Liquid Time-constant Networks

Pradeep Nayak[1],Ankitha B[2], Bhumika S K [2], Satheesh D S[2] ,Sreejith R [2]

ALVAS INSTITUE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE AND EGINEERING

ABSTRACT

Liquid Time-Constant Networks (LTCs) are a new type of time-continuous recurrent neural networks (RNNs) that use linear first-order dynamical systems with nonlinear connected gates. This review study investigates LTCs. Compared to conventional RNNs, LTCs are more stable and expressive in capturing intricate temporal connections since they dynamically modulate their time-constants. We provide an overview of the theoretical underpinnings of LTCs, highlighting their expressive potential in latent trajectory space and their stable and constrained behaviour. Their improved performance on time-series prediction problems is demonstrated empirically. There is also discussion of the ramifications of these developments and possible future paths.

---

1 Introduction

Ordinary differential equations (ODEs) define the continuous-time hidden states of recurrent neural networks, which are efficient methods for modelling time series data that are widely utilised in commerce, industry, and medicine. The solution of this equation (Chen et al. 2018):
$dx(t)/dt = f(x(t),I(t), t, \theta)$ with a neural network f parametrised by θ defines the state of a neural ODE, $x(t) \in R D$. The state can then be calculated using a numerical ODE solver, and the network can be trained by applying the adjoint method, gradient descent through the solver
(Lechner et al. 2019), reverse-mode automatic differentiation (Rumelhart, Hinton, and Williams 1986), or by viewing the solver as a black-box (Chen et al. 2018; Dupont, Doucet, and Teh 2019; Gholami, Keutzer, and Biros 2019).
The following formula (Funahashi and Nakamura 1993) may be used to identify a more stable continuous-time recurrent neural network (CT-RNN) as opposed to explicitly defining the derivatives of the hidden-state by a neural network f:
This formula, $dx(t)dt = -x(t)\ \tau + f(x(t),I(t), t, \theta)$, uses the term $-x(t)\ \tau$ to help the autonomous system get to equilibrium with a time-constant τ. Time is represented by t, the hidden state is denoted by x(t), the input is denoted by I(t), and f is parametrised by θ.
We provide an alternative formulation in which a system of linear ODEs of the type $dx(t)/dt = -x(t)/\tau + S(t)$ declares the hidden state flow of a network, and $S(t) \in RM$ denotes the nonlinearity identified by $S(t) = f(x(t),I(t), t, \theta)(A-x(t))$, with parameters θ and A. Next, by inserting S into the

$$dx(t)\ dt = -\ h\ 1\ \tau + f(x(t),I(t), t, \theta)\ i\ x(t) + f(x(t),I(t), t, \theta)A$$

A unique time-continuous RNN instance with many properties and advantages is expressed by Eq. 1.
Time-constant liquid. In addition to calculating the derivative of the hidden state x(t), a neural network f also acts as an input-dependent variable time-constant for the learning system ($\tau sys = \tau\ 1+\tau\ f(x(t),I(t),t,\theta)$). A time constant is a parameter that describes the speed and coupling sensitivity of an ODE.This characteristic makes it possible for individual components of the hidden state to recognise certain dynamical systems for input characteristics that arrive at every time point. These models are known as liquid time-constant recurrent neural networks, or LTCs for short. LTCs may be built using any number of different ODE solvers.

We provide a workable fixed-step ODE solver in Section 2 that combines the computational effectiveness of the explicit Euler techniques with the stability of the implicit Euler.

Automated distinction of LTCs in reverse mode. Differentiable computational graphs are realised via LTCs. They can be taught using a variety of gradient-based optimisation strategies, just as neural ODEs. Instead of optimising LTCs using an adjoint-based technique, we elect to use a vanilla backpropagation through-time strategy, which compromises memory for numerical precision during a backward-pass (Pontryagin 2018). We provide a detailed justification for our decision in Section 3.
Dynamics with boundaries: stability. We demonstrate that the state and time-constant of LTCs are limited to a finite range in Section 4.

