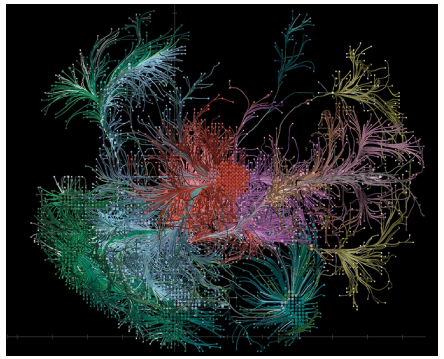
Liquid Neural Networks

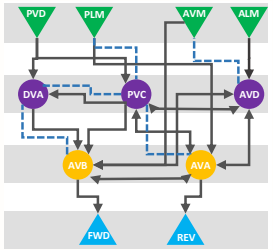
Ramin Hasani
CSAIL MIT

Jun 12th, 2024

Nervous Systems
(Image: Allen Institute for Brain Science)



Neural Circuits



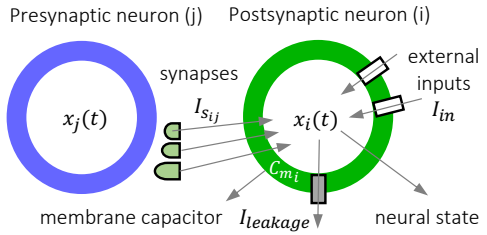
ICML 2020
ICRA 2019
NeurIPS Deep RL Symp 2017

Liquid Neural Networks

$$d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t)$$

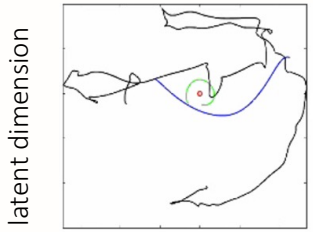
$$\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$$

Neurons & Synapses



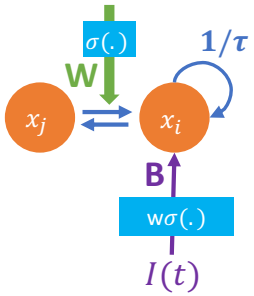
Nature Machine Intelligence 2020

Expressivity



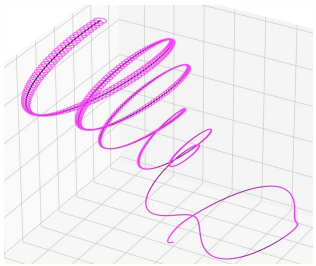
AAAI 2021

Causality



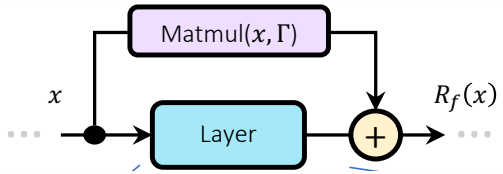
NeurIPS 2021

Robustness



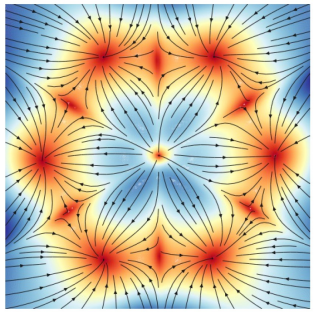
AAAI 2021 & 2022

memory



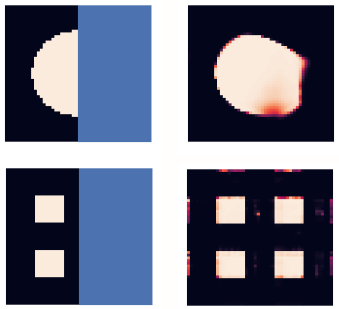
NeurIPS 2022, ICLR 2023
Nature MI 2022

Generative Modeling



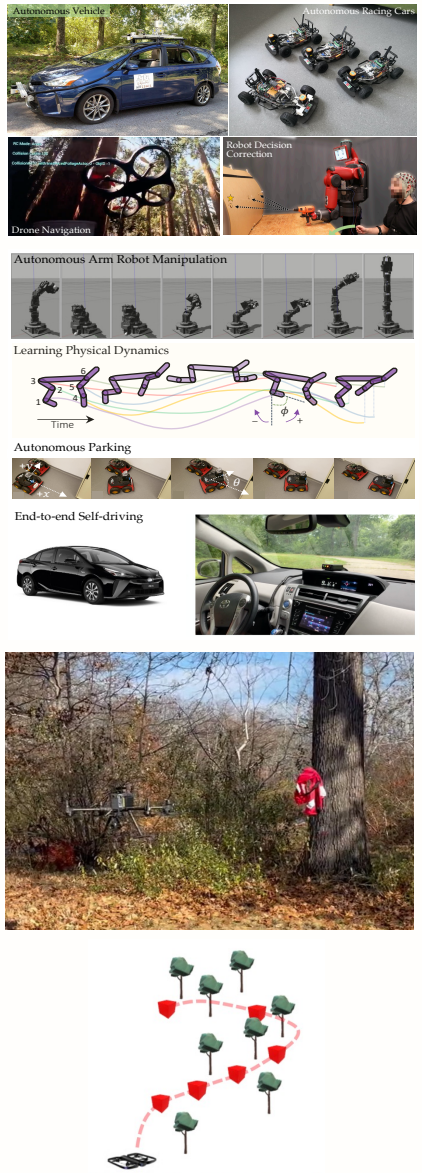
NeurIPS 2021

Extrapolation

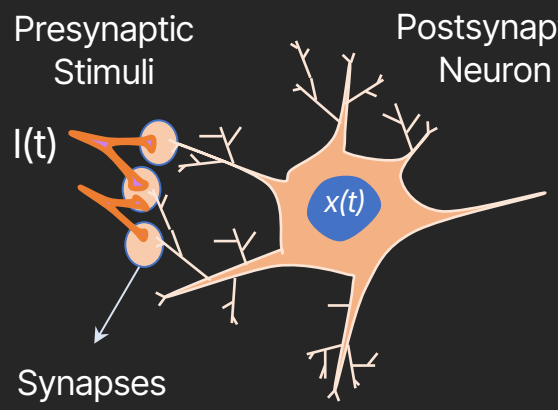


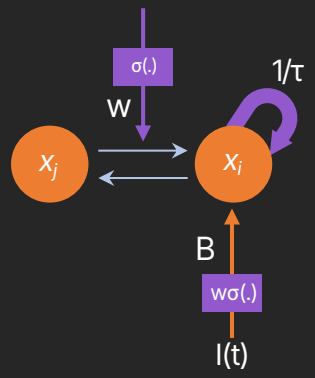
Science Robotics 2023

Mixed-horizon Decision-making



Brains

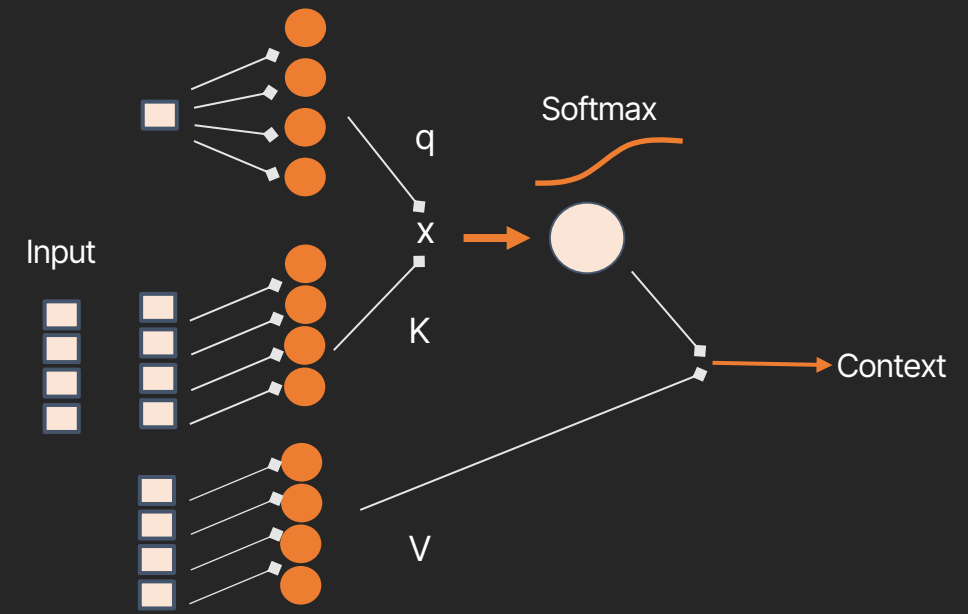


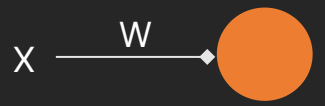


Main Operations

I. Scalar weighted operations
II. Threshold activations
III. Feedforward operations
IV. Multi-scale feedback
V. Probabilistic synapses
    I. Nonlinear functions
    II. Feedback system
    III. Information release pattern
VI. Nodes are continuous processes
VII. Governed by state-space models

Base of Gen AI: Transformers

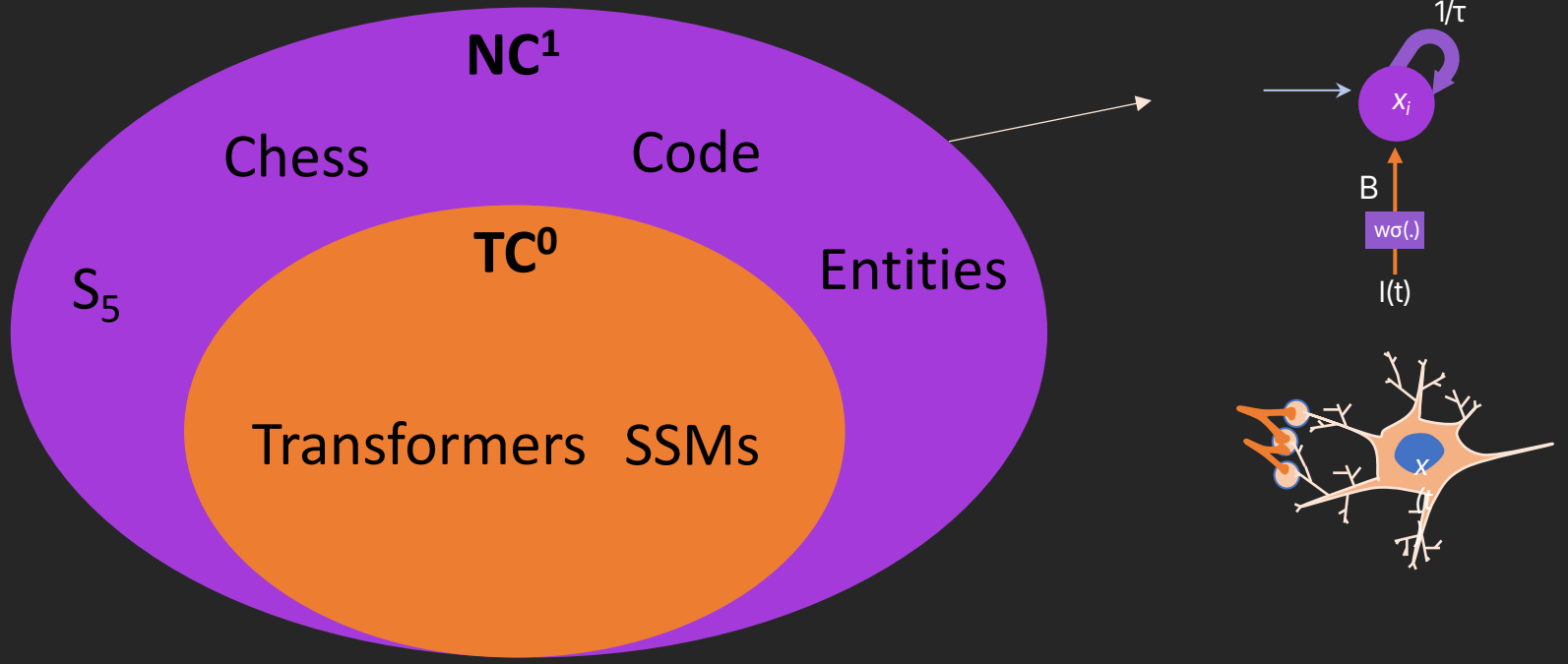




Main Operations

I. Scalar weighted operations
II. Threshold activations
III. Feedforward operations

Why bother?



NC (Nick's Class) = Problems that can be efficiently solved on a parallel Computer.
TC0 = Threshold Circuits of Depth 0
SSM = State-space model
RNN = recurrent neural network

Merrill et al. ICML 2024

Outline

- ✓ Liquid time-constant networks
- ✓ Autonomy with liquid networks
- ✓ Causality
- ✓ Closed-form Continuous-time decision-making
- ✓ Task understanding and out of distribution generalization

Outline

- ✓ **Liquid time-constant networks**
- ✓ Autonomy with liquid networks
- ✓ Causality
- ✓ Closed-form Continuous-time decision-making
- ✓ Task understanding and out of distribution generalization

Autonomy as a continuous-time dynamical system

- ✓ Practical decision making is naturally modeled a **continuous process**
- ✓ Dynamical system described by **differential equations**
- ✓ Recurrence, memory, and sparsity

What is a continuous- time/depth neural network?

$$\frac{dx}{dt} = f_{(n,k,l_{type})}(x(t), I(t), \theta)$$

Number of layers

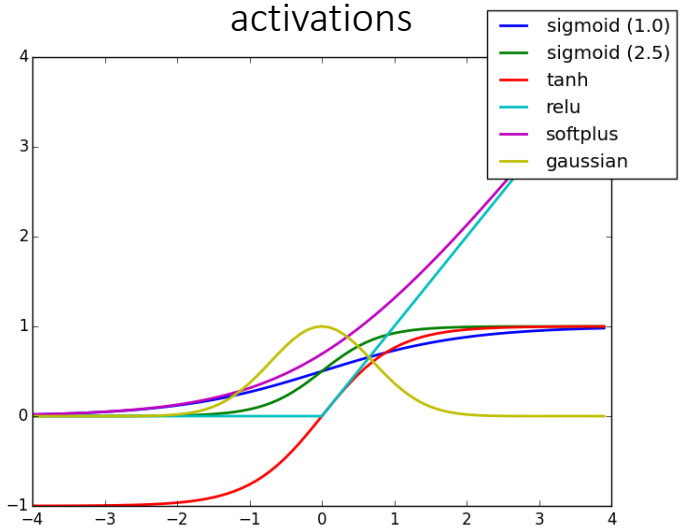
Width

Hidden state

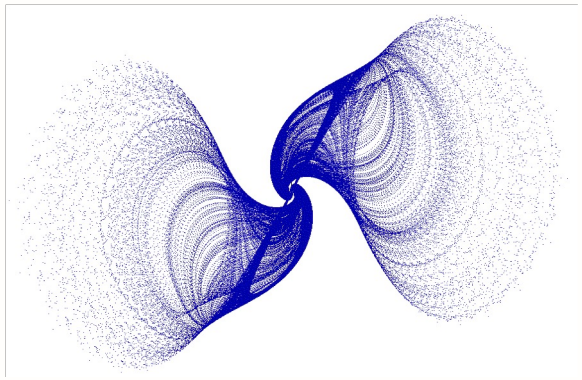
Inputs

Model parameters

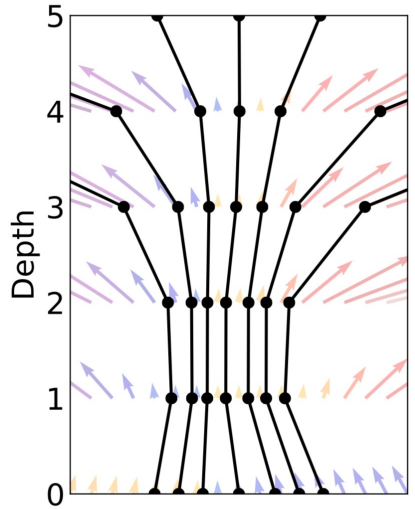
activations



Dynamical systems



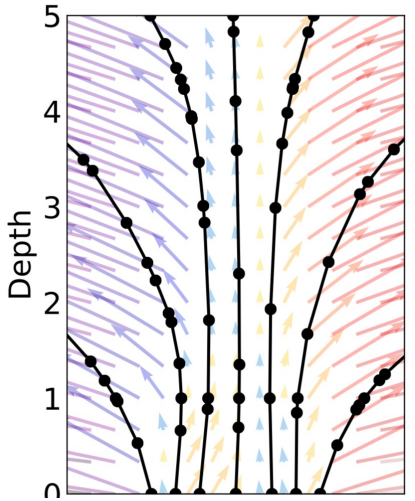
Residual Network



$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t)$$

He et al.
CVPR 2016

ODE Network



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

Chen et al.
NeurIPS 2018

Figure Credit: Chen et al. NeurIPS 2018

What is a continuous- time/depth neural network?

