Liquid Time-Constant Networks

Jyotirmaya Shivottam
23226001
CS660/CS460 - Machine Learning, Spring 2024
Subhankar Mishra Lab
National Institute of Science Education and Research
An OCC of Homi Bhabha National Institute
Bhubaneswar, Odisha, India

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: $\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$, with parameters θ and A.
- Then, the Liquid-Time Constant (LTC) Network models the following continous-time dynamical system:

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

- Here, τ defines the system's **time-constant**. LTCs represent ODEs that **vary their time-constants in an input-dependent manner** $\rightarrow$ "**liquid**".

Motivation

- Instead of modeling implicit nonlinearities, LTCs model linear first-order dynamical systems modulated via nonlinear interlinked gates.
- Inspired by the computational models of neural dynamics in small species.
- The LTC update is also similar to that of bilinear-approximated Dynamic Causal Models (DCMs), that are useful in learning on complex fMRI time-series signals.
- The expressivity of the LTC formulation can be studied via trajectory length analysis.
- **The goal is to capture complex non-linear interactions in potentially irregular time-series data.**

New Semi-implicit Fused ODE Solver

Algorithm LTC update by fused ODE Solver

Parameters: $\theta = \{\tau^{(N\times1)} = \text{time-constant}, \gamma^{(M\times N)} = \text{weights}, \gamma_r^{(N\times N)} = \text{recurrent weights}, \mu^{(N\times1)} = \text{biases}\}$, $A^{(N\times1)} = \text{bias vector}$, $L = \text{Number of unfolding steps}$, $\Delta t = \text{step size}$, $N = \text{Number of neurons}$,
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 $i = 1 \dots L$ **do**
$\mathbf{x}_{t+\Delta t} = \text{FusedStep}(\mathbf{x}(t), \mathbf{I}(t), \Delta t, \theta)$
end for
return $\mathbf{x}_{t+\Delta t}$

Recursively Folding Solver Output and Training via BPTT

Algorithm Training LTC by BPTT - Vanilla SGD

Inputs: Dataset of traces $[I(t), y(t)]$ of length T, RNNcell $= f(I, x)$
Parameter: Loss func $L(\theta)$, initial param θ_0, learning rate α, Output w $= W_{out}$, and bias $= b_{out}$
for $i = 1 \dots$ number of training steps **do**
$(I_b, y_b) =$ Sample training batch, $\quad x := x_{t_0} \sim p(x_{t_0})$
for $j = 1 \dots T$ **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

Complexity comparison for a single layer NN

	Vanilla BPTT	**Adjoint**
Time	$O(L \times T \times 2)$	$O((L_f + L_b) \times T)$
Memory	$O(L \times T)$	$\mathbf{O(1)}$
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.

Summary of Main Contributions

- A new paradigm in continuous-time neural networks, effective in learning on irregularly-sampled data.
- LTC presents a novel approach for forward and backward passes, that balances accuracy and computation time.
- Trajectory length analysis shows that LTCs are significantly more expressive as compared to Neural Ordinary Differential Equations (NODE) and established sequence models.
- LTC time-constant and neural states are provably stable for unbounded inputs.
- LTCs are **universal approximators**.
- Architectures created using LTCs **can greatly reduce model size, while aiding explainability**.
- **LTCs can vary their behavior even post-training**.