When the system's inputs keep growing, this feature is ideal since it guarantees the output dynamics' stability.

We examine the approximation capabilities of LTCs both theoretically and quantitatively in Section 5. We demonstrate the universality of LTCs using a functional analysis technique. After that, we examine their expressivity in further detail in comparison to other time-continuous models. To do this, we use a latent trajectory representation to measure the trajectory length of network activations. In order to gauge the expressivity of feed-forward deep neural networks, trajectory length was introduced (Raghu et al. 2017). We apply these standards to the continuous-time family.
Time-series modeling. In Section 6, we carry out a number of eleven time-series prediction tests and contrast how well contemporary RNNs perform with the time-continuous models. We see that LTCs perform better in the vast majority of circumstances.
Why this particular formulation? The selection of this specific depiction was made for two main reasons: I) According to Hasani et al. (2020), the LTC model shares certain similarities with the computational models of brain dynamics in tiny species when combined with synaptic transmission processes. A system of linear ODEs of the following form may be used to express the dynamics of non-spiking neurones' potential, v(t): dv/dt = −glv(t) + S(t), where S is the total of all synaptic inputs to the cell from presynaptic sources and gl is a leakage conductance.
It is possible to estimate all synaptic currents to the cell. In steady state by the subsequent nonlinearity (Wicks, Roehrig, and Rankin 1996; Koch and Segev 1998):
S(t) = f(v(t),I(t)),(A − v(t)), where f(.) is a sigmoidal nonlinearity that depends on its external inputs, I(t), and the state of all neurones, v(t), that are presynaptic to the present cell. These two equations can be entered to get an equation that resembles Eq. 1.
This foundation serves as an inspiration for LTCs.

II) Using a bilinear dynamical system approximation (Penny, Ghahramani, and Friston 2005), Eq. 1 may approximate the well-known Dynamic Causal Models (DCMs) (Friston, Harrison, and Penny 2003).
A second-order (Bilinear) approximation of the dynamical system

$$dx/dt = F(x(t),I(t), \theta)$$

is used to define DCMs, which would provide the following notation (Friston, Harrison, and Penny 2003): dx/dt = (A + I(t)B)x(t) + CI(t) with A = dF dx , B = dF 2 dx(t)dI(t) , C = dF dI(t)

Bilinear dynamical systems and DCM have demonstrated potential in the acquisition of complicated fMRI time-series signals. With a loose biological inspiration, LTCs are presented as variations of continuous-time (CT) models that exhibit excellent expressivity, stability, and time series modelling capabilities.

Forward-passing two LTCs using a fused ODE solver Because the LTC semantics are nonlinear, it is not easy to solve Eq. 1 analytically. However, a number may be used to calculate the state of the system of ODEs at any given time point T.

Algorithm 1 LTC update by fused ODE Solver

Parameters: $\theta = \{\tau^{(N\times1)}$ = time-constant, $\gamma^{(M\times N)}$ = weights, $\gamma_r^{(N\times N)}$ = recurrent weights, $\mu^{(N\times1)}$ = biases$\}$, $A^{(N\times1)}$ = 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 $\mathbf{x}_{t+\Delta t}$

Cal ODE solver that uses a trajectory from x(0) to x(T) to simulate the system. The continuous simulation interval [0, T] is broken down into a temporal discretisation, [t0, t1,... tn], via an ODE solver. The updating of the neural states from ti to ti+1 is therefore all that is required for a solver's step.
A system of stiff equations is realised by the ODE of LTCs (Press et al. 2007). An exponential number of discretisation steps are needed to simulate this kind of ODE using an integrator based on RungeKutta (RK). As a result, RK-based ODE solvers, such Dormand-Prince (default in torchdiffeq; Chen et al. 2018), are inappropriate for LTCs. Therefore, we combine the explicit and implicit Euler techniques to create a novel ODE solver (Press et al. 2007). This selection of An implicit updating equation achieves stability through the use of the discretisation approach. In order to do this, a given dynamical system of the type