Standard Recurrent Neural Network (RNN)
Hopfield 1982

$$x(t+1) = f(x(t), I(t), t\,; \theta)$$

Neural ODE
Chen et al. NeurIPS, 2018

$$\frac{dx(t)}{dt} = f(x(t), I(t), t\,; \theta)$$

Continuous-time (CT) RNN
Funahashi et al. 1993

$$\frac{dx(t)}{dt} = -\frac{x(t)}{\tau} + f(x(t), I(t), t\,; \theta)$$

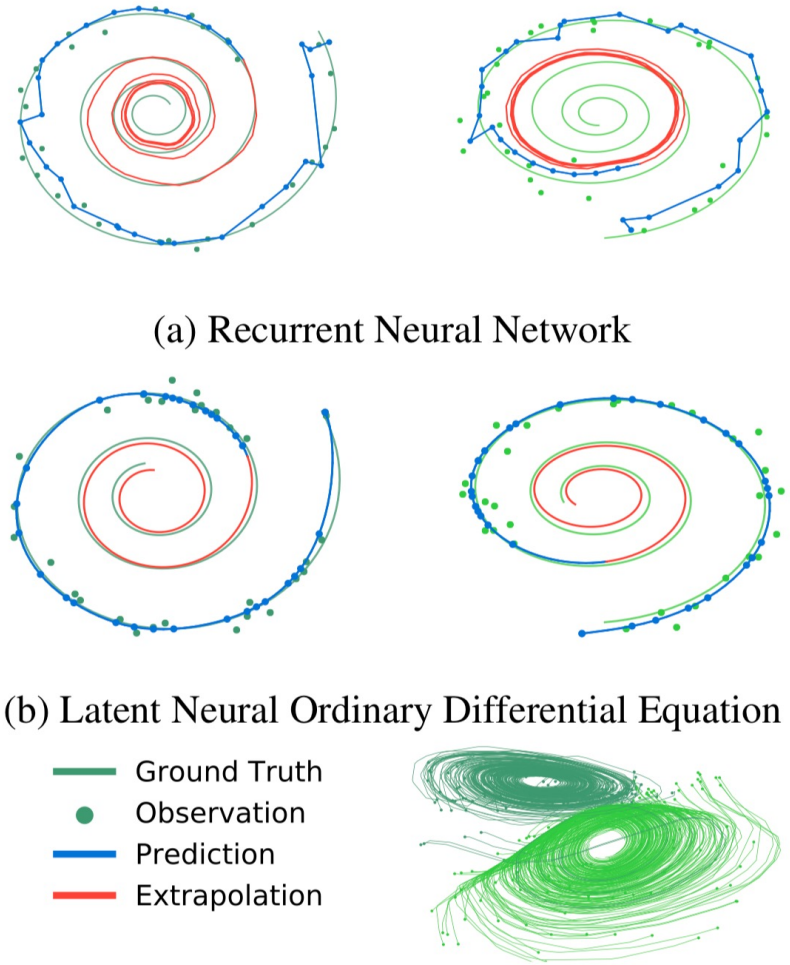


Figure Credit: Chen et al. NeurIPS 2018

Liquid Time-Constant (LTC) networks

1. Linear state-space model

$$d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t) \qquad \mathbf{S}(t) \in \mathbb{R}^M$$

2. Non-linear synapse Model

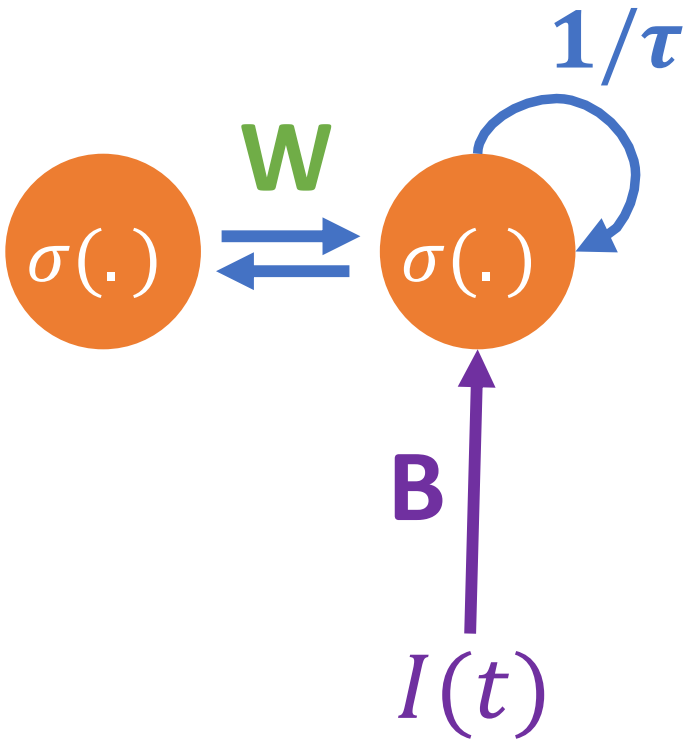
$$\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$$

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

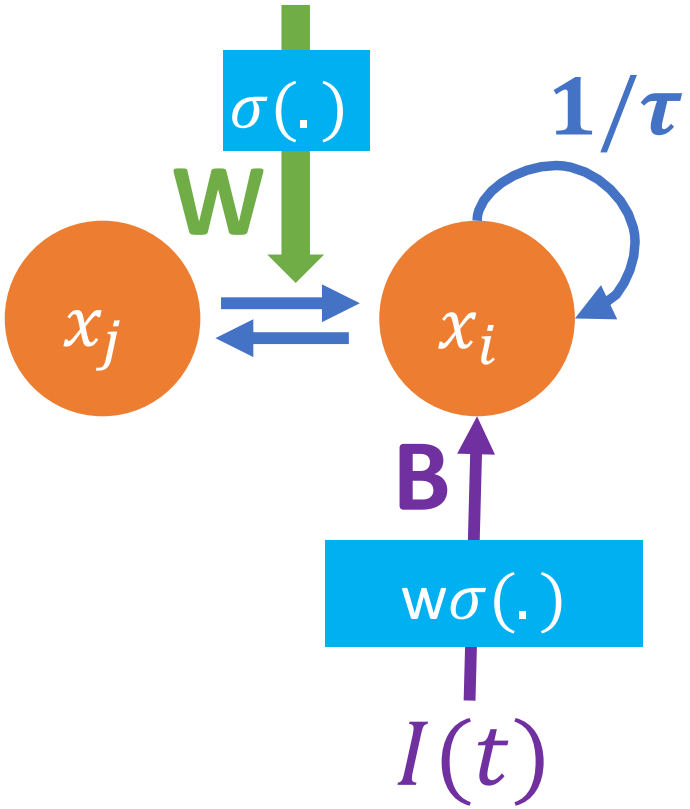
"Liquid" = variable

Liquid Time-Constant (LTC) networks

Standard Neural Nets



Liquid Neural Nets

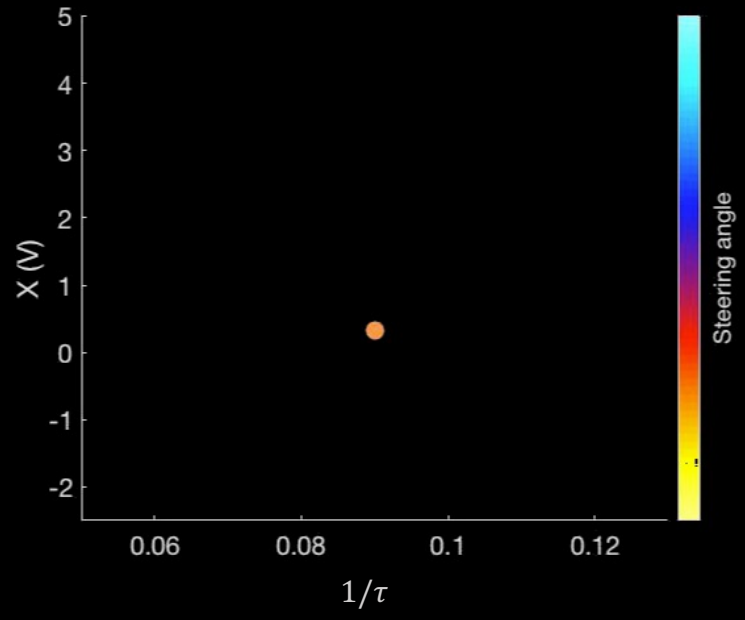


$1/\tau$ intrinsic coupling

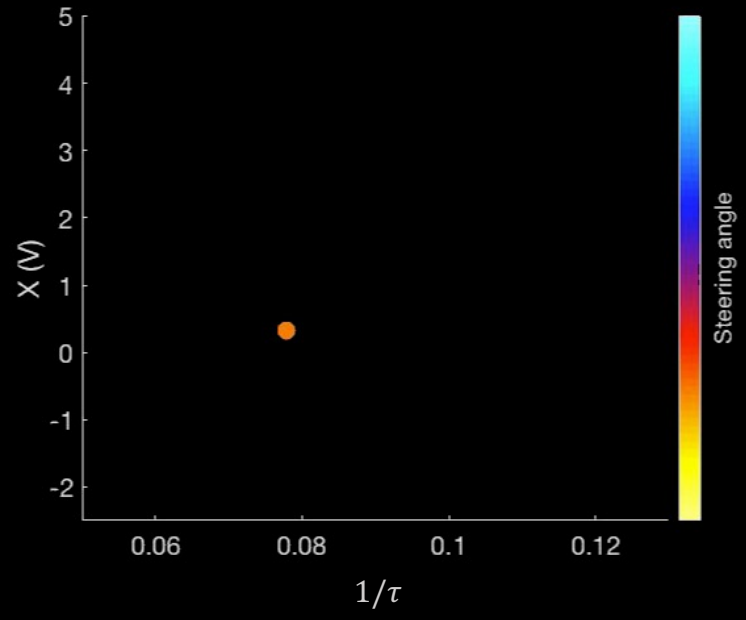
$\mathbf{W}\sigma(.)$ liquidity modulator

B input regulator

Standard Neural Network



Liquid Neural Network



LTCs have stable state and time-constant

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

System time-constant

$$\tau_{sys} = \frac{\tau}{1+\tau f(\mathbf{x}(t),\mathbf{I}(t),t,\theta)}$$

Theorem 1. Time-constant of each LTC neuron is bounded:

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

Theorem 2. The state of each LTC neuron is bounded

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

Liquid Time-Constant Networks are Universal Approximators

Theorem 3. LTCs are universal approximators

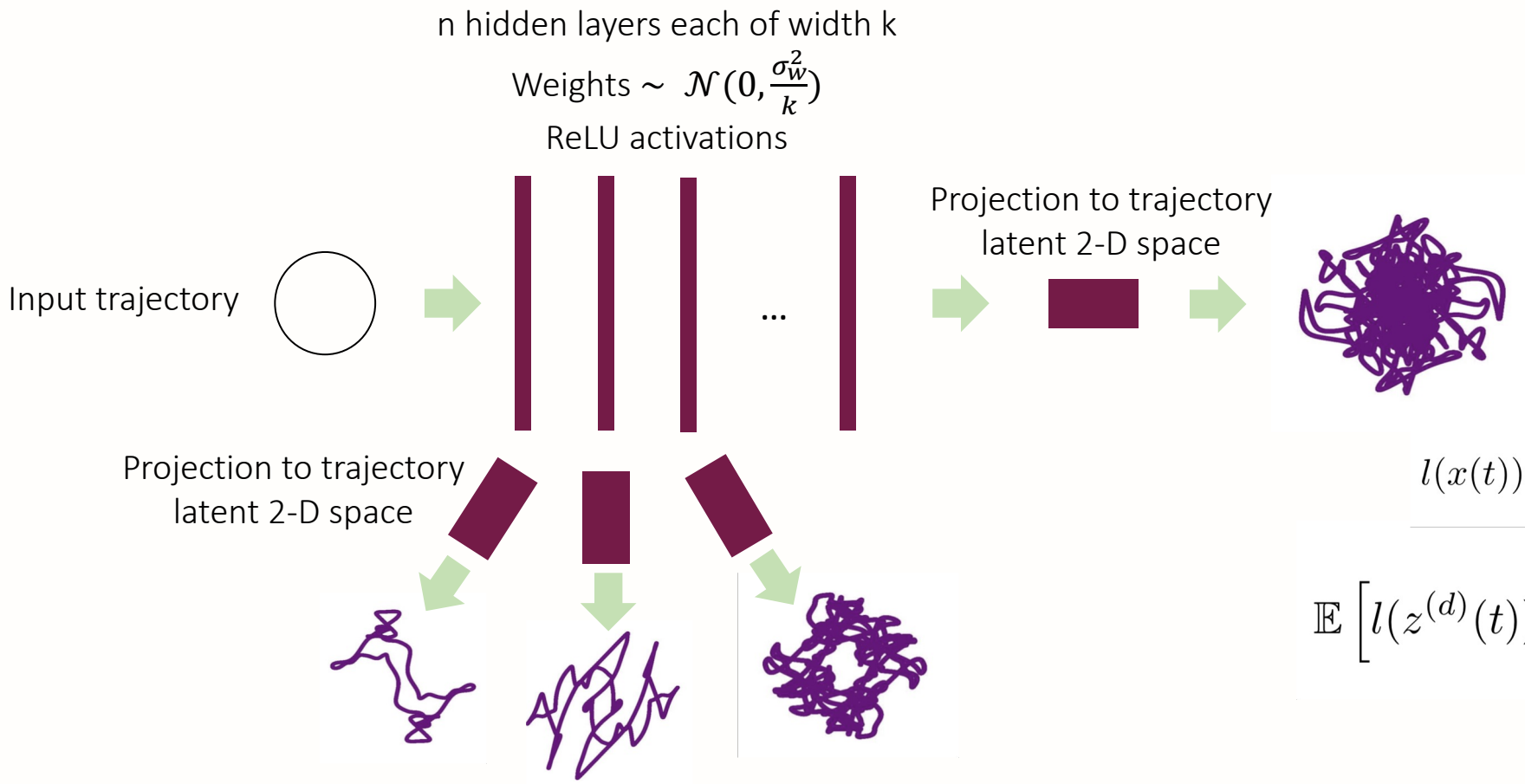
$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A$$

Expressivity

Defining a better measure

Trajectory Length as a measure of expressivity of Deep networks

[Raghu et al. ICML 2017]



$$l(x(t)) = \int_t \left|\left| \frac{dx(t)}{dt} \right|\right| dt$$ Arc-length

$$\mathbb{E}\left[l(z^{(d)}(t)) \right] \geq O\left(\frac{\sigma_w \sqrt{k}}{\sqrt{k+1}} \right)^d l(x(t))$$

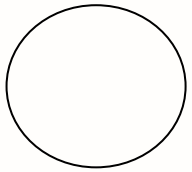
[Raghu et al. ICML 2017]

Expressivity

Trajectory length as a measure of expressivity

Let's implement the trajectory space for time-continuous models

Input trajectory



LTC
CT-RNN
Neural ODE

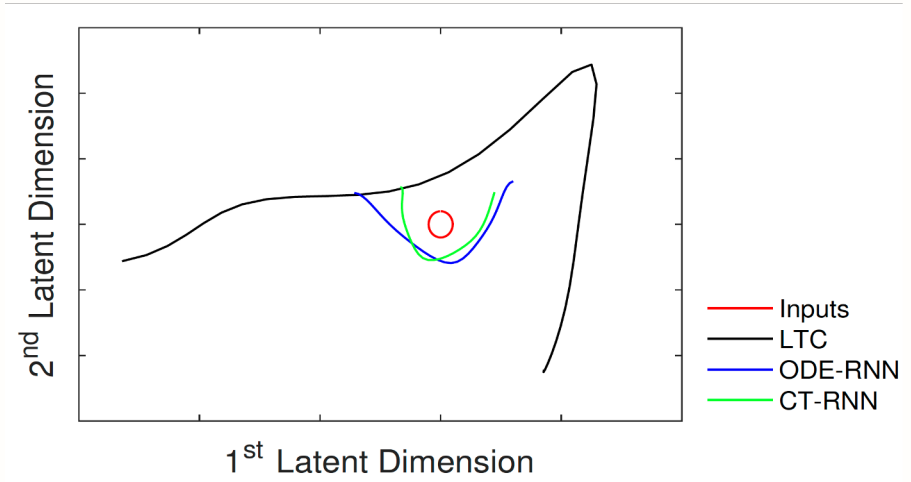
Weights ~ $\mathcal{N}(0, \frac{\sigma_w^2}{k})$

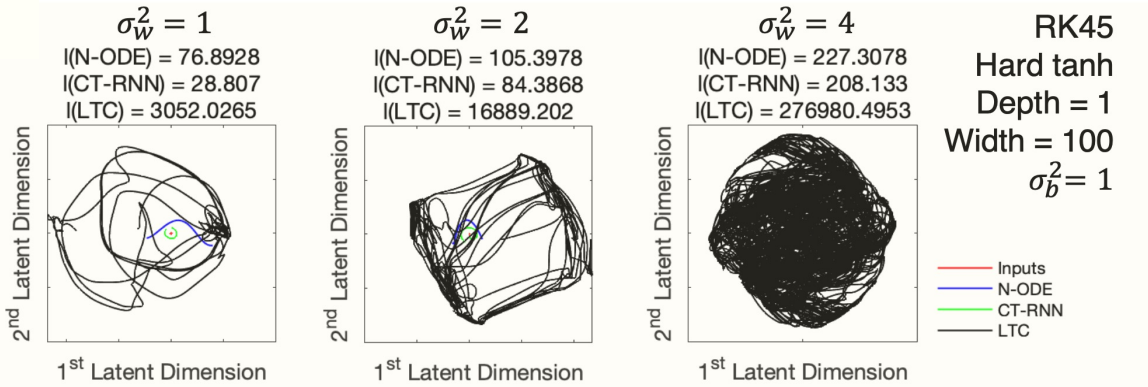
activation functions:
ReLU, tanh, logistic sigmoid