Expressivity Measure – Trajectory Length

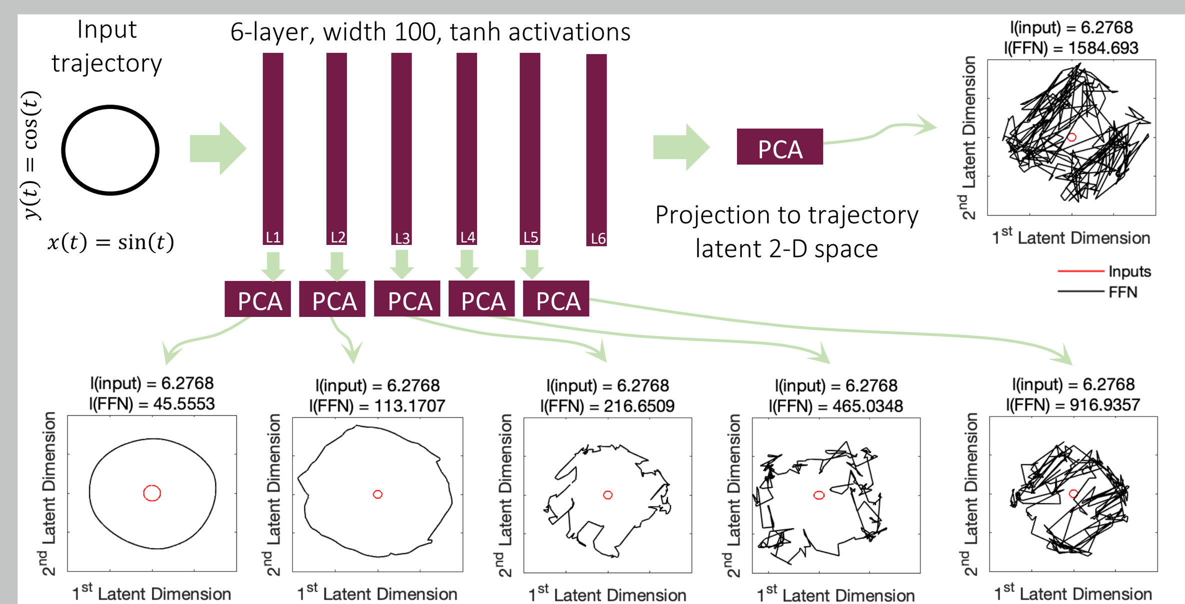

Figure: LTC's trajectory latent space becomes more complex with depth

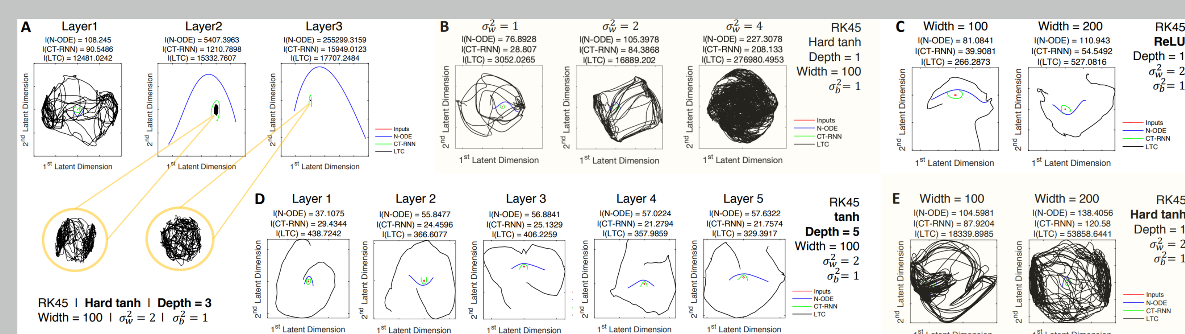

Figure: Trajectory latent w.r.t. different activations

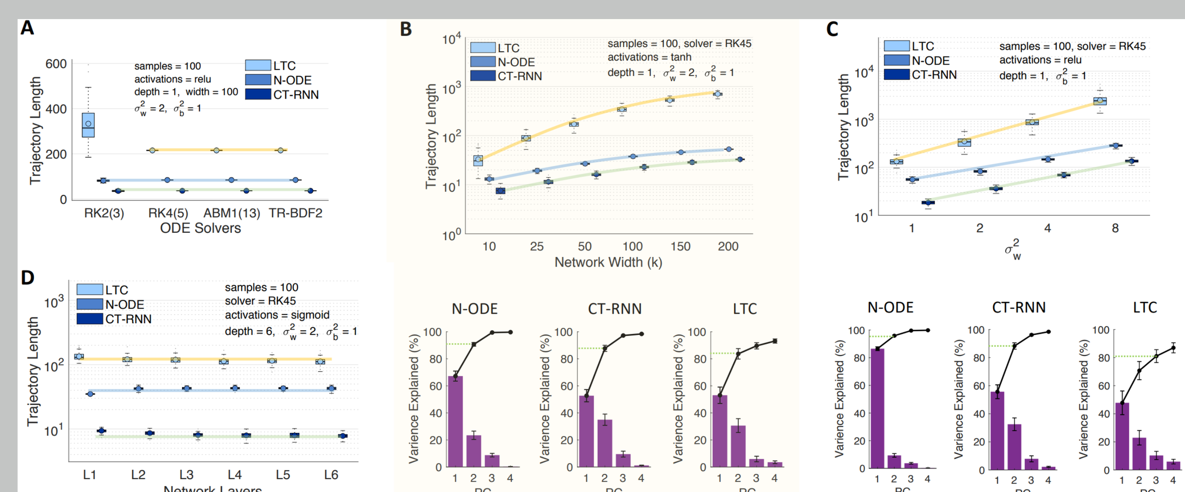

Figure: Trajectory latent w.r.t. different solvers vis-à-vis baselines

Expressivity Measure – Trajectory Length Lower Bounds

Expressivity
Trajectory length lower bound

Weight scale · Bias scale · Width · Depth · Number of discretization steps

$$\text{Neural ODE:}\quad \mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{\sigma_w\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t))$$

$$\text{CT-RNN:}\quad \mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{(\sigma_w-\sigma_b)\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t))$$