$$dx/dt = f(x)$$

is numerically unrolled by the Fused Solver by:

$$x(ti+1) = x(ti) + \Delta t f(x(ti), x(ti+1)).$$

Specifically, we substitute x(ti+1) for just the x(ti) that appear linearly in f. Consequently, it is possible to answer Eq 2 symbolically for x(ti+1). When we solve the LTC representation for x(t + Δt) using the Fused solver, we obtain:

$$x(t + \Delta t) = x(t) + \Delta t f(x(t),I(t), t, \theta)A\ 1 + \Delta t\ 1/\tau + f(x(t),I(t), t, \theta)\ .$$

For an LTC network, Eq. 3 calculates one update state. Accordingly, Algorithm 1 demonstrates how to construct an LTC network given a parameter space θ. It is assumed that f has an arbitrary activation function (e.g., f = tanh(γrx + γI + μ) for a tanh nonlinearity).
For an input sequence of length T, the algorithm's computational complexity is O(L × T), where L is the number of discretisation steps. It makes sense that a dense representation of an LTC network with N neurones and a dense representation of an LSTM network with N cells (Hochreiter and Schmidhuber 1997) would be of the same complex.

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

3 TRAINING LTC NETWORKS BY BPTT

In order to conduct reverse-mode automated differentiation, it was proposed that neural ODEs be trained using a fixed memory cost for each layer in a neural network f using the adjoint sensitivity approach (Chen et al. 2018). However, when using the adjoint approach in reverse mode, numerical mistakes occur. As the community has frequently remarked, this issue occurs because the adjoint technique forgets the forward-time computational paths (Gholami, Keutzer, and Biros 2019; Zhuang et al. 2020).

Conversely, during the reverse mode integration, direct backpropagation through time (BPTT) sacrifices memory in order to accurately recover the forwardpass (Zhuang et al. 2020). In order to preserve a very precise backward-pass integration via the solver, we therefore set out to create a vanilla BPTT method. For this reason, an RNN may be constructed by iteratively folding the output of an ODE solver (a vector of neural states), and the system can then be trained using the learning technique outlined in technique 2. A standard stochastic gradient descent (SGD) is used in Algorithm 2.
A more effective version of the SGD, such Adam (Kingma and Ba 2014), which we employ in our research, can be used in its place.

COMPLEXITY. The complexity of our vanilla BPTT algorithm in comparison to an adjoint technique is summed up in Table 1.
At significant memory costs, we attain a high level of accuracy on both forward and backward integration paths with comparable computational complexity.

4 BOUNDS ON T AND NEURAL STATE OF LTCS

An ODE that changes its time constant according to inputs is used to depict LTCs. Therefore, it's critical to determine if

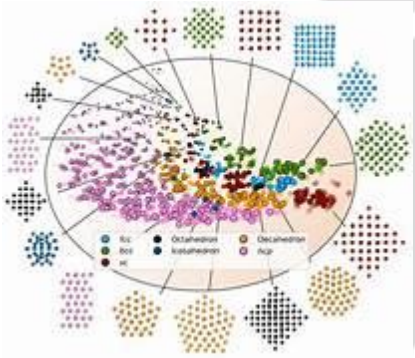

Figure 1: As the input moves through hidden layers, the trajectory's latent space gets increasingly complicated.