Projection to a latent trajectory 2-D space

PCA

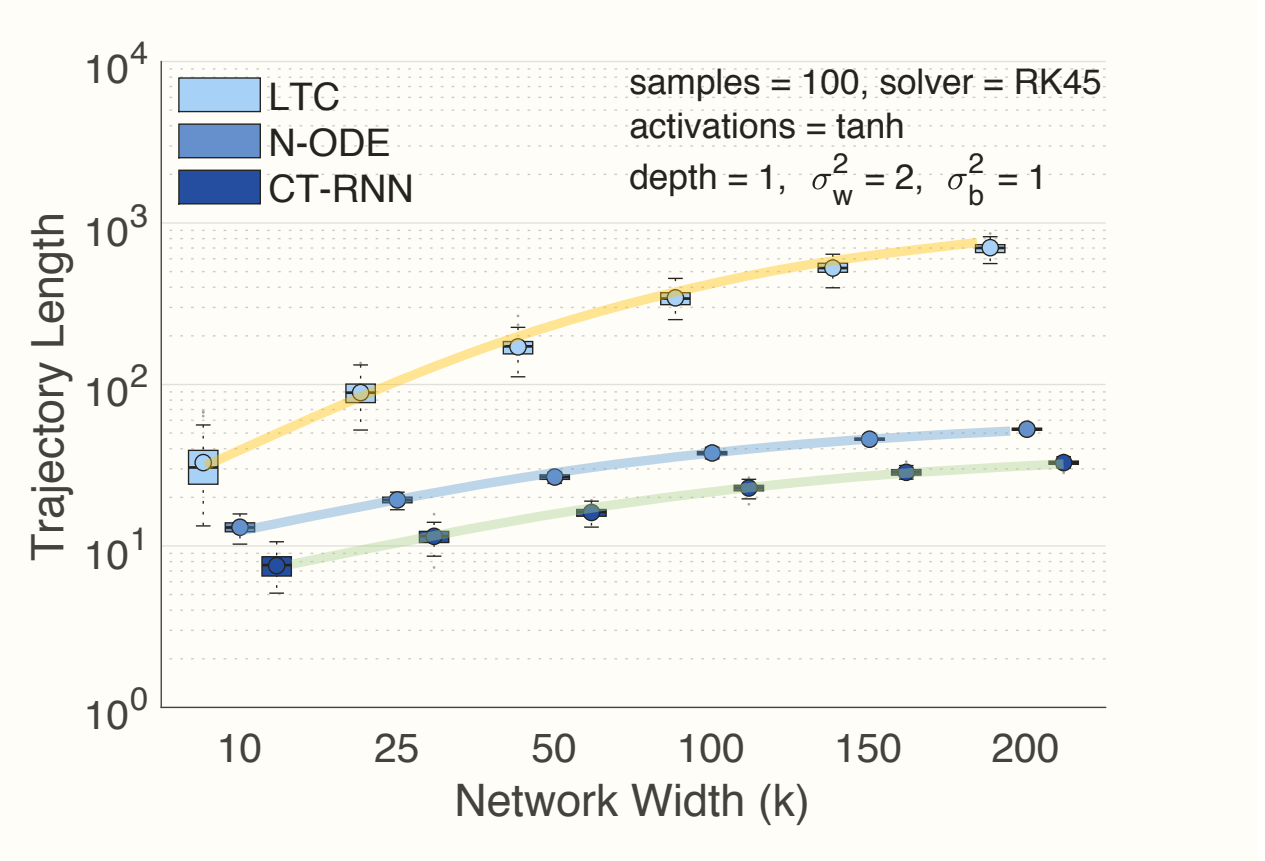
PCA = Principle Component Analysis





Expressivity

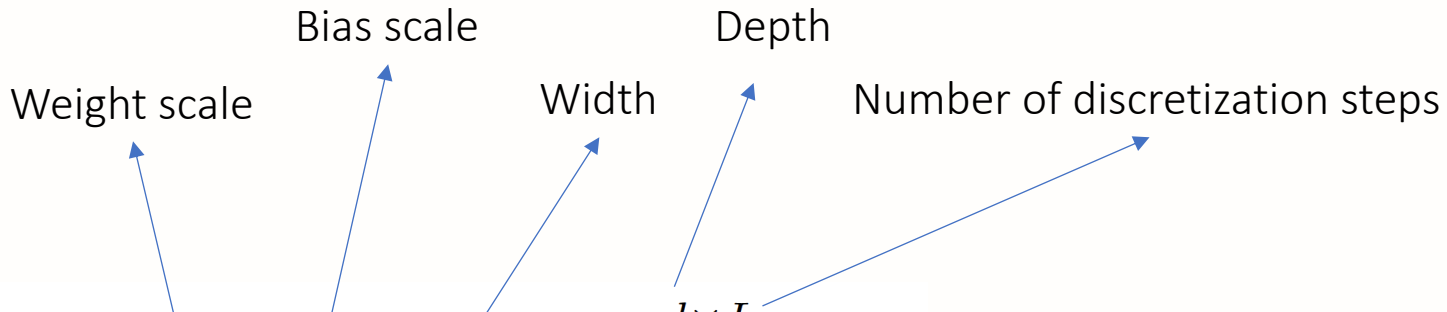
Trajectory length as a measure of expressivity



Hasani et al. AAAI 2021

Expressivity

Trajectory length lower bound



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

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

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

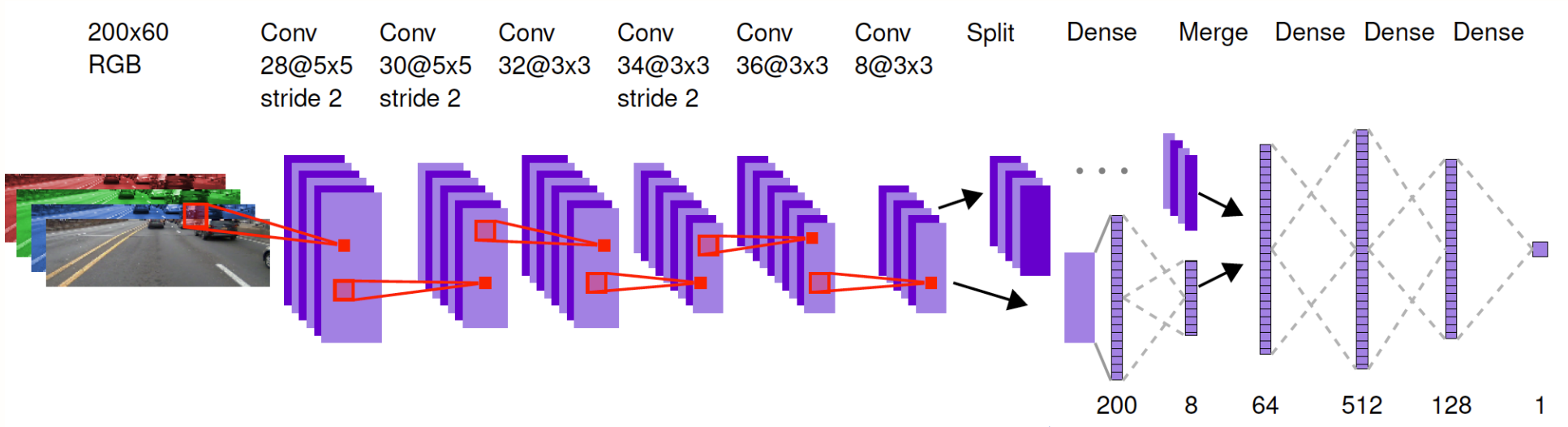
System's dynamic time-scale

Outline

- ✓ Liquid time-constant networks
- ✓ **Autonomy with liquid networks**
- ✓ Causality
- ✓ Closed-form Continuous-time decision-making
- ✓ Task understanding and out of distribution generalization

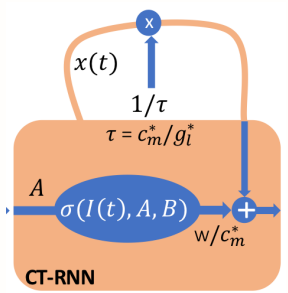
LTCs: Performance

High-fidelity autonomy by LTCs - end-to-end learning

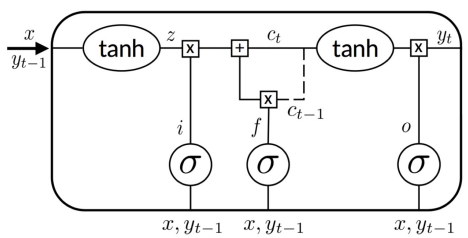


What if we replace the fully connected layers by a recurrent neural network?

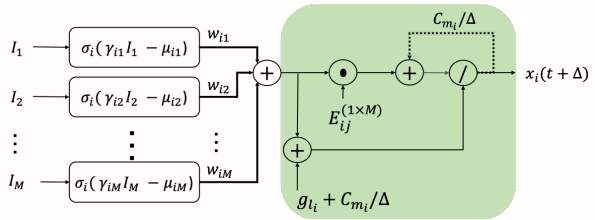
ODE-RNN



LSTMs



LTC-based Networks?



Lechner*, Hasani* et al. Nature MI, 2020

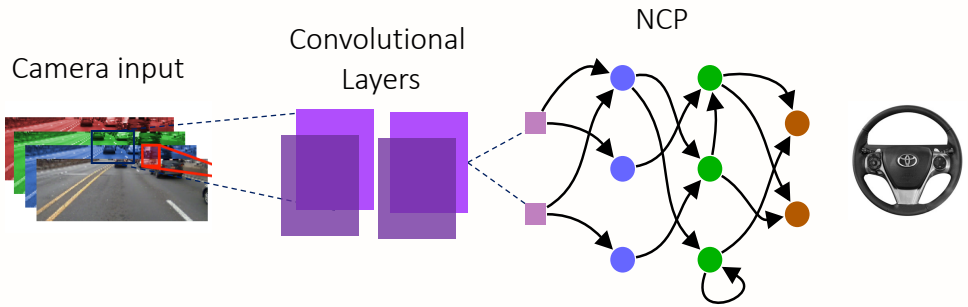
LTCs: Performance

High-fidelity autonomy by LTCs

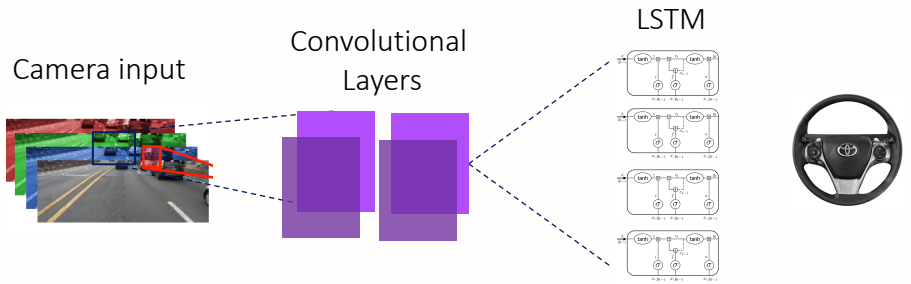
end-to-end learning of Neural Circuit Policies (NCP)

Now we compare properties of NCPs with a number of other models

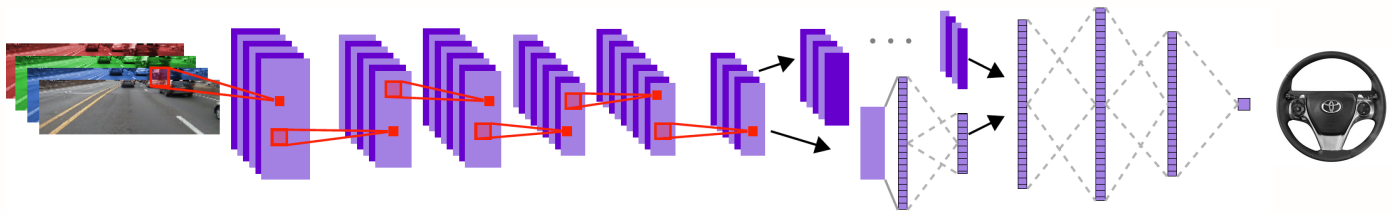
NCPs



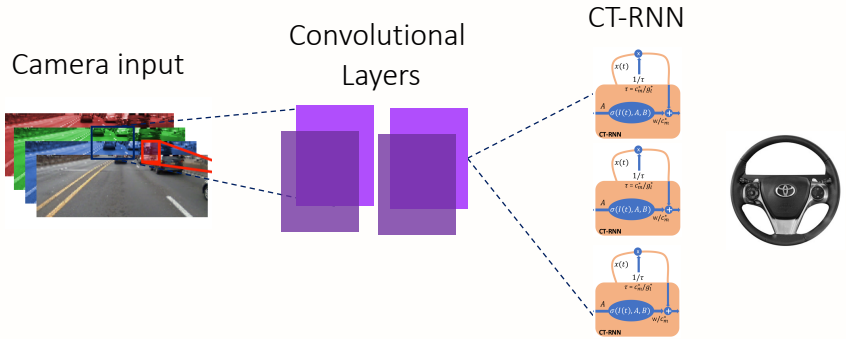
LSTMs



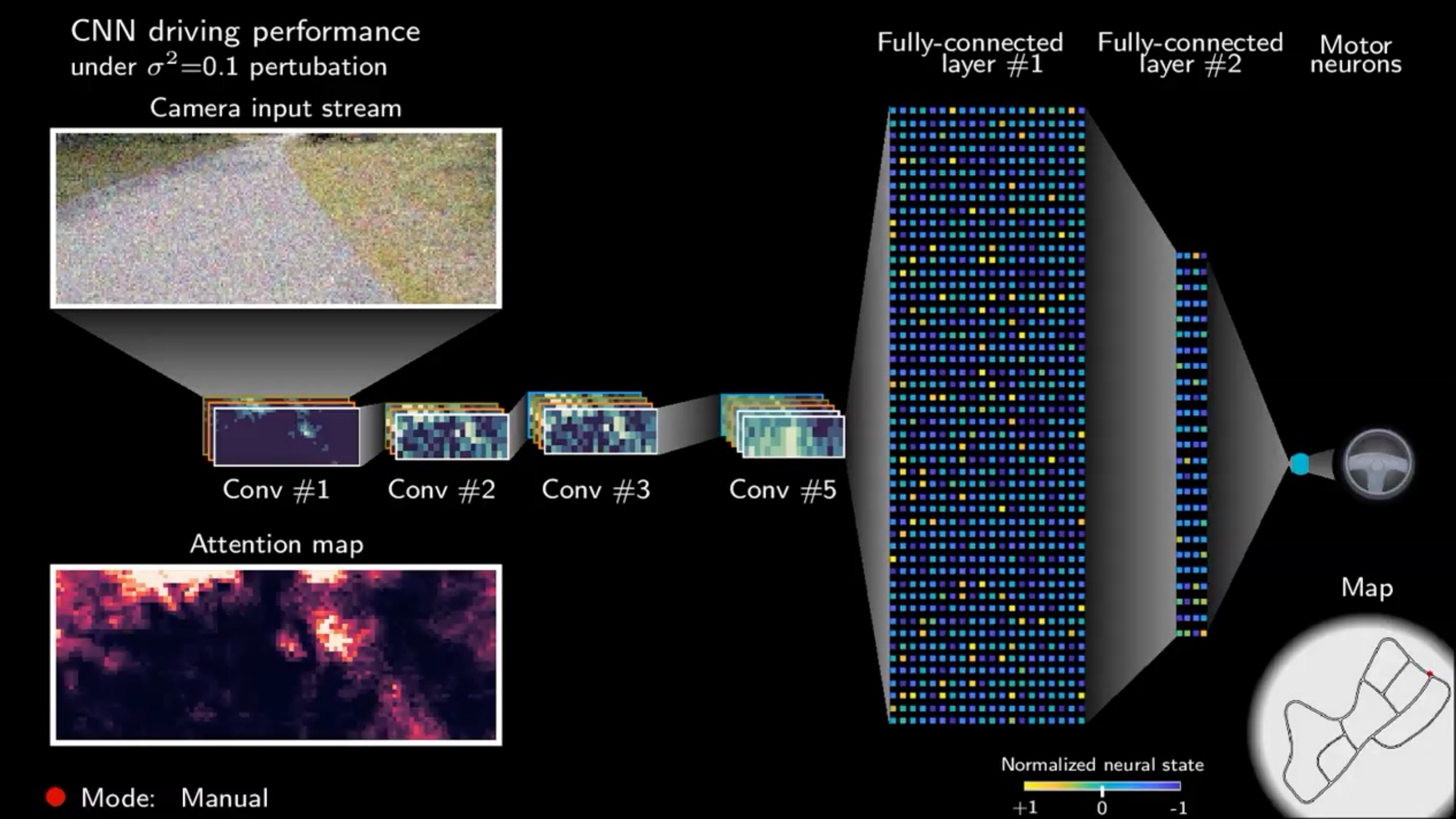
Convolutional Neural Networks (CNNs)

