/liquid_time_constant_networks.