For infinitely many incoming inputs, LTCs remain stable (Hasani et al. 2019; Lechner et al. 2020b). In this section, we demonstrate that the LTC neurones' state and time-constant are constrained to a finite range, as shown in accordance with the theorems .**Theorem 1:** Let neurone i receive M incoming connections, and let xi represent its state inside an LTC network described by Eq. 1. The neuron's time-constant, τsysi, is then limited to the range shown below:

$$\tau i/(1 + \tau i Wi) \leq \tau sysi \leq \tau i ,$$

The Appendix contains the proof. It is built on the basis of neural network f's limited, monotonically growing sigmoidal nonlinearity and how the dynamics of the LTC network replace it. As we learn more formally in Section 5, the expressivity of this type of time-continuous RNNs is greatly Improved by a stable variable time-constant. The second theorem. Let neurone i receive M incoming connections, and let xi represent its state inside an LTC as determined by Eq. 1. Then, over a finite interval Int $\in$ [0, T], the hidden state of any neurone i is limited as follows:

$$\min(0, Amin\ i\ ) \leq xi(t) \leq \max(0, Amax\ i\ )$$

The Appendix contains the proof. It is built using an explicit Euler discretisation to approximate the ODE model and the sign of the compartments in the LTC equation. Theorem 2 demonstrates state stability, a desired characteristic of LTCs that ensures that even when their inputs increase to infinity, their outputs will never burst. The expressive capability of LTCs is next examined in relation to the family of time-continuous models, which includes neural ordinary differential equations and CT-RNNs (Chen et al. 2018; Rubanova, Chen, and Duvenaud 2019).

5 ON THE EXPRESSIVE POWER OF LTCS

The expressivity problem is the understanding of how the structural characteristics of neural networks dictate which functions they can calculate. Theoretical studies based on functional analysis are among the earliest attempts to quantify the expressivity of neural networks. They demonstrate how three-layer neural networks may

accurately mimic any finite collection of continuous mappings. The universal approximation theorem is what this is called (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, σ 2 w = 2, σ 2 b = 1.

Standard RNNs (Funahashi 1989) and even continuous-time RNNs (Funahashi and Nakamura 1993) were included in the extension of universality. We can also demonstrate that LTCs are universal approximators by carefully examining them.
The third theorem states that $x \in Rn$, $S \subset Rn$, and $\dot{x} = F(x)$ are autonomous ODEs with F: S → Rn a C 1-mapping on S. Assume that the simulation of the system is bounded in the interval I = [0, T] and let D represent a compact subset of S. An LTC network with N hidden units, n output units, and an output internal state u(t) is therefore present for a positive {, as defined by Eq. 1, such that for each rollout $\{x(t)|t \in I\}$ of the system with starting value $x(0) \in D$,

$\max t \in I |x(t) - u(t)|$

The definition of an n-dimensional dynamical system and its placement within a higher level system constitute the central notion of the proof. An LTC is the second system. When comparing the semantics of the two systems, the main difference between the proof of LTC's universality and that of CT-RNNs (Funahashi and Nakamura 1993) is that the LTC network has a nonlinear input-dependent term in its time-constant module, which makes some parts of the proof non-trivial.
The expressive potential of a neural network model is generally examined via the universal approximation theorem. The theorem however, does not provide us with a foundational measure on where the separation is between different neural network architectures. Consequently, a more exacting expression metric is required to compare models, particularly those networks that specialise in 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 timecontinuous models, both theoretically and quantitatively, which we explore in the next section.