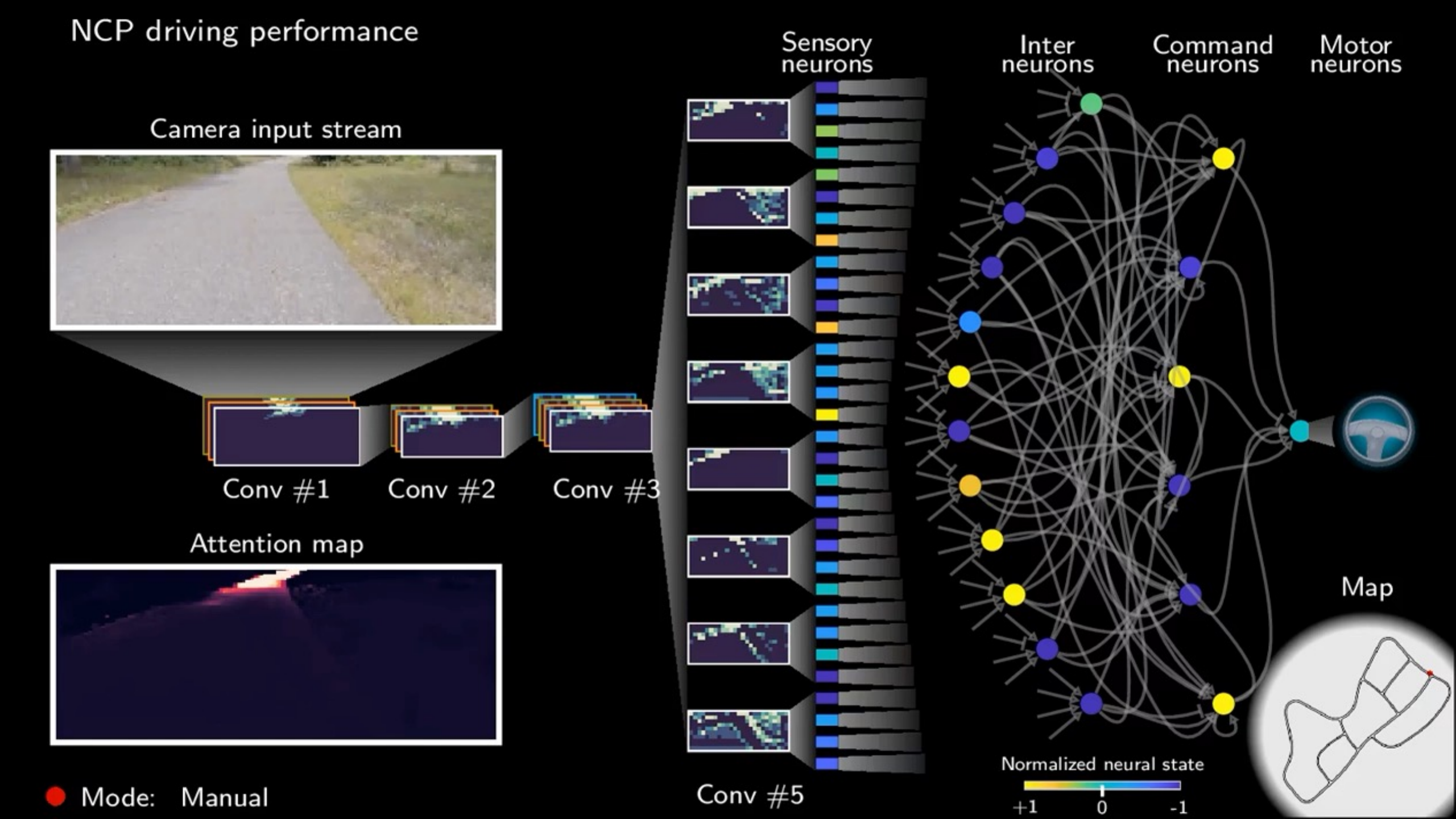


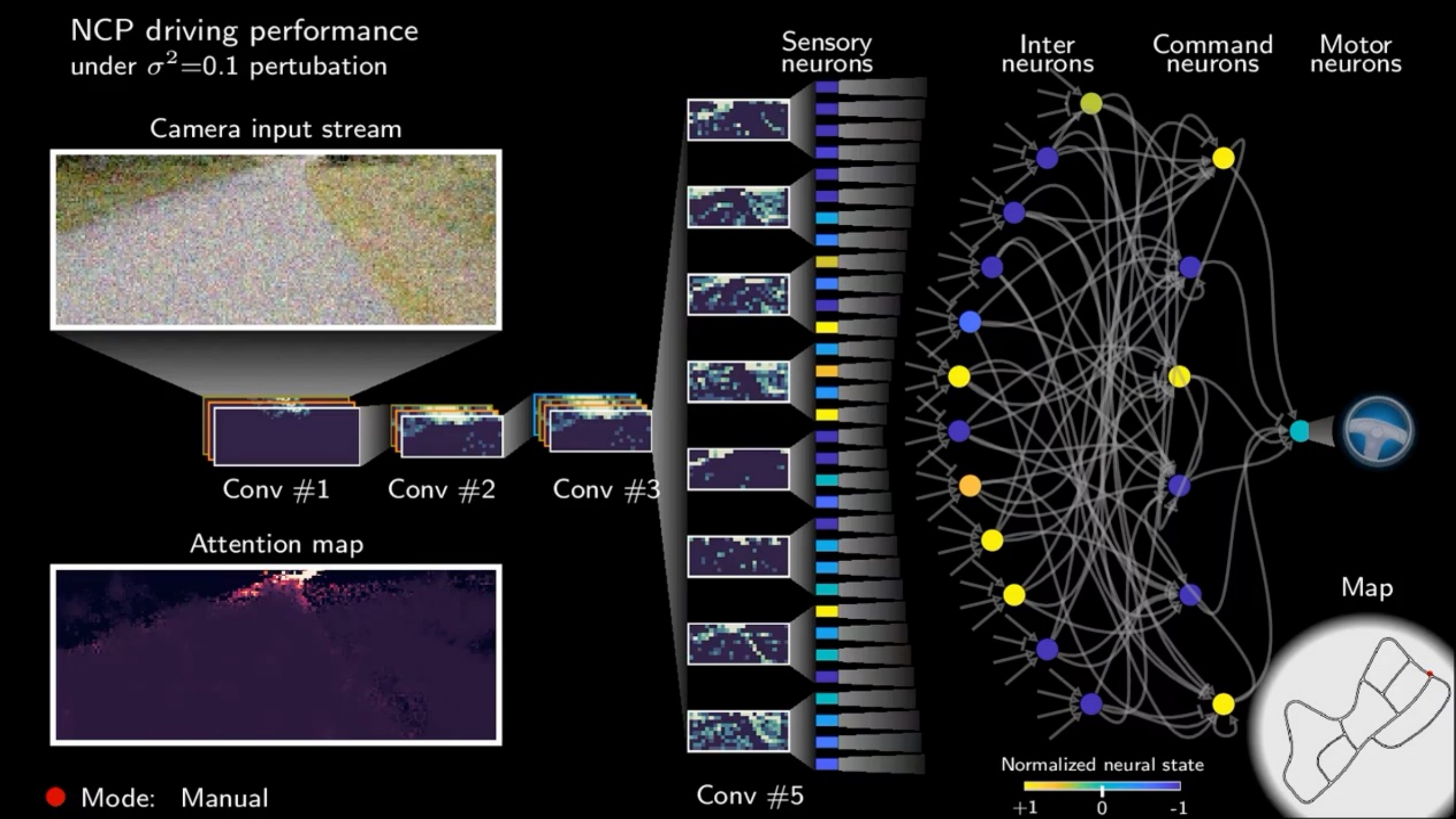
CT-RNNs



Lechner*, Hasani* et al. Nature MI, 2020


CNN driving performance
under σ^2=0.1 pertubation
Camera input stream
Attention map
Conv #1
Conv #2
Conv #3
Conv #5
Fully-connected layer #1
Fully-connected layer #2
Motor neurons
Map
Normalized neural state
+1
0
-1
Mode: Manual


NCP driving performance
Camera input stream
Conv #1
Conv #2
Conv #3
Attention map
Sensory neurons
Conv #5
Inter neurons
Command neurons
Motor neurons
Map
Normalized neural state
+1
0
-1
Mode: Manual

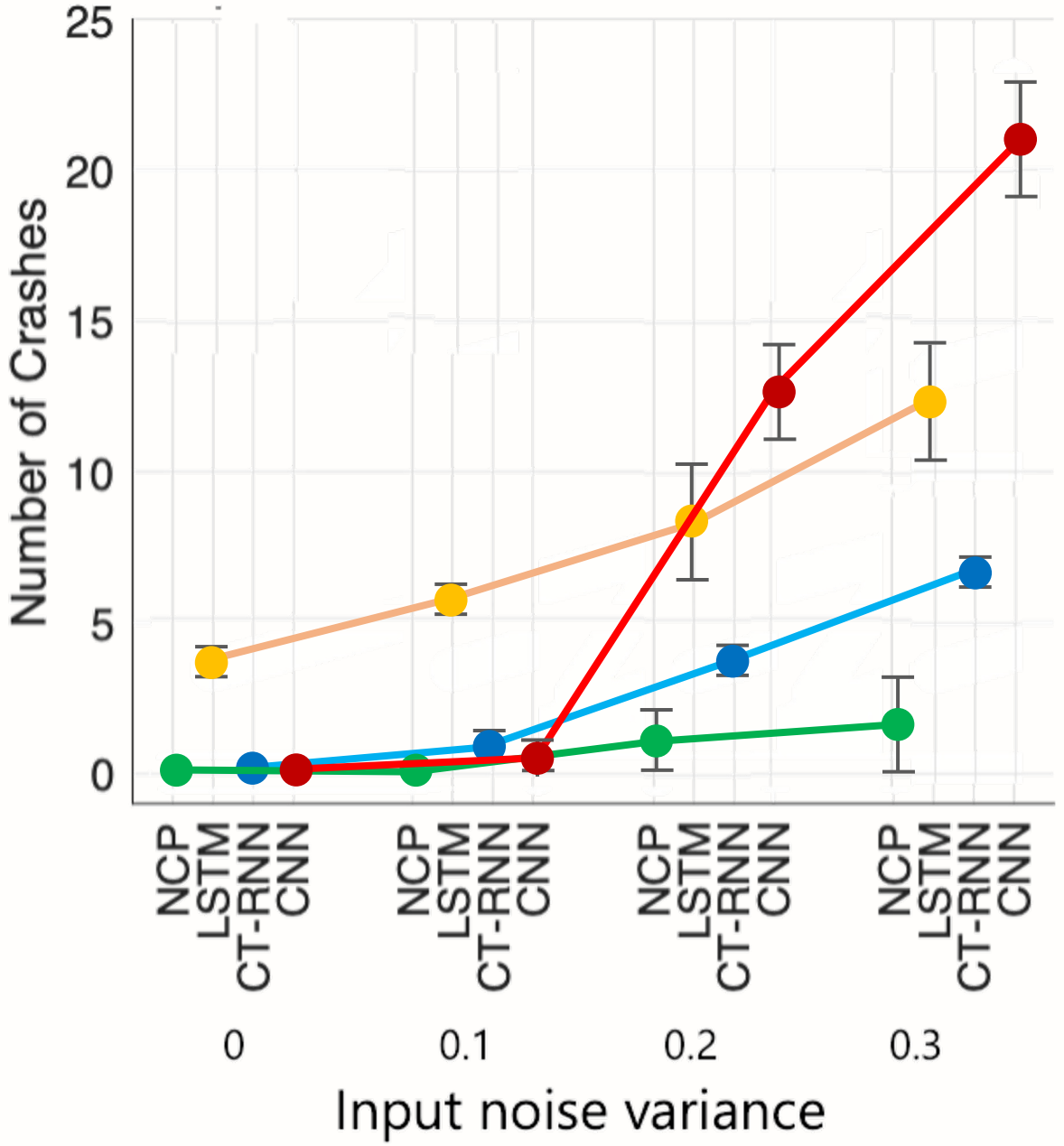

NCP driving performance
under σ^2=0.1 pertubation
Camera input stream
Conv #1
Conv #2
Conv #3
Attention map
Sensory neurons
Conv #5
Inter neurons
Command neurons
Motor neurons
Normalized neural state
+1
0
-1
Map
Mode: Manual

LTCs: Performance

Robustness to perturbations

CNN
LSTM
CT-RNN
LTC (ours)

Liquid time-constant neuron resilience to sensory noise

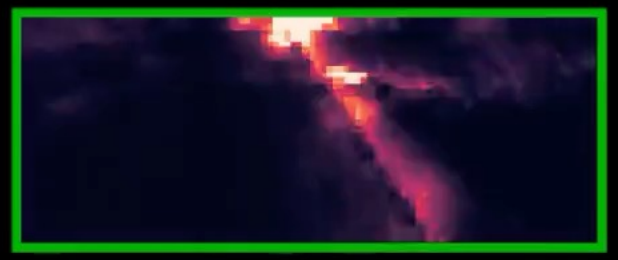


Lechner*, Hasani* et al. Nature MI, 2020

Camera input stream

Map

CNN



Mode: Autonomous

CT-RNN

Mode: Autonomous

LSTM

Mode: Autonomous

Our solution

Mode: Autonomous

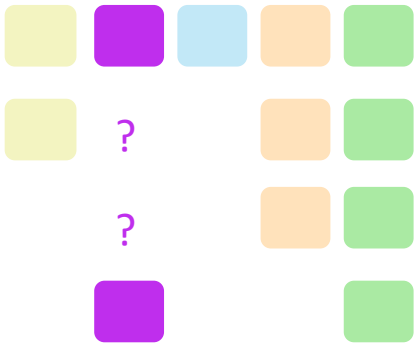
Outline

- ✓ Liquid time-constant networks
- ✓ Autonomy with liquid networks
- ✓ **Causality**
- ✓ Closed-form Continuous-time decision-making
- ✓ Task understanding and out of distribution generalization

Why can LTCs learn better causal relationships?

Taxonomy of Models

Adapted from: Peters, Janzing, Schölkopf, MIT Press, 2017



Physical

Structural causal

Causal graphs

Statistical models

Insights about system

Learn from data

Answer counter factual

Predict in IID

Interventions

Dynamic Causal Models

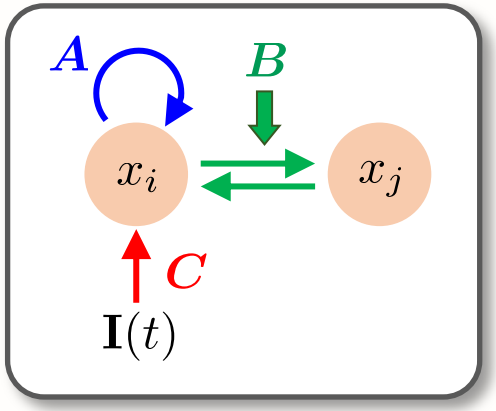
$$\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t), \mathbf{I}(t); \theta)$$

$$(A + \mathbf{I}(t)B)\,\mathbf{x}(t) + C(\mathbf{I}(t))$$

Internal coupling

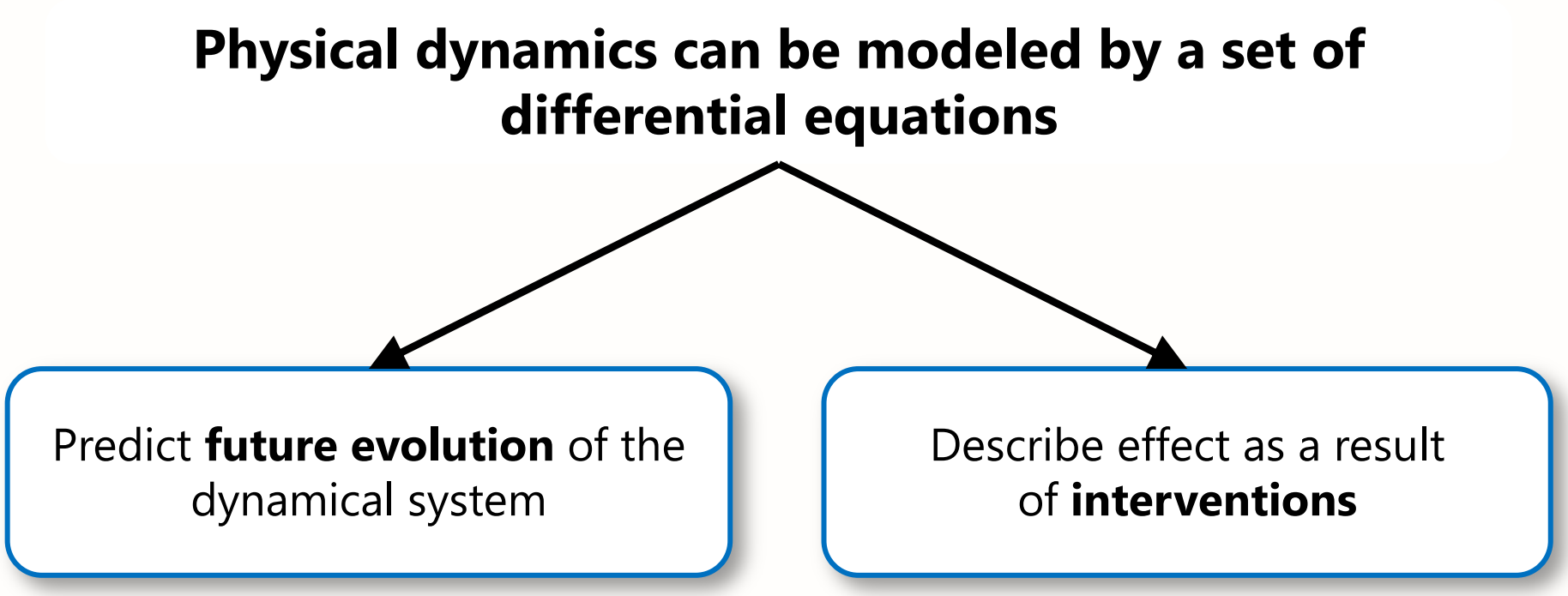
Internal Intervention

External Intervention



The **LTC** model reduces mathematically to a **Dynamic Causal Model**

Differential equations can form causal structures



(Friston et al., 2003)

Differential equations can form causal structures

Given the following system of differential equations:

$$\frac{d\mathbf{x}}{dt} = g(\mathbf{x}),$$

where $\mathbf{x} \in \mathbb{R}^d, \quad \mathbf{x}(0) = x_0, \quad g(\mathbf{x}) = \text{nonlinearity}$

Picard-Lindelöf theorem

(Nevanlinna, 1989)

states that above DE has a unique solution as long as g is Lipschitz

Euler solution

The Euler method unrolling:

$$\mathbf{x}(t + \delta t) = \mathbf{x}(t) + dt\, g(\mathbf{x})$$

Causal structure

(Schölkopf, 2019)

Representation under uniqueness conditions forms a temporally **causal structure**

Dynamic Causal Models (DCMs)

$$\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t), \mathbf{I}(t); \theta)$$

Bilinear approximation

(Friston et al., 2003)

$$\frac{d\mathbf{x}}{dt} = (A + \mathbf{I}(t)B)\,\mathbf{x}(t) + C(\mathbf{I}(t))$$

Internal coupling

$$A = \left.\frac{\partial g}{\partial \mathbf{x}}\right|_{\mathbf{I}=0}$$

Regulates hidden state

Internal Intervention

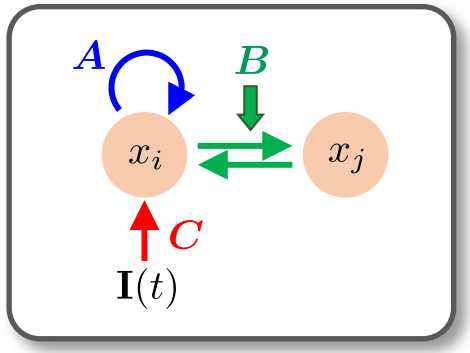
$$B = \frac{\partial^2 g}{\partial \mathbf{x}\,\partial \mathbf{I}}$$

