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:

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

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

[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:

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

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:

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

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.g. 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, $x := x_{t_0} \sim p(x_{t_0})$
  for $j = 1 \ldots T$ **do**
    $x = f(I(t), x)$, $\hat{y}(t) = W_{out}.x + b_{out}$, $L_{total} = \sum_{j=1}^{T} L(y_j(t), \hat{y}_j(t))$, $\nabla L(\theta) = \frac{\partial L_{tot}}{\partial \theta}$
    $\theta = \theta - \alpha \nabla L(\theta)$
  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

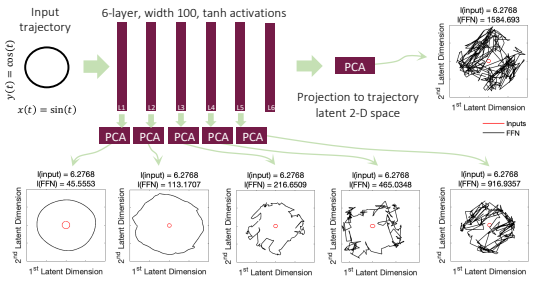

Figure 1: Trajectory's latent space becomes more complex as the input passes through hidden layers.

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 $F : S \rightarrow \mathbb{R}^n$ 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

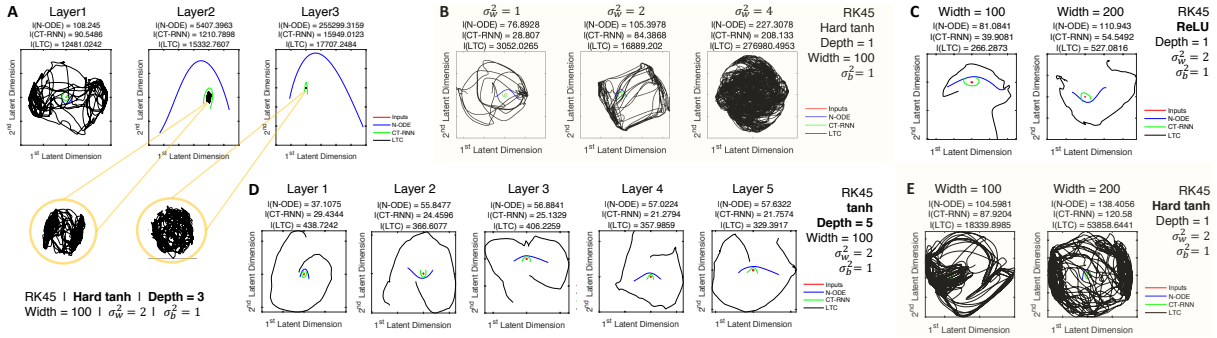

Figure 2: Trajectory length deformation A) in network layers with Hard-tanh activations, B) as a function of the weight distribution scaling factor, C) as a function of network width (ReLU), D) in network layers with logistic-sigmoid activations and E) as a function of width (Hard-tanh).

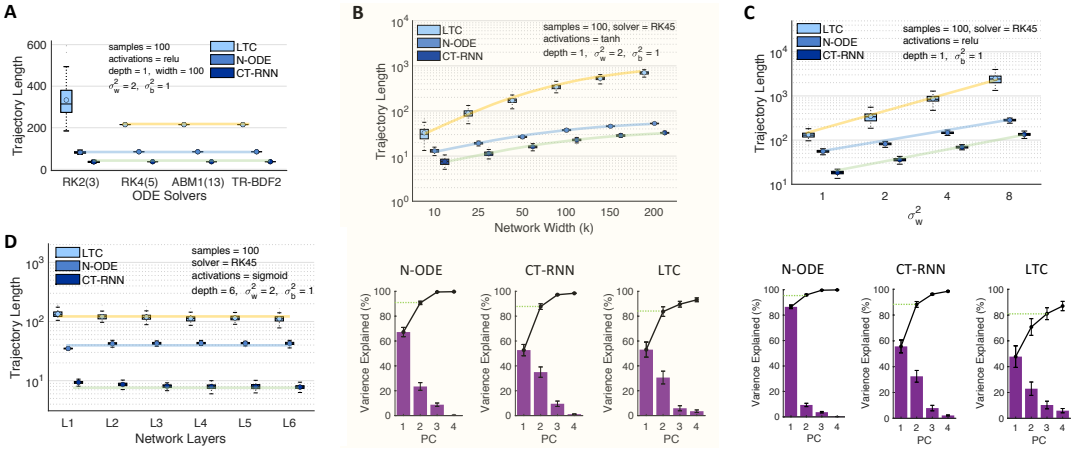

Figure 3: Dependencies of the trajectory length measure. A) trajectory length vs different solvers (variable-step solvers). RK2(3): Bogacki-Shampine Runge-Kutta (2,3) (Bogacki and Shampine 1989). RK4(5): Dormand-Prince explicit RK (4,5) (Dormand and Prince 1980). ABM1(13): Adams-Bashforth-Moulton (Shampine 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% ± 0.59	59.01% ± 1.22	46.97% ± 3.03	68.31% ± 1.78	**69.55%** ± **1.13**
Occupancy	(accuracy)	93.18% ± 1.66	94.54% ± 0.54	90.15% ± 1.71	91.44% ± 1.67	**94.63%** ± **0.17**
Activity recognition	(accuracy)	95.85% ± 0.29	95.73% ± 0.47	**97.26%** ± 0.10	96.16% ± 0.39	95.67% ± 0.575
Sequential MNIST	(accuracy)	**98.41%** ± 0.12	96.73% ± 0.19	97.61% ± 0.14	98.27% ± 0.14	97.57% ± 0.18
Traffic	(squared error)	0.169 ± 0.004	0.224 ± 0.008	1.512 ± 0.179	0.389 ± 0.076	**0.099** ± 0.0095
Power	(squared-error)	0.628 ± 0.003	0.742 ± 0.005	1.254 ± 0.149	**0.586** ± 0.003	0.642 ± 0.021
Ozone	(F1-score)	0.284 ± 0.025	0.236 ± 0.011	0.168 ± 0.006	0.260 ± 0.024	**0.302** ± 0.0155

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

Algorithm	**Accuracy**
LSTM	83.59%± 0.40
CT-RNN	81.54%± 0.33
Latent ODE	76.48%± 0.56
CT-GRU	85.27%± 0.39
LTC (ours)	**85.48%± 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 4^{th} 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

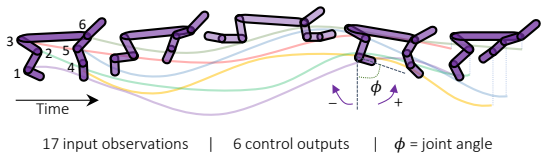

Figure 4: Half-cheetah physics simulation

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.