5.1 MEASURING EXPRESSIVITY BY TRAJECTORY LENGTH

Given the network's capacity (depth, width, type, and weights configuration), a measure of expressivity must consider the many levels of complexity that a learning system may calculate. In order to unify expressivity measures for static deep networks, Raghu et al. created the trajectory length 2017). Here, one assesses how a deep model gradually converts a particular input trajectory—such as a circular, two-dimensional input—into a more intricate pattern.
After that, we may examine the activations of the generated network using principal component analysis (PCA). Then, To determine the output trajectory's relative complexity, we measure its length in a two-dimensional latent space (see Fig. 1). According to Raghu et al. (2017), the trajectory length is the arc length of a given trajectory I(t), such as a circle in 2D space: $l(I(t)) = R\ t\ kdI(t)/dtk\ dt$. In contrast to many other measures of expressivity (Pascanu, Montufar, and Bengio 2013; Montufar et al. 2014; Serra, Tjandraatmadja, and Ramalingam 2017; Gabrie et al. 2018; Hanin and Rolnick 2018, ´ 2019; Lee, Alvarez-Melis, and Jaakkola 2019), a barrier between networks of shallow and deep architectures can be established by setting a lower-bound for the trajectory length growth, independent of any assumptions regarding the network's weight configuration (Raghu et al. 2017). Our goal was to lower-bound the trajectory length in order to compare the expressivity of the models and to expand the trajectory-space analysis of static networks to time-continuous (TC) models. In order to do this, we created examples of shared f neural ODEs, CT-RNNs, and LTCs. Weights ∼ N (0, σ2 w/k) and biases ∼ N (0, σ2 b) were used to initialise the networks. Next, for varied weight profiles, we run forward-pass simulations using various ODE solver types while subjecting the networks to a circular input trajectory I(t) = {I1(t) = sin(t), I2(t) = cos(t)}, for $t \in [0, 2\pi]$. We consistently found more complicated trajectories for LTCs when we examined the first two principal components of hidden layer activations (with an average variance explained of > 80%).

A preview of our empirical findings is shown in Fig. 2. Every network positions of CT-RNNs, LTCs, and neural ODEs Having a common f. Weights ∼ N (0, σ2) were used to initialise the networks. biases ∼ N (0, σ2 b), and w/k). Next, for varied weight profiles, we run forward-pass simulations using various ODE solver types while subjecting the networks to a circular input trajectory I(t) = {I1(t) = sin(t), I2(t) = cos(t)}, for t ∈ [0, 2π]. We consistently found more complicated trajectories for LTCs when we examined the first two principal components of hidden layer activations (with an average variance explained of > 80%). A preview of our empirical findings is shown in Fig. 2. The Dormand-Prince explicit Runde-Kutta(4,5) solution (Dormand and Prince 1980) with a configurable step size implements all networks.

The following observations were made by us:

I) Regardless of their weight profile, neural ODEs and CT-RNNs with Hard-tanh and ReLU activations show exponential development in trajectory length (Fig. 2A) and an unaltered latent space structure.

II) When constructed using Hard-tanh and ReLU, LTCs exhibit a slower trajectory length growth rate at the expense of achieving high levels of complexity (Fig. 2A, 2C, and 2E).

III) In every instance, we saw a longer and more intricate latent space behaviour for the LTC networks (Fig. 2B to 2E), with the exception of multi-layer time-continuous models constructed via Hard-tanh and ReLU activations. IV) In contrast to static deep networks (Fig. 1), we observed that in multi-layer continuous-time networks implemented using tanh and sigmoid (Fig. 2D), the trajectory length does not increase with depth. V.In conclusion, we found that the breadth, depth, weight and bias distribution variance, and activations of a model all affect the trajectory length in TC models. In Fig. 3, we provided this in a more methodical manner.

IV) VI) A network's breadth and trajectory length increase linearly (Fig. 3B; note the logarithmic growth of the curves on the log-scale Yaxis). VII) As the variance increases, the growth is noticeably quicker (Fig. 3C). VIII) When choosing an ODE solver, trajectory length is hesitant (Fig. 3A). IX) The complex patterns investigated by the TC system are varied by activation functions, with ReLU and Hard-tanh networks exhibiting higher levels of complexity for LTCs. The existence of recurrent linkages among the cells in each layer is a major contributing factor. Computational Depth (L) definition. For a single concealed layer of L is the average number of integration steps the solver takes for each incoming input in a time-continuous network, where f example. Keep in mind that we define the overall depth for a f with n layers as n × L. We have developed lower constraints for the trajectory length expansion of continuous-time networks as a result of these results.