Controls coupling sensitivity among network's nodes

External Intervention

$$C = \left.\frac{\partial g}{\partial \mathbf{I}}\right|_{\mathbf{x}=0}$$

Regulates external input



The **Liquid Time-constant (LTC)** model reduces to a **Dynamic Causal Model** of this form if

1. $g(\cdot)$ **is continuous and bounded**

$e.g.,\quad \tanh(W_r\mathbf{x}(t) + W\mathbf{I}(t) + b)$

(Hirsch and Smale, 1973, Hasani, et al. 2021)

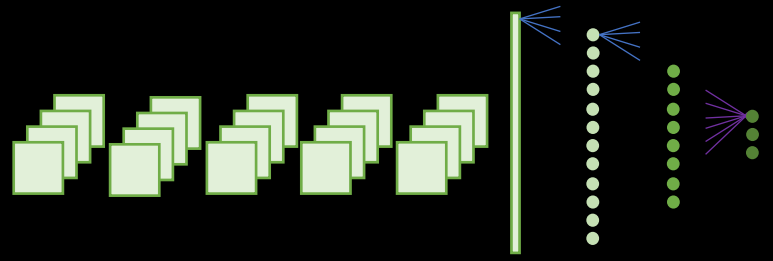
2. τ **is positive**

Enforced by activation constraint

(Funahashi and Nakamura, 1993)

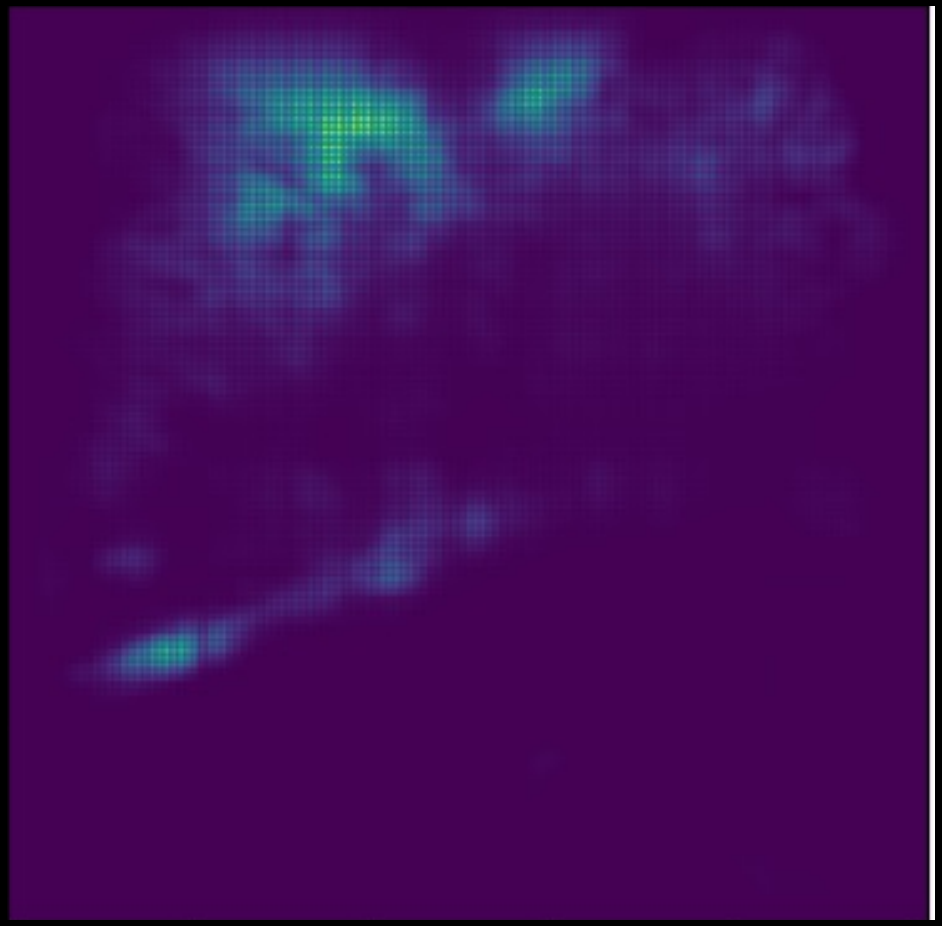
LTC-based: Neural Circuit Policies

Performance – Attention



Flying Performance

Visual Backprop Attention Map

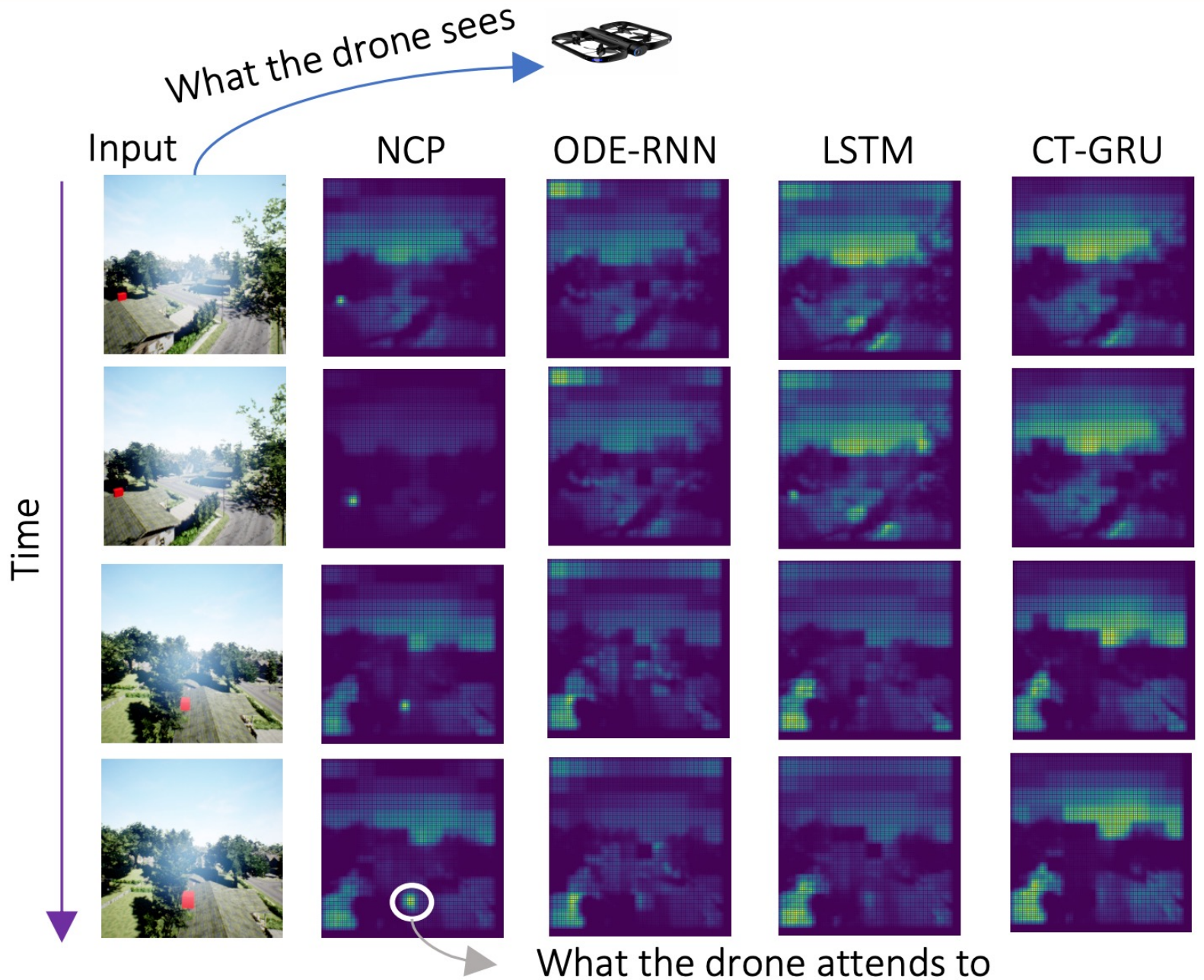


Vorbach*, Hasani*, et al. NeurIPS 2021

Neural Circuit Policies

Performance – Attention

- Red cubic target is fixed.
- Drone learns to navigate to target by visual inputs only



Liquid Network's Limitations

- The complexity of LTC-based networks depends on the complexity of their ODE-solvers
  → long training and test time for larger networks
    solution: 1) use stable fixed-step ODE-solvers [Hasani et al, AAAI 2021]
    2) use Sparse Flows [Liebenwein* & Hasani * et al, NeurIPS 2021]
    3) use Hypersolvers [Poli et al, NeurIPS 2020]
    4) **use Closed-form variants** [Hasani et al, Nature MI, 2022]

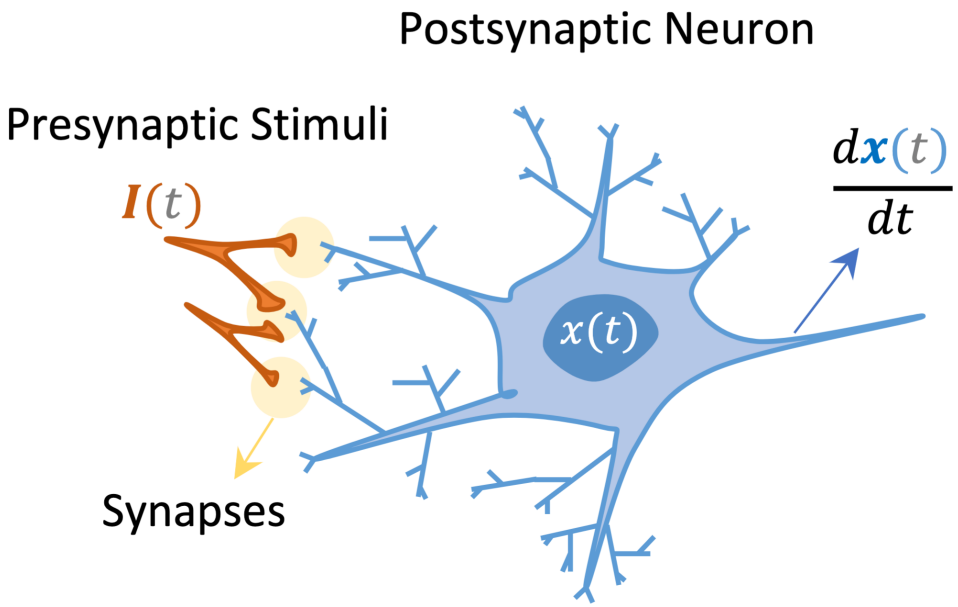
- Similar to all ODE-based networks they might express vanishing/exploding gradients issues
  → Face difficulties in learning very long sequences
    solution: 1) use mixed memory wrappers with them: [Lechner and Hasani, MemARI NeurIPS 2022]
    2) Liquid Structural State-Space Models [Hasani et al. ICLR 2023]

Outline

- ✓ Liquid time-constant networks
- ✓ Autonomy with liquid networks
- ✓ Causality
- ✓ **Closed-form Continuous-time decision-making**
- ✓ Task understanding and out of distribution generalization

Closed-form Solution of Liquid Networks

Closed-form Continuous-time Neural Networks



$$S(t) = f\big(\boldsymbol{I}(t)\big)\,(A - \boldsymbol{x}(t))$$

$$\frac{d\boldsymbol{x}(t)}{dt} = -\frac{\boldsymbol{x}(t)}{\tau} + S(t)$$

this is a liquid time-constant differential equation instance

we solve this in closed-form

$$\boldsymbol{x}(t) = (\boldsymbol{x}(0) - A)\, e^{-\left[\frac{1}{\tau} + f(I(t))\right]t}\, f(-I(t)) + A$$

$\boldsymbol{x}(t)$	Postsynaptic neuron's potential
A	Synaptic reversal potential
$f(.)$	Synaptic release nonlinearity
τ	Postsynaptic neuron's time-constant

Closed-form Solution of Liquid Networks

Closed-form Continuous-time Neural Networks (CfCs)

Time Complexity of the process to compute K solver's steps

Method	Complexity	Local Error
p-th order solver	$\mathcal{O}(K \cdot p)$	$\mathcal{O}(\epsilon^{p+1})$
adaptive–step solver	–	$\mathcal{O}(\tilde{\epsilon}^{\,p+1})$
Euler `hypersolver`	$\mathcal{O}(K)$	$\mathcal{O}(\delta\epsilon^2)$
p-th order `hypersolver`	$\mathcal{O}(K \cdot p)$	$\mathcal{O}(\delta\epsilon^{p+1})$
CfC (Ours)	$\mathcal{O}(\tilde{K})$	**not relevant**

ϵ is step size

$\tilde{\epsilon}$ is the maximum step size

$\delta \ll 0$

$\tilde{K}$ is the input time steps

Hasani et al. Nature MI 2022

Closed-form Solution of Liquid Networks

Closed-form Continuous-time Neural Networks (CfCs)

Sequence and time-step prediction complexity

Model	Sequence prediction	Time-step prediction
RNN	$\mathcal{O}(nk)$	$\mathcal{O}(k)$
ODE-RNN	$\mathcal{O}(nkp)$	$\mathcal{O}(kp)$
Transformer	$\mathcal{O}(n^2k)$	$\mathcal{O}(nk)$
CfC	$\mathcal{O}(nk)$	$\mathcal{O}(k)$

n is sequence length
k is the number of hidden units
p is the order of the ODE-solver

Hasani et al. Nature MI 2022

Deriving the Closed-form Solution of LTCs

Theorem

For a given LTC initial-value problem determined by $\frac{d\mathbf{x}}{dt} = -(w_\tau + f(\mathbf{x}, \mathbf{I}, \theta))\mathbf{x}(t) + Af(\mathbf{x}, \mathbf{I}, \theta)$,
Constructed by one cell,
receiving a single dimensional time-series input $I(t)$
with no self-connections,
the following expression $\tilde{x}(t) = (x(0) - A)e^{-\left[w_\tau t + f(I(t))t\right]} f(-I(t)) + A$
is an approximation of its closed-form solution by the following sharpness results:

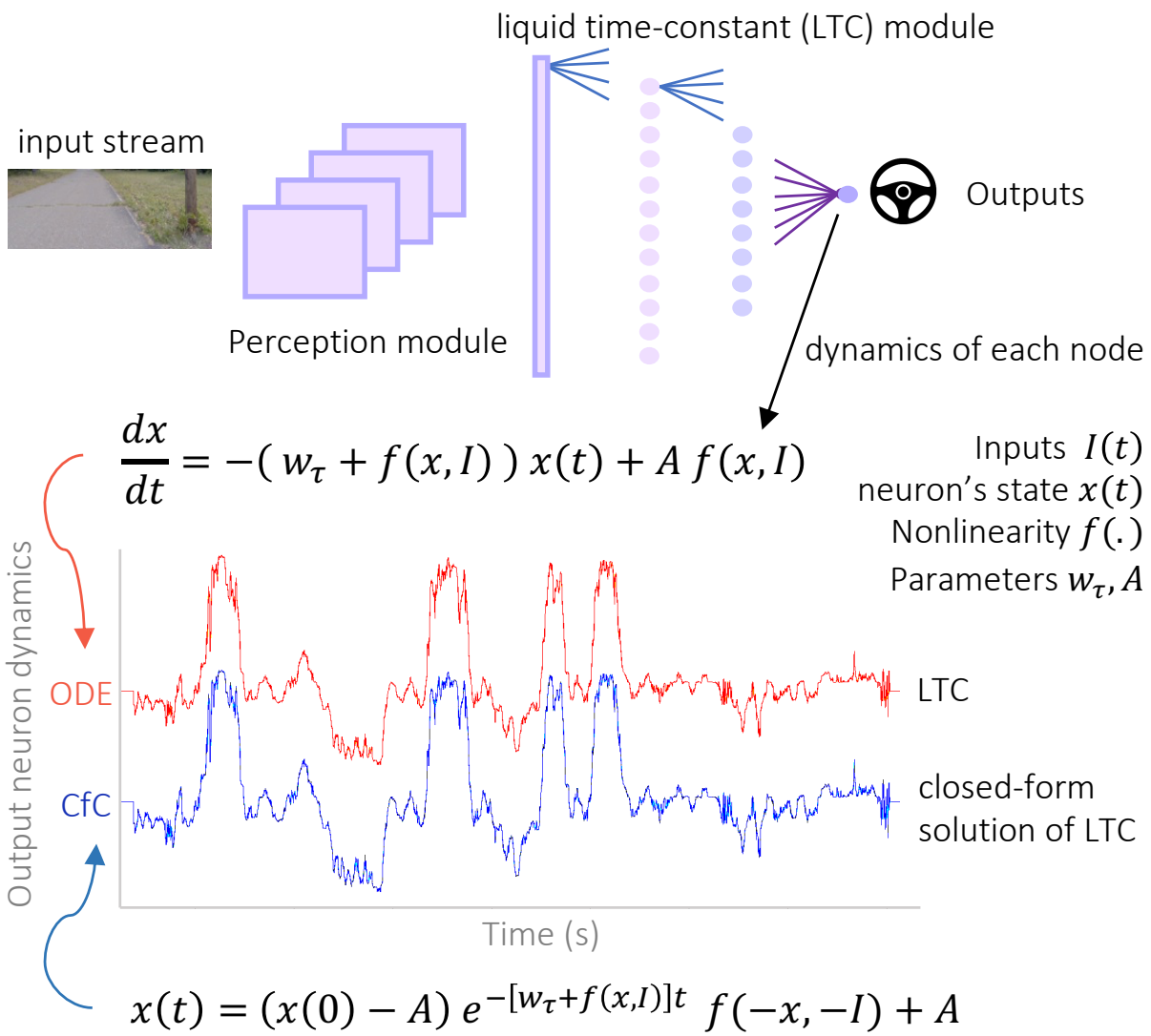
$$|x(t) - \tilde{x}(t)| \leq |x(0) - A|e^{-w_\tau t} \text{ for all } t \geq 0$$

1. For any $t \geq 0$, we have $\sup\{\frac{1}{c}(x(t) - \tilde{x}(t)) \mid I : [0; t] \to \mathbb{R}\} = e^{-w_\tau t}$.