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

System's dynamic time-scale

Practical Application — Lane Following - Network size comparison

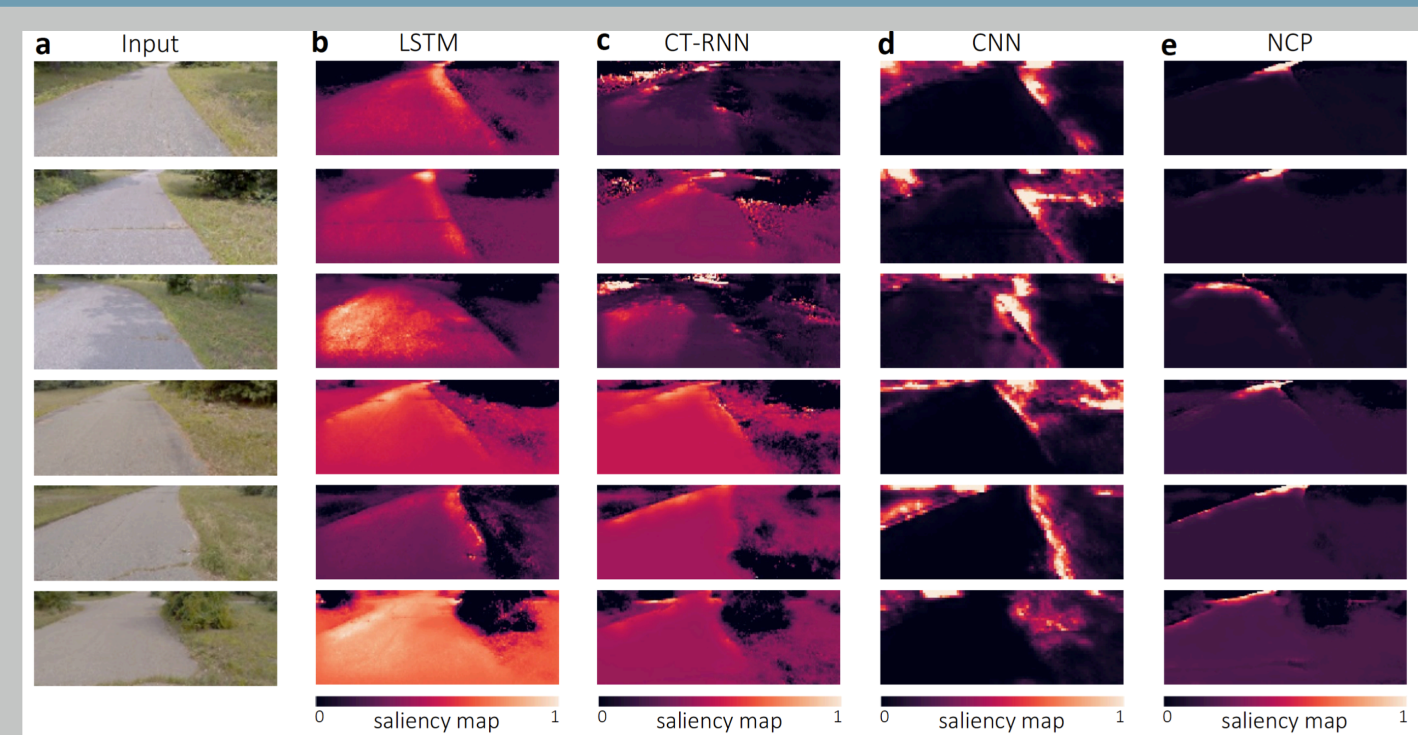

Figure: Saliency Maps - Where each network learns to attend while driving

Table: Network size comparison

Model	CNN parameters	RNN neurons	RNN synapses	RNN trainable parameters
CNN	5,068,900	-	-	-
CT-RNN	79,420	64	6,112	6,273
LSTM	79,420	64	24,640	24,897
NCP	79,420	19	253	1,065

Limitations

- Vanishing gradient phenomenon limiting applicability to learning long-term dependencies.
- Performance is tied to ODE solver used.
- Highly expressive but at an added time and memory cost.

References & Further Reading

- Liquid Time Constant Networks, Hasani et al, 2020, 10.48550/arXiv.2006.04439
- Liquid Time Constant Networks - Simons Institute Presentation: `https://youtu.be/watch?v=9AxYrmUlA0I`
- Neural circuit policies enabling auditable autonomy, Lechner et al, 2020, 10.1038/s42256-020-00237-3
- Closed-form continuous-time neural networks, Hasani et al, 2022, 10.1038/s42256-022-00556-7
- Neural ordinary differential equations, Chen et al, 2018, 10.48550/arXiv.1806.07366
- On the Expressive Power of Deep Neural Networks, Raghu et al, 2016, 10.48550/arXiv.1606.05336