Theorem 4. Growth in Trajectory Length Limitations for CT-RNNs and Neural ODEs. Consider a neural ODE represented by dx/dt = fn,k(x(t),I(t), θ) with θ = {W, b}, and a CT-RNN represented by dx(t) dt = −x(t) τ + fn,k(x(t),I(t), θ) with θ = {W, b, τ}. f is randomly weighted using Hard-tanh activations. Let I(t) be a 2D input trajectory, with L = number of solver-steps, and its progressive points (i.e., I(t + δt)) having a perpendicular component to I(t) for all δt. Then, for Neural ODE and CT-RNNs, respectively, we define the projection of the scores of the first two principal components of the hidden states over one another as the 2D latent trajectory space of a layer d, z (d) (I(t)) = z (d)(t). This gives us:

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

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

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, dz/dt(d+1) , 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 dz/dt(d+1) ⊥ , and therefore we have the assumption on input I(t).

Theorem 5.
The fifth theorem is the growth rate of the trajectory length of LTC. Let's Determine an LTC using θ = {W, b, τ, A} in Eq. 1. As in Theorem 4, we get the following with the same restrictions on f and I(t):

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

The Appendix contains the proof. In a nutshell, we establish the recurrence between the components of the right-hand side of LTC and the norm of the hidden state gradients independently, gradually increasing the constraint.

Discussion of the theoretical bounds
The lower bound of LTC, which is displayed in Eq. 9, supports this. IV) The lower bound of LTC likewise shows the trajectory length's linear development with width, k, confirming the findings in 3B. V) A longer trajectory length of LTC networks in the experiments of Section 5 is justified by the computed lower bound for neural ODEs, CT-RNNs, and LTCs, given the computational depth of the models L in Table 2 for Hard-tanh activations. The expressive capability of LTCs is then evaluated using a number of real-world time-series prediction challenges. I) As anticipated, the bound for neural ODEs is very similar to that of a n layer static deep network, L., except for the exponential dependencies to the number of solver-steps. II) The base of the exponent indicates that the CT-RNNs have a shorter trajectory length than neural ODEs. Our investigations, which are shown in Figs. 2 and 3, are consistently consistent with these results. III) The trajectory length of LTC grows more quickly than linearly as a function of weight distribution variation, as seen in Figs. 2B and 3C.

6.EXPERIMENTAL EVALUATION

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**

6.2 Dataset of person activities. We employ the "Human Activity" dataset in two different frameworks, as detailed in Rubanova, Chen, and Duvenaud (2019). The dataset includes 6554 human activity sequences (such as sitting, walking, and laying) with a 211 ms duration. To assess the performance of the models, we created two experimental frameworks. The input representations in the First Setting remain the same, and the baselines are the previously mentioned models (details in Appendix). Table 4 demonstrates how LTCs perform significantly

better than other models, but especially CTRNNs and neural ODEs. With the exception of having a state damping factor τ, the CT-RNN architecture is identical to the ODE-RNN described in Runova, Chen, and Duvenaud (2019). In the second setting, we meticulously arrange the To provide a fair comparison between LTCs and a more varied collection of RNN variations included in (Rubanova, Chen, and Duvenaud 2019), the experiment should be adjusted to mirror the changes made by (Rubanova, Chen, and Duvenaud 2019) (See supplements). LTCs perform better than other models by a significant margin. Table 5 provides a summary of the findings.

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: Runova, Chen, and Duvenaud (2019) are the direct sources of accuracy for methods denoted by ∗. According to Rubanova, Chen, and Duvenaud (2019), RNN Δt = conventional RNN + input delays. According to Mozer, Kazakov, and Lindsey (2017), RNN-Decay is RNN with exponential decay on the hidden states. According to Che et al. (2018), GRU-D is equal to the gated recurrent unit plus exponential decay plus input imputation. (Rubanova, Chen, and Duvenaud 2019) Denc. = RNN encoder. (Rubanova, Chen, and Duvenaud 2019) Cenc = ODE encoder. n = 5.