2. For any $t \geq 0$, we have $\inf\{\frac{1}{c}(x(t) - \tilde{x}(t)) \mid I : [0; t] \to \mathbb{R}\} = e^{-w_\tau t}(e^{-t} - 1)$.

$c = x(0) - A$

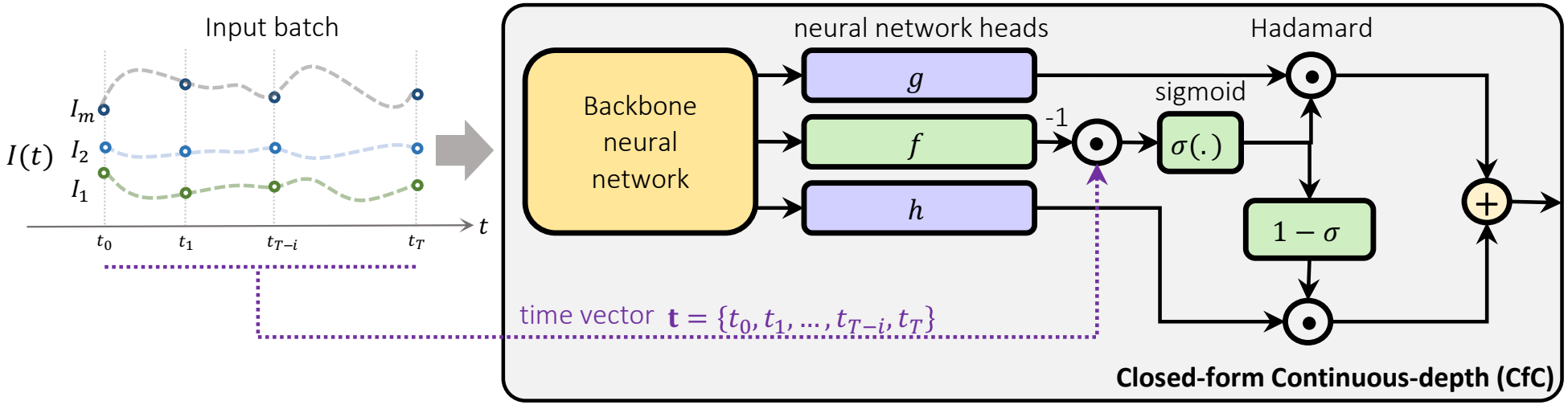
Tightness of the Closed-form Solution in Practice



Hasani et al. Nature MI 2022

Closed-form Continuous-time Neural Networks

Liquid CfCs



$$\mathbf{x}(t) = \underbrace{\sigma(-f(\mathbf{x}, \mathbf{I}; \theta_f)\ \mathbf{t})}_{\text{time-continuous gating}} \odot g(\mathbf{x}, \mathbf{I}; \theta_g) + \underbrace{\left[1 - \sigma(-[f(\mathbf{x}, \mathbf{I}; \theta_f)]\ \mathbf{t})\right]}_{\text{time-continuous gating}} \odot h(\mathbf{x}, \mathbf{I}; \theta_h)$$

LTC's Closed-form solution

$$\tilde{x}(t) = (x(0) - A)e^{-\left[w_\tau t + f(I(t))t\right]} f(-I(t)) + A$$

Liquid ODE

$$\frac{d\mathbf{x}}{dt} = -(w_\tau + f(\mathbf{x}, \mathbf{I}, \theta))\mathbf{x}(t) + Af(\mathbf{x}, \mathbf{I}, \theta)$$

How Well liquid CfCs perform in Time-series modeling?

Physical Dynamics Modeling

Table 6: **Per time-step regression**. Walker2d kinematic dataset. (mean $\pm$ std, $N = 5$)

Model	Square-error
†ODE-RNN (Rubanova et al., 2019)	1.904 $\pm$ 0.061
†CT-RNN (Funahashi and Nakamura, 1993)	1.198 $\pm$ 0.004
†Augmented LSTM (Hochreiter and Schmidhuber, 1997)	1.065 $\pm$ 0.006
†CT-GRU (Mozer et al., 2017)	1.172 $\pm$ 0.011
†RNN-Decay (Rubanova et al., 2019)	1.406 $\pm$ 0.005
†Bi-directional RNN (Schuster and Paliwal, 1997)	1.071 $\pm$ 0.009
†GRU-D (Che et al., 2018)	1.090 $\pm$ 0.034
†PhasedLSTM (Neil et al., 2016)	1.063 $\pm$ 0.010
†GRU-ODE (Rubanova et al., 2019)	1.051 $\pm$ 0.018
†CT-LSTM (Mei and Eisner, 2017)	1.014 $\pm$ 0.014
†ODE-LSTM (Lechner and Hasani, 2020)	0.883 $\pm$ 0.014
coRNN (Rusch and Mishra, 2021)	3.241 $\pm$ 0.215
Lipschitz RNN (Erichson et al., 2021)	1.781 $\pm$ 0.013
LTC (Hasani et al., 2021)	**0.662** $\pm$ 0.013
Transformer (Vaswani et al., 2017)	0.761 $\pm$ 0.032
Cf-S (ours)	0.948 $\pm$ 0.009
CfC-noGate (ours)	**0.650** $\pm$ 0.008
CfC (ours)	**0.643** $\pm$ 0.006
CfC-mmRNN (ours)	**0.617** $\pm$ 0.006

Outline

- ✓ Liquid time-constant networks
- ✓ Autonomy with liquid networks
- ✓ Causality
- ✓ Closed-form Continuous-time decision-making
- ✓ **Task understanding and out of distribution generalization**

Can AI Systems Understand the Task They Are Given?

Fly-to-target tasks

Chahine* & Hasani* et al. Science Robotics 2023

Liquid Networks in closed-loop Aerial Vehicle Control

Sample Training Data

Chahine* & Hasani* et al. Science Robotics 2023

Standard Deep Neural Network

Task: Identify and navigate to target
Deploy: Closed-loop testing
Result: Spurious attention → Fail

Agent

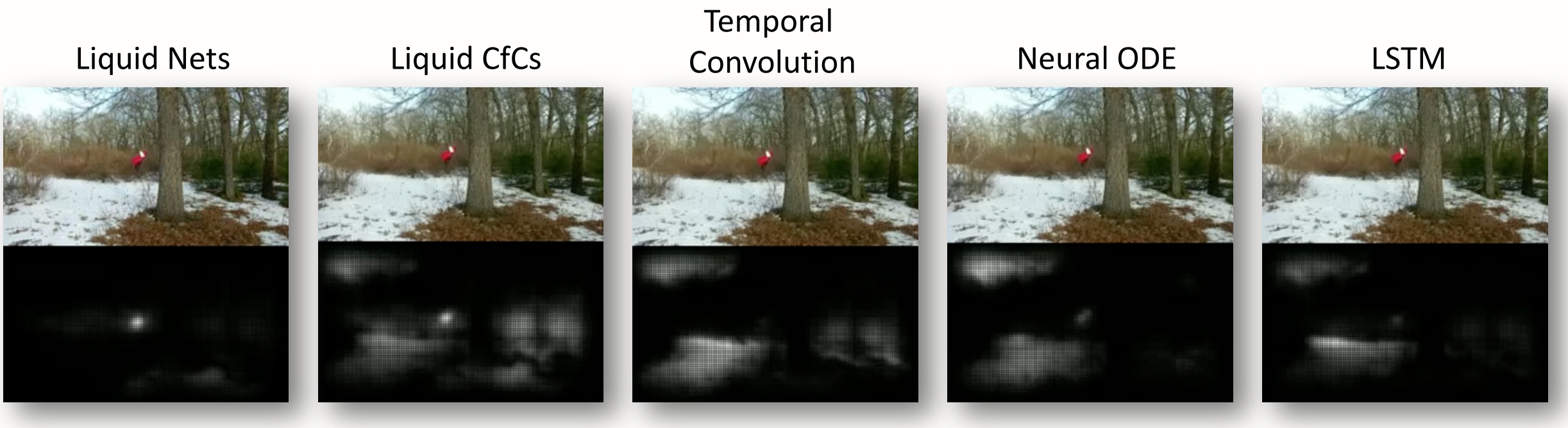
Target

LTC-based Network

Task: Identify and navigate to target
Deploy: Closed-loop testing
Result: Causal attention → Success

Generalization and Task understanding with Liquid Networks

Testing attention profiles of different networks offline

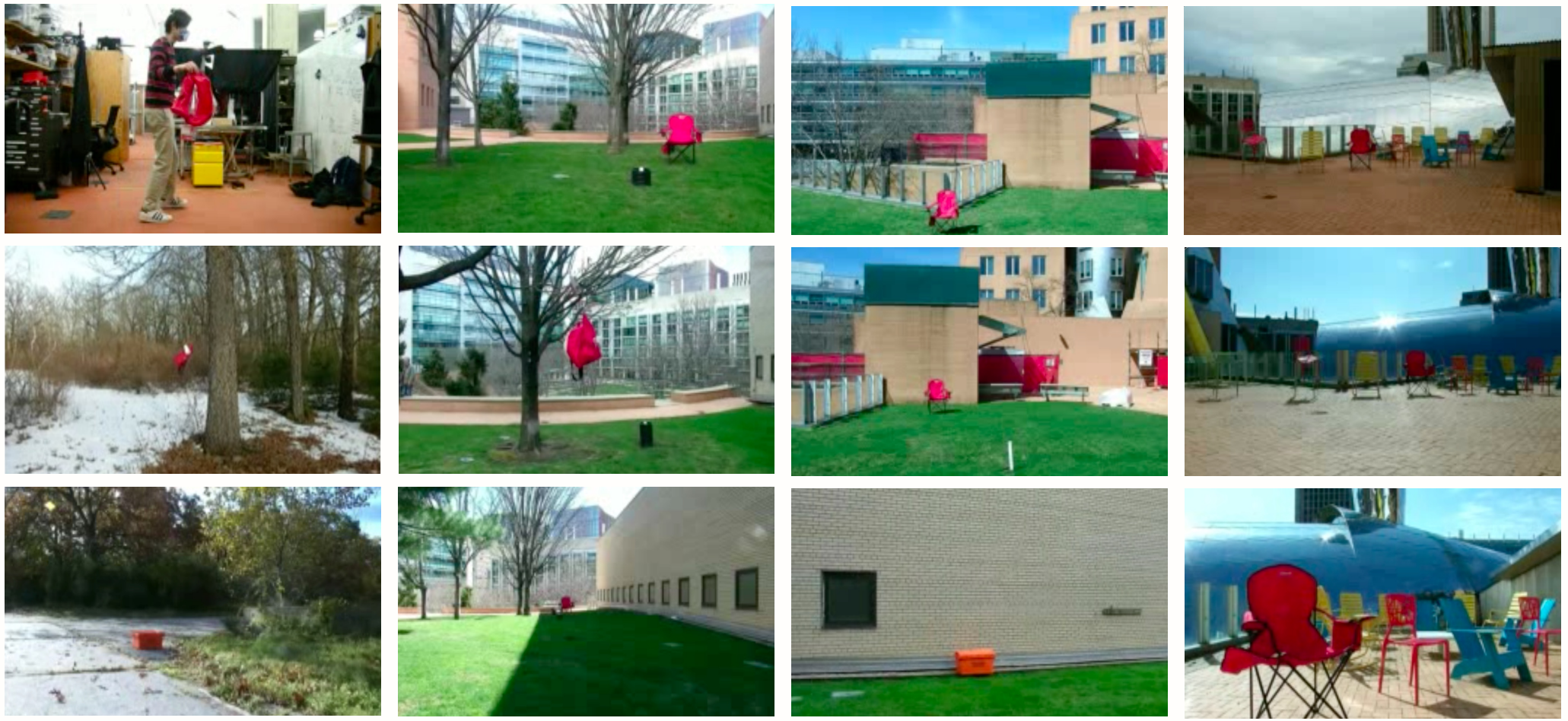


Liquid CfC = liquid closed-form continuous-time
Neural ODE = neural ordinary differential equations
LSTM = long short-term memory

Chahine* & Hasani* et al. Science Robotics 2023

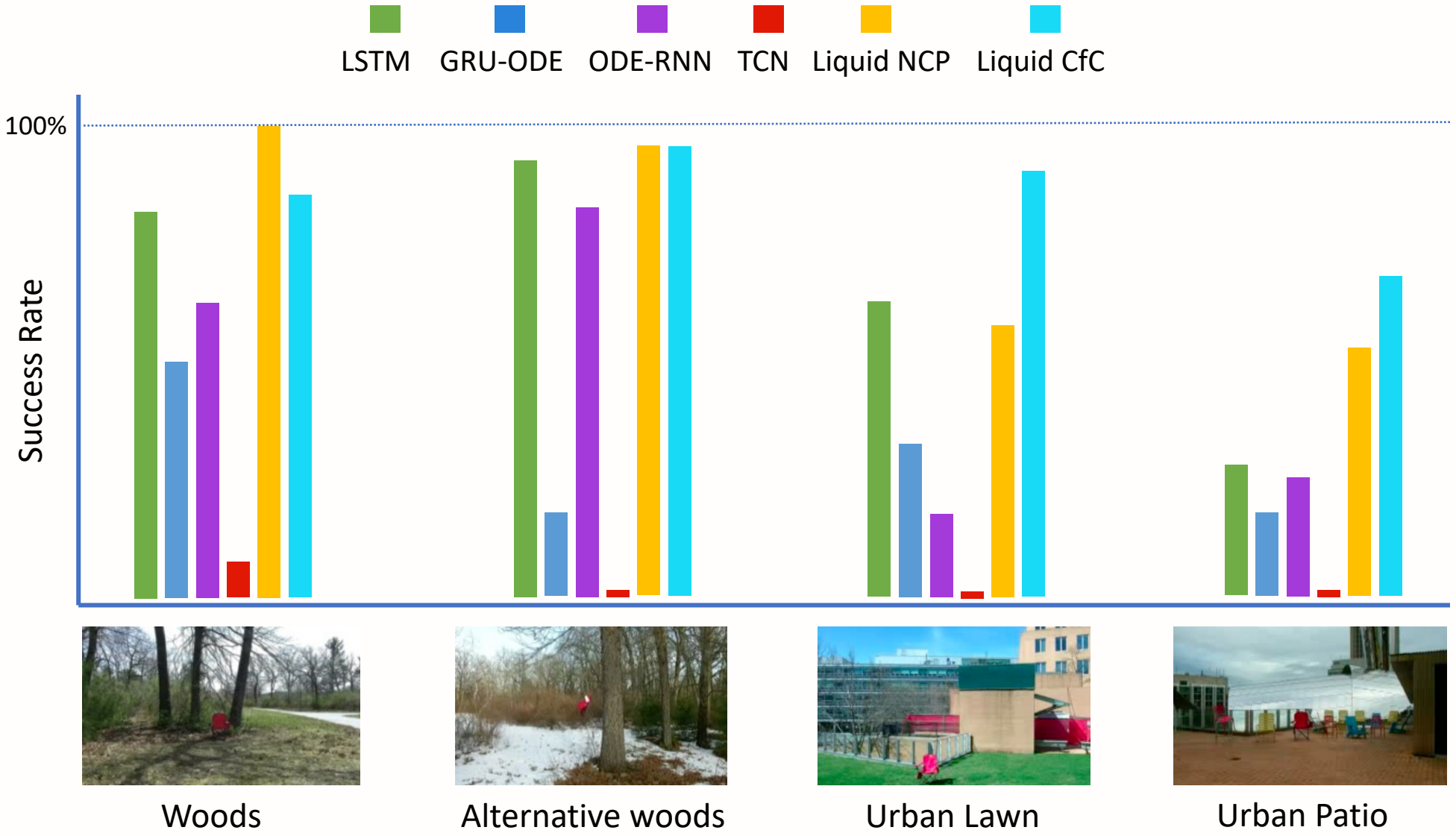
Liquid networks zero-shot transfer performance.

Generalization out-of-distribution



Chahine* & Hasani* et al. Science Robotics 2023

In and Out of Distribution Performance



Chahine* & Hasani* et al. Science Robotics 2023

Science Robotics
AAAS

Science Robotics
AAAS

Liquid Structural State-Space Models

Liquid S4

$$\dot{x}(t) = \left[\mathbf{A} + \mathbf{B}\, u(t)\right] x(t) + \mathbf{B}\, u(t)$$
$$y(t) = \mathbf{C}\, x(t)$$

$x(t)$ is an N-dimensional latent state, receiving a 1-dimensional input signal $u(t)$, and computing a 1-dimensional output signal $y(t)$. $\mathbf{A}^{(N\times N)}$, $\mathbf{B}^{(N\times 1)}$, and $\mathbf{C}^{(1\times N)}$. Note that $\mathbf{D}$ is set to zero for simplicity.

Hasani et al. ICLR 2023

Liquid vs Transformers

Performance on sequence modeling tasks:

- *Sequential image processing*

  46% accuracy improvement over Transformers

- *Text sentiment analysis*

  10% accuracy improvement over Transformers

- *Recovering long info paths from 2D trajectories*

  16% accuracy improvement (context length 1k)
  96% improvement (context length 16k)

- *Log list operations*

  30% accuracy improvement over Transformers

- *Information retrieval*

  11% accuracy improvement

Model (input length)	ListOps 2048	IMDB 2048	AAN 4000	CIFAR 1024	Pathfinder 1024	Path-X 16384	Avg.
Random*	10.00	50.00	50.00	10.00	50.00	50.00	36.67
Transformer* (Vaswani et al., 2017)	36.37	64.27	57.46	42.44	71.40	x	54.39
Local Att.* (Tay et al., 2020b)	15.82	52.98	53.39	41.46	66.63	x	46.06
Sparse Transformer* (Child et al., 2019)	17.07	63.58	59.59	44.24	71.71	x	51.24
Longformer* (Beltagy et al., 2020)	35.63	62.85	56.89	42.22	69.71	x	53.46
Linformer* (Wang et al., 2020)	16.13	65.90	53.09	42.34	75.30	x	50.55
Reformer* (Kitaev et al., 2019)	37.27	56.10	53.40	38.07	68.50	x	50.56
Sinkhorn Trans.* (Tay et al., 2020a)	33.67	61.20	53.83	41.23	67.45	x	51.23
BigBird* (Zaheer et al., 2020)	36.05	64.02	59.29	40.83	74.87	x	55.01
Linear Trans.* (Katharopoulos et al., 2020)	16.13	65.90	53.09	42.34	75.30	x	50.46
Performer* (Choromanski et al., 2020)	18.01	65.40	53.82	42.77	77.05	x	51.18
FNet** (Lee-Thorp et al., 2021)	35.33	65.11	59.61	38.67	77.80	x	54.42
Nyströmformer** (Xiong et al., 2021)	37.15	65.52	79.56	41.58	70.94	x	57.46
Luna-256** Ma et al. (2021)	37.25	64.57	79.29	47.38	77.72	x	59.37
H-Transformer-1D** (Zhu and Soricut, 2021)	49.53	78.69	63.99	46.05	68.78	x	61.41
CDIL (Cheng et al., 2022)	44.05	86.78	85.36	66.91	91.70	x	74.96
DSS (Gupta, 2022)	57.6	76.6	87.6	85.8	84.1	85.0	79.45
S4 (original)** (Gu et al., 2022a)	58.35	76.02	87.09	87.26	86.05	88.10	80.48
S4-LegS (Gu et al., 2022b)	59.60 (0.07)	86.82 (0.13)	90.90 (0.15)	88.65 (0.23)	94.20 (0.25)	96.35	86.09
S4-FouT (Gu et al., 2022b)	57.88 (1.90)	86.34 (0.31)	89.66 (0.88)	89.07 (0.19)	94.46 (0.26)	x	77.90
S4-LegS/FouT (Gu et al., 2022c)	60.45 (0.75)	86.78 (0.26)	90.30 (0.28)	89.00 (0.26)	94.44 (0.08)	x	78.50
S4D-LegS (Gu et al., 2022b)	60.47 (0.34)	86.18 (0.43)	89.46 (0.14)	88.19 (0.26)	93.06 (1.24)	91.95	84.89
S4D-Inv (Gu et al., 2022b)	60.18 (0.35)	87.34 (0.20)	91.09 (0.01)	87.83 (0.37)	93.78 (0.25)	92.80	85.50
S4D-Lin (Gu et al., 2022b)	60.52 (0.51)	86.97 (0.23)	90.96 (0.09)	87.93 (0.34)	93.96 (0.60)	x	78.39
S5 (Smith et al., 2022)	61.00	86.51	88.26	86.14	87.57	85.25	82.46
Liquid-S4 (ours)	**62.75** (0.2) p = 5	**89.02** (0.04) p=6	**91.20** (0.01) p=2	**89.50** (0.4) p=3	**94.8** (0.2) p=2	**96.66**(0.001) p=2	**87.32**

Hasani et al. ICLR 2023

Liquid vs Transformers

Performance on sequential long-term signal classification:

30%

Performance improvement over Transformers

Model	Accuracy
Transformer (Trinh et al., 2018)	62.2
FlexConv (Romero et al., 2021a)	80.82
TrellisNet (Bai et al., 2018)	73.42
LSTM	63.01
r-LSTM (Trinh et al., 2018)	72.2
UR-GRU (Gu et al., 2020b)	74.4
HiPPO-RNN (Gu et al., 2020a)	61.1
LipschitzRNN (Erichson et al., 2021)	64.2
Liquid-S4-PB (ours)	**92.02** (0.14) p=3

Hasani et al. ICLR 2023

Use Case: Order Book Price Forecasting – Overview

Typical LOBSTER Data Sample

$$\underbrace{[p_1^a, v_1^a, p_1^b, v_1^b, \ldots, p_d^a, v_d^a, p_d^b, v_d^b]}_{\text{Length } 4d \ (d=\text{depth})}$$

$p_d^a,\ p_d^b$ - Ask/Bid prices at depth d

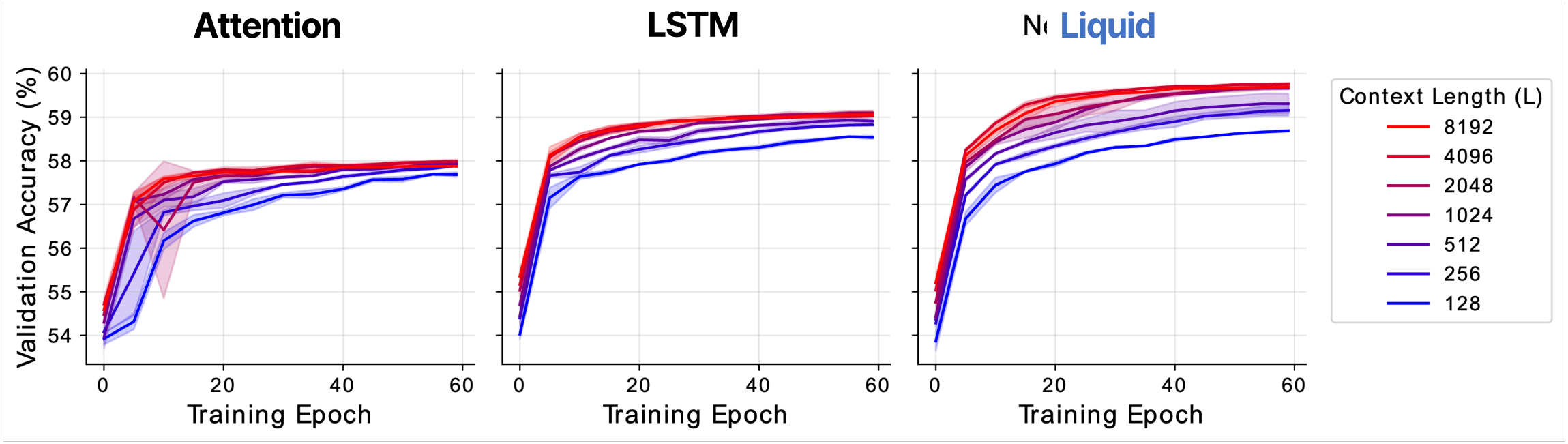
$v_d^a,\ v_d^b$ - Ask/Bid volumes at depth d

LOBSTER Dataset

- ✓ High frequency order book data at irregularly frequencies
- ✓ Non-stationary with low signal-to-noise ratio
- ✓ Signals at multiple timescales, with long-range interactions
- ✓ **Goal: Predict whether mid price will move up/down in the next t_f seconds**

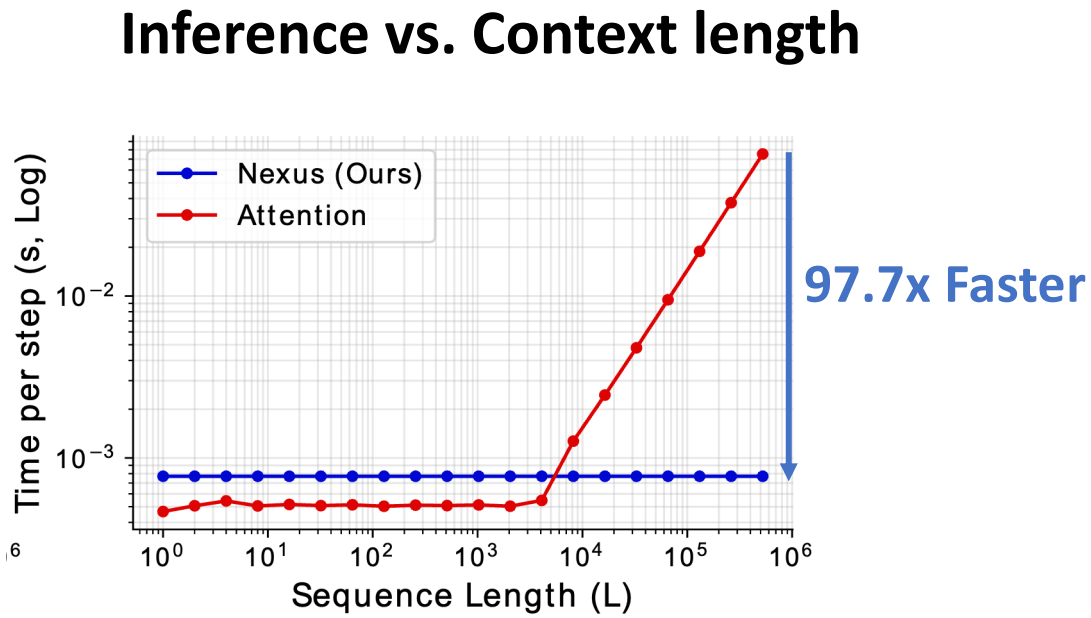
Use Case: Order Book Price Forecasting – Results

Context Length vs. Accuracy



Liquid AI's models can effectively leverage long contexts lengths

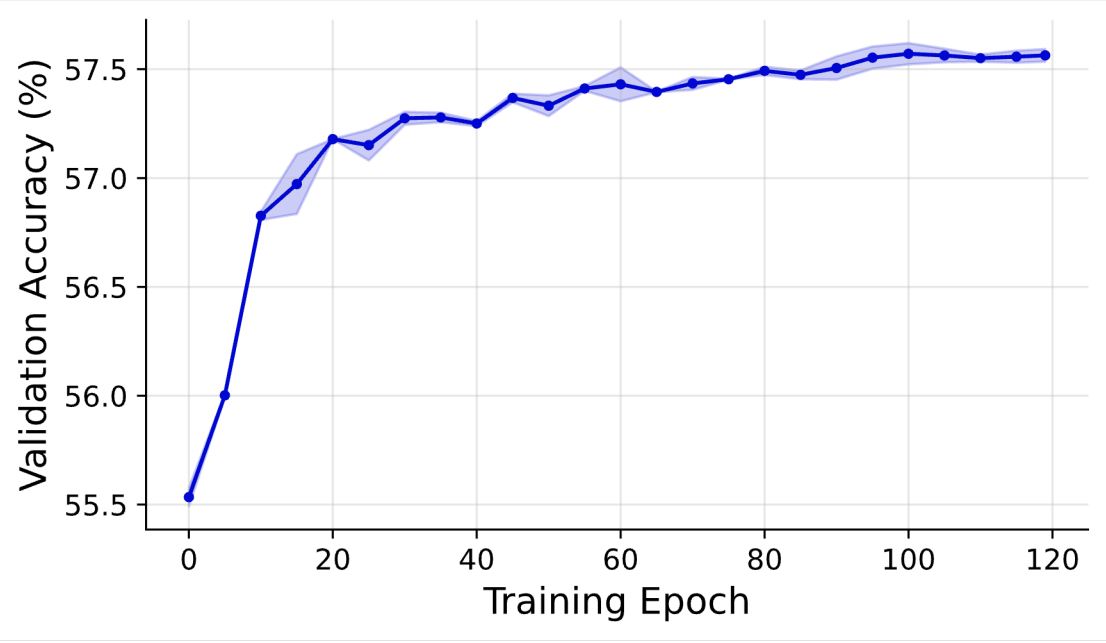
Use Case: Order Book Price Forecasting – Results



At inference time, Liquid AI models' inference speeds are Constant in sequence length, while attention gets slower with longer sequences

Use Case: Order Book Price Forecasting – Results

GPSM Nexus model trained on 130k context length with $\Delta t = 0.05s$



Liquid AI models scale to ultra long (>130k) context lengths *in training*

Use Case: Credit Card Fraud Detection

Liquid resulted in **5% recall accuracy improvement** and **82% reduction in false positive rate**

	LSTM-Attention	Liquid GPSM	
Parameters	32k	7k	34k
Recall (%)	94.9%	99.4%	**99.9%**
False Positive	0.027	0.01	**0.002**
Training Step (ms)	4.52	4.36	8.09
Inference (ms)	1.89	**1.35**	2.48

Liquid vs Transformers

Performance on medical time-series (Human vital signals - BIDMC)

33x

Accuracy improvement over Transformers.

	BIDMC		
Model	HR	RR	SPO2
UnICORNN (Rusch & Mishra, 2021b)	1.39	1.06	0.869
coRNN (Rusch & Mishra, 2021a)	1.81	1.45	-
CKConv*	2.05	1.214	1.051
NRDE (Morrill et al., 2021)	2.97	1.49	1.29
LSTM (Rusch & Mishra, 2021b)	10.7	2.28	-
Transformer*	12.2	2.61	3.02
XGBoost (Tan et al., 2021)	4.72	1.67	1.52
Random Forest (Tan et al., 2021)	5.69	1.85	1.74
Ridge Regress. (Tan et al., 2021)	17.3	3.86	4.16
Liquid-S4-PB (ours)	**0.303** (0.002) p=3	**0.158** (0.001) p=2	**0.066** (0.002) p=4

Hasani et al. ICLR 2023

Use Case: Speech Recognition on the Edge

	Attention	Liquid Nexus Model
Hidden Size	128	128
Test Accuracy (5 epochs)	0.86	**0.943**
Time per Batch (s)	158.71	**0.4**

- Controlling devices with voice commands is challenging:
  - Requires an efficient & accurate model that recognizes words amidst noisy / low-quality audio

- **Liquid Nexus Model efficiently and accurately classifies high sample rate audio data in Speech Commands.**

- **Over 300X Time per Batch improvement** while maintaining efficiency

USE CASE

Liquid Networks to fly US Air Force jet aircraft X-62 VISTA (derived from the F-16D Fighting Falcon)

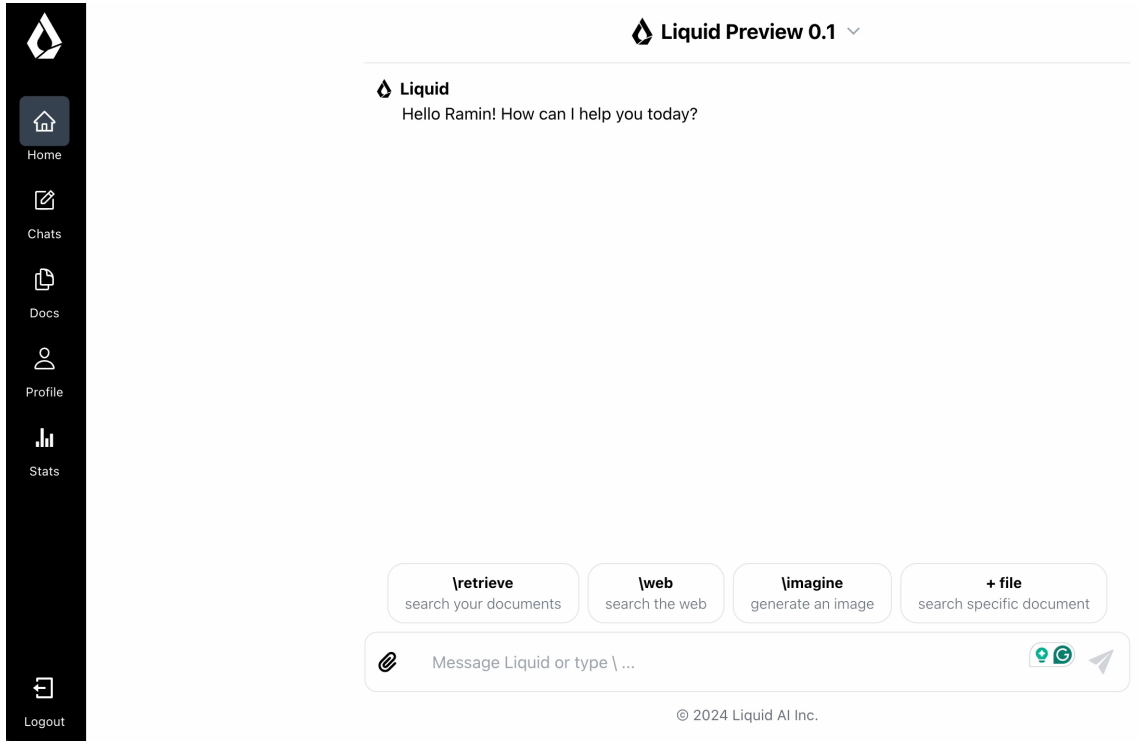
©Massachusetts Institute of Technology MIT Lincoln Laboratory, Defense Advanced Research Projects Agency (DARPA) EpiSci AFWERX and the Test Pilot School 412th Test Wing, Edwards Air Force Base at Edwards Air Force Base.