7.RELATED WORKS

time-continuous models. TC networks are now more popular than ever before. This is because a number of advantages have been demonstrated, including improved continuous time-series modelling, memory, parameter efficiency, and adaptive calculations (Chen et al. 2018). Numerous alternative methods have attempted to further characterise neural ODEs (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), employ neural ODEs in particular contexts (Rubanova, Chen, and Duvenaud 2019; Lechner et al. 2019), and enhance and stabilise the adjoint method (Gholami, Keutzer, and Biros 2019). In order to enhance the expressivity and performance of neural ODEs, we examined their expressive capacity in this study and suggested a novel ODE model. Numerous recent studies have attempted to address issues like why specific architectural and deeper networks function better and where the approximation capabilities of shallow and deep networks differ. (Montufar et al. 2014).

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**

Eldan and Shamir (2016) demonstrated that there is a class of radial functions that smaller networks are unable to produce, Bengio (2013) proposed counting the number of linear regions of neural networks as a measure of expressivity, and Poole et al. (2016) investigated the exponential expressivity of neural networks by transient chaos. These techniques are convincing, but they reduce bound expressivity in a manner comparable to that of (Serra, Tjandraatmadja, and Ramalingam 2017; Gabrie et al. 2018; Hanin ´ and Rolnick 2018, 2019; Lee, Alvarez-Melis, and Jaakkola 2019) by being restricted to certain weight configurations of a given network. An interconnected idea that measures a static network's expressiveness by trajectory length was presented by Raghu et al. (2017). We produced lowerbound estimates by extending their expressivity analysis to time-continuous networks.

8 CONCLUSIONS, SCOPE AND LIMITATIONS

We looked into applying a new class of time-continuous neural network models produced by combining unique nonlinear weight combinations with linear ODE neurones. We demonstrated that arbitrary variable and fixed step ODE solvers may successfully implement them, and be taught over time by backpropagation. In comparison to conventional and contemporary deep learning models, we showed their better expressivity, constrained and stable dynamics, and outperforming performance in supervised learning time series prediction challenges enduring reliance. When trained by gradient descent, LTCs exhibit the vanishing gradient phenomena, which is similar to many other variations of time continuous models (Pascanu, Mikolov, and Bengio 2013; Lechner and Hasani 2020). The model would not be the most apparent option for learning, despite its potential on a range of time-series prediction problems long-term ties in their existing structure selection of an ODE solver. The numerical implementation technique of time-continuous models has a significant impact on their performance (Hasani 2020). Even while LTCs work well with sophisticated variable-step solvers and the Fused fixed-step solver shown here, using off-the-shelf explicit Euler techniques has a significant impact on their performance. Memory and Time. Comparing neural ODEs to more complex models like LTCs, they are incredibly quick. However, they are not expressive. The expressive potential of TC models is greatly increased by our suggested model in its current form, but at the cost of increased time and memory complexity, which needs more research. causality. Causal structures are fundamental to models defined by time-continuous differential equation semantics (Scholkopf 2019), particularly those that have recurring processes to translate past experiences into predictions for the future. Given that the semantics of performant recurrent models like LTCs approximate dynamic causal models (Friston, Harrison, and Penny 2003) with a bilinear dynamical system approximation (Penny, Ghahramani, and Friston 2005), investigating the causality of these models would be an attractive avenue for future study. Therefore, controlling robots in continuous-time observation and action spaces, where causal structures like LTCs might aid in enhancing cognition, would be a logical application domain (Lechner et al. 2020a).

9.ACKNOWLEDGMENTS

Boeing provides some help for R.H. and D.R. Grant No. 783163 (iDev40) from the Horizon-2020 ECSEL Project provided some funding for R.H. and R.G. The Austrian Science Fund (FWF) provided M.L. with partial funding under grant Z211-N23 (Wittgenstein Award). A.A. is supported by the National Science Foundation (NSF) Graduate Research Fellowship Program. This study is based in part on R.H.'s doctoral dissertation.

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.
Gabrie, M.; Manoel, A.; Luneau, C.; Macris, N.; Krzakala, ´ F.; Zdeborova, L.; et al. 2018. Entropy and mutual infor- ´ mation 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 Universitat 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.