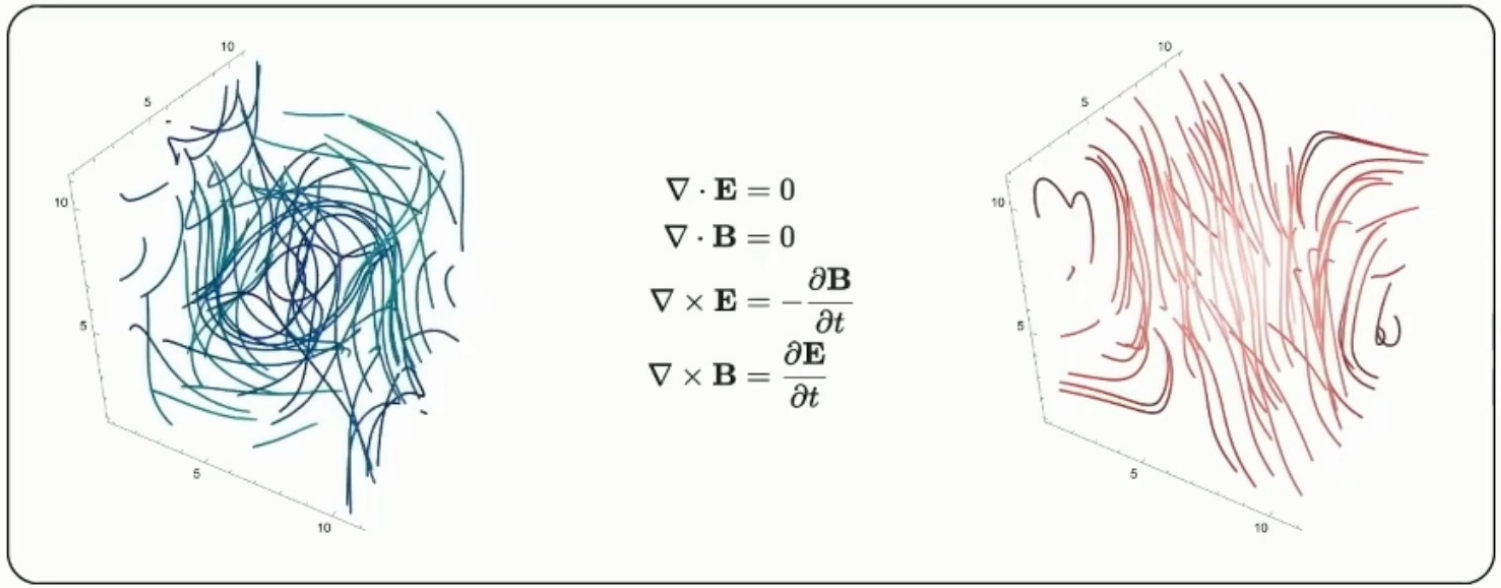
Creator: Hand-out | Photo Credit: Lockheed Martin Aeronautics

Liquid Foundation Models (LFMs)

Model Sizes	Context Length
1-100B	32k-20M

Scaled Liquid models from 80k-parameter network that drive cars to:

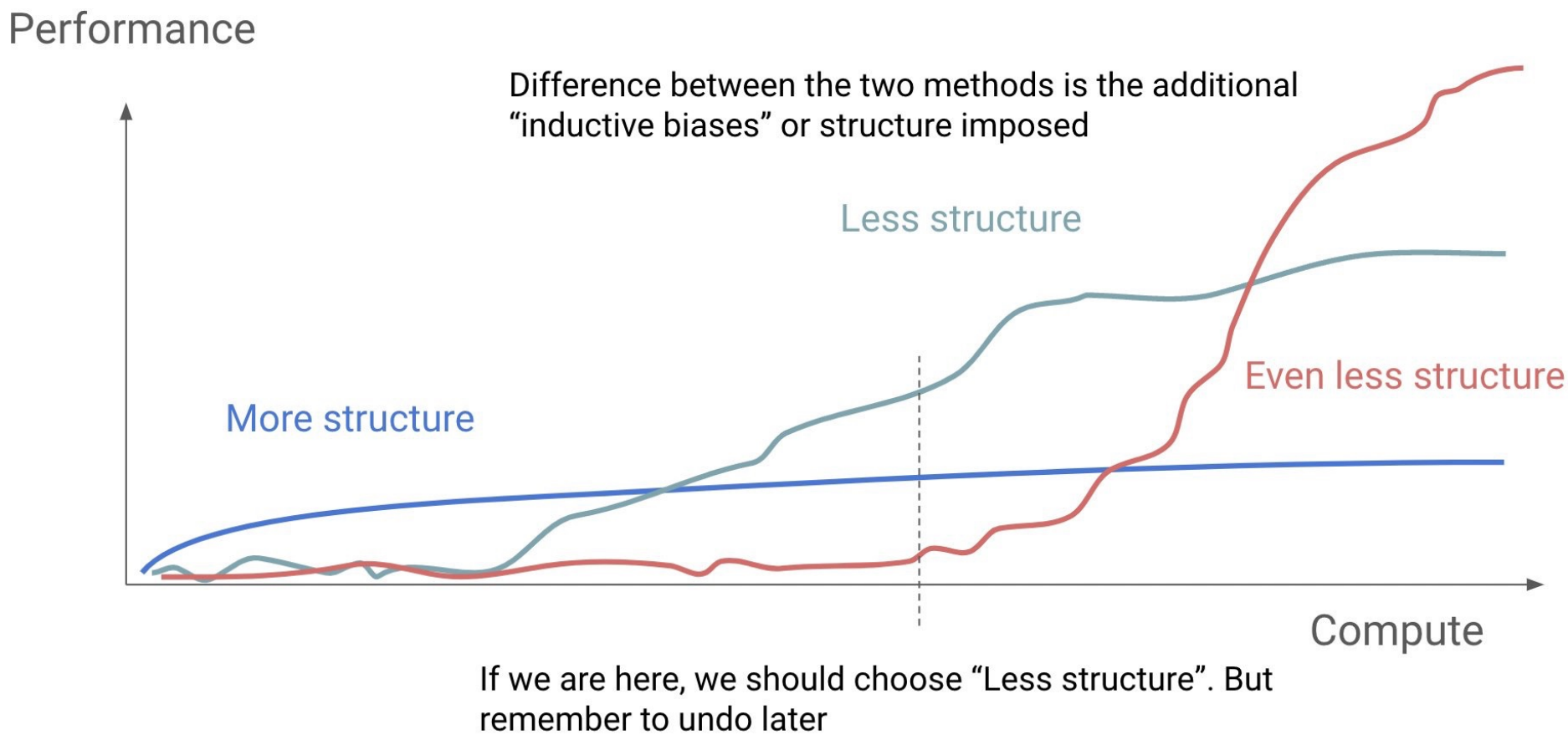




GP PRIORS FOR SYSTEMS OF LINEAR PDE WITH CONSTANT COEFFICIENTS

Marc Härkönen[1,2], Markus Lange-Hegermann[3], Bogdan Raiță[4]

[1]Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
[2]Fano Labs, Hong Kong SAR, China
[3]Institute Industrial IT, TH-OWL, Lemgo, Germany
[4]Scuola Normale Superiore di Pisa, Pisa, Italy



Credit: Hyung Won Chung, OpenAI
Talk at Stanford CS 25, yesterday!

Resources

Get Hands-on with **LTC-based networks:**
github.com/mlech26l/keras-ncp
github.com/raminmh/liquid_time_constant_networks

Get Hands-on with the **closed-form networks:**
github.com/raminmh/CfC

Get Hands-on with **Liquid-S4:**
https://github.com/raminmh/liquid-s4

Get in touch: rhasani@mit.edu

Learning representations from sequences of data requires expressive temporal and structural credit assignment.

Liquid Time-Constant (LTC) networks

1. Linear state-space model

$$d\mathbf{x}(t)/dt = -\mathbf{x}(t)/\tau + \mathbf{S}(t) \qquad \mathbf{S}(t) \in \mathbb{R}^M$$

2. Non-linear synapse Model

$$\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$$

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

"Liquid" = variable

Liquid Networks Limitations

1. Liquid networks' complexity is tied to their numerical solver
2. Learning Long term dependencies due to gradient issues

Solutions

1. Closed-form Continuous-Time Neural Networks [Hasani et al. Nature MI, 2022]

2. Mixed Memory Recurrent Networks [Lechner and Hasani, MemARI NeurIPS 2022]

3. **Bring Liquid Dynamics in Structural State Space Models** [Hasani et al. ICLR 2023]

Structural State-Space Models [Gu et al. ICLR 2022]

State-Space Models (SSM)

State-Space Models

$$\dot{x}(t) = \mathbf{A}\, x(t) + \mathbf{B}\, u(t)$$
$$y(t) = \mathbf{C}\, x(t) + \mathbf{D}\, u(t)$$

$$\mathbf{A}^{(N\times N)},\ \mathbf{B}^{(N\times 1)},\ \mathbf{C}^{(1\times N)} \text{ and } \mathbf{D}^{(1\times 1)}$$

$x(t)$ is an N-dimensional latent state, receiving a 1-dimensional input signal $u(t)$

Discretization

$$x_k = \overline{\mathbf{A}}\, x_{k-1} + \overline{\mathbf{B}}\, u_k$$
$$y_k = \overline{\mathbf{C}}\, x_k$$

$$\overline{\mathbf{A}} = (\mathbf{I} - \frac{\delta t}{2}\mathbf{A})^{-1}(\mathbf{I} + \frac{\delta t}{2}\mathbf{A})$$
$$\overline{\mathbf{B}} = (\mathbf{I} - \frac{\delta t}{2}\mathbf{A})^{-1}\, \delta t\, \mathbf{B}$$
$$\overline{\mathbf{C}} = \mathbf{C}$$

Structural State-Space Models

SSM Convolutional Kernel

$$x_k = \overline{\mathbf{A}}\, x_{k-1} + \overline{\mathbf{B}}\, u_k$$
$$y_k = \overline{\mathbf{C}}\, x_k$$

$$x_0 = \overline{\mathbf{B}}u_0, \quad x_1 = \overline{\mathbf{AB}}u_0 + \overline{\mathbf{B}}u_1, \quad x_2 = \overline{\mathbf{A}}^2\overline{\mathbf{B}}u_0 + \overline{\mathbf{AB}}u_1 + \overline{\mathbf{B}}u_2, \quad \dots$$
$$y_0 = \overline{\mathbf{CB}}u_0, \quad y_1 = \overline{\mathbf{CAB}}u_0 + \overline{\mathbf{CB}}u_1, \quad y_2 = \overline{\mathbf{CA}}^2\overline{\mathbf{B}}u_0 + \overline{\mathbf{CAB}}u_1 + \overline{\mathbf{CB}}u_2, \quad \dots$$

The mapping $u_k \to y_k$ can now can be formulated into a convolutional kernel explicitly:

$$y_k = \overline{\mathbf{CA}}^k\overline{\mathbf{B}}u_0 + \overline{\mathbf{CA}}^{k-1}\overline{\mathbf{B}}u_1 + \dots \overline{\mathbf{CAB}}u_{k-1} + \overline{\mathbf{CB}}u_k$$
$$y = \overline{\mathbf{K}} * u$$

$$\overline{\mathbf{K}} \in \mathbb{R}^L := \mathcal{K}_L(\overline{\mathbf{C}}, \overline{\mathbf{A}}, \overline{\mathbf{B}}) := \left(\overline{\mathbf{CA}}^i\overline{\mathbf{B}}\right)_{i\in[L]} = \left(\overline{\mathbf{CB}}, \overline{\mathbf{CAB}}, \dots, \overline{\mathbf{CA}}^{L-1}\overline{\mathbf{B}}\right)$$

Diagonal Plus Low Rank decomposition (DPLR) + Hippo Initialization will do the job!

$$\boldsymbol{A}_{nk} = -\begin{cases} (2n+1)^{\frac{1}{2}}(2k+1)^{\frac{1}{2}} & n > k \\ n+1 & n = k \\ 0 & n < k \end{cases}$$

$$\boldsymbol{B}_n = (2n+1)^{\frac{1}{2}} \quad \boldsymbol{P}_n = (n+1/2)^{\frac{1}{2}}$$

(HiPPO-LegS matrix used in S4)

(4)

$$\boldsymbol{A}^{(N)}_{nk} = -\begin{cases} (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n > k \\ \frac{1}{2} & n = k \\ (n+\frac{1}{2})^{1/2}(k+\frac{1}{2})^{1/2} & n < k \end{cases}$$

$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \boldsymbol{P}\boldsymbol{P}^\top, \qquad \boldsymbol{A}^{(D)} := \mathrm{eig}(\boldsymbol{A}^{(N)})$$

(Normal / diagonal plus low-rank form)

Gu et al. ICLR 2022

Liquid Structural State-Space Models

Liquid S4

$$\dot{x}(t) = \left[\mathbf{A} + \mathbf{B}\, u(t)\right] x(t) + \mathbf{B}\, u(t)$$
$$y(t) = \mathbf{C}\, x(t)$$

$x(t)$ is an N-dimensional latent state, receiving a 1-dimensional input signal $u(t)$, and computing a 1-dimensional output signal $y(t)$. $\mathbf{A}^{(N\times N)}$, $\mathbf{B}^{(N\times 1)}$, and $\mathbf{C}^{(1\times N)}$. Note that $\mathbf{D}$ is set to zero for simplicity.

Hasani et al. ICLR 2023

Liquid Structural State-Space Models

Liquid S4

A forward Euler Transformation of the Linear LTC

$$x_k = \left(\overline{\mathbf{A}} + \overline{\mathbf{B}}\, u_k\right) x_{k-1} + \overline{\mathbf{B}}\, u_k$$
$$y_k = \overline{\mathbf{C}}\, x_k$$

$\overline{\mathbf{A}} = \mathbf{I} + \frac{\delta t}{2}\mathbf{A}$, $\overline{\mathbf{B}} = \delta t\, \mathbf{B}$, and $\overline{\mathbf{C}} = \mathbf{C}$

$$x_0 = \overline{\mathbf{B}} u_0,$$
$$y_0 = \overline{\mathbf{CB}} u_0$$
$$x_1 = \overline{\mathbf{AB}} u_0 + \overline{\mathbf{B}} u_1 + \overline{\mathbf{B}}^2 u_0 u_1,$$
$$y_1 = \overline{\mathbf{CAB}} u_0 + \overline{\mathbf{CB}} u_1 + \overline{\mathbf{CB}}^2 u_0 u_1$$
$$x_2 = \overline{\mathbf{A}}^2\overline{\mathbf{B}} u_0 + \overline{\mathbf{AB}} u_1 + \overline{\mathbf{B}} u_2 + \overline{\mathbf{AB}}^2 u_0 u_1 + \overline{\mathbf{AB}}^2 u_0 u_2 + \overline{\mathbf{B}}^2 u_1 u_2 + \overline{\mathbf{B}}^3 u_0 u_1 u_2$$
$$y_2 = \overline{\mathbf{CA}}^2\overline{\mathbf{B}} u_0 + \overline{\mathbf{CAB}} u_1 + \overline{\mathbf{CB}} u_2 + \overline{\mathbf{CAB}}^2 u_0 u_1 + \overline{\mathbf{CAB}}^2 u_0 u_2 + \overline{\mathbf{CB}}^2 u_1 u_2 + \overline{\mathbf{CB}}^3 u_0 u_1 u_2$$
$$\dots$$

$$y_k = \overline{\mathbf{CA}}^k\overline{\mathbf{B}} u_0 + \overline{\mathbf{CA}}^{k-1}\overline{\mathbf{B}} u_1 + \dots \overline{\mathbf{CAB}} u_{k-1} + \overline{\mathbf{CB}} u_k + \sum_{p=2}^{\mathcal{P}} \sum_{u_i u_{i+1} \dots u_p \in \Pi(k+1,p)} \overline{\mathbf{CA}}^{(k+1-p-i)}\overline{\mathbf{B}}^p u_i u_{i+1} \dots u_p$$

$$\text{for } i \in \mathbb{Z} \text{ and } i \geq 0, \quad \rightarrow \quad y = \overline{\mathbf{K}} * u + \overline{\mathbf{K}}_{\text{liquid}} * u_{\text{correlations}}$$

Hasani et al. ICLR 2023

Liquid Structural State-Space Models

Example: Liquid S4

$$\overline{\mathbf{K}}_{\text{liquid}} * u_{\text{correlations}} = \begin{pmatrix} \overline{\mathbf{CA}}^{(k-1)}\overline{\mathbf{B}}^{2}, \\ \vdots \\ \overline{\mathbf{CB}}^{2}, \\ \vdots \\ \overline{\mathbf{CA}}^{(k-2)}\overline{\mathbf{B}}^{3}, \\ \vdots \\ \overline{\mathbf{CB}}^{3}, \\ \vdots \\ \overline{\mathbf{CA}}^{(k-3)}\overline{\mathbf{B}}^{4}, \\ \vdots \\ \overline{\mathbf{CB}}^{4} \end{pmatrix}^{T} * \begin{bmatrix} u_0 u_1 \\ \vdots \\ u_{k-1}u_k \\ \vdots \\ u_0 u_1 u_2 \\ \vdots \\ u_{k-2}u_{k-1}u_k \\ \vdots \\ u_0 u_1 u_2 u_3 \\ \vdots \\ u_{k-3}u_{k-2}u_{k-1}u_k \end{bmatrix}$$

Here, $u_{\text{correlations}}$ is a vector of length $\binom{k+1}{2} + \binom{k+1}{3} + \binom{k+1}{4}$, and the kernel $\overline{\mathbf{K}}_{\text{liquid}} \in \mathbb{R}^{\binom{k+1}{2}+\binom{k+1}{3}+\binom{k+1}{4}}$. This additional kernel takes the temporal correlation of incoming input samples into consideration.

Hasani et al. ICLR 2023

Liquid Structural State-Space Models

Liquid S4 Kernel

$$y_k = \overline{\mathbf{CA}}^k \overline{\mathbf{B}} u_0 + \overline{\mathbf{CA}}^{k-1} \overline{\mathbf{B}} u_1 + \ldots \overline{\mathbf{CAB}} u_{k-1} + \overline{\mathbf{CB}} u_k +$$

$$\sum_{p=2}^{\mathcal{P}} \sum_{u_i u_{i+1} \ \ldots \ u_p \in \Pi(k+1,p)} \overline{\mathbf{CA}}^{(k+1-p-i)} \overline{\mathbf{B}}^p u_i u_{i+1} \ \ldots \ u_p$$

$$\text{for } i \in \mathbb{Z} \text{ and } i \geq 0, \quad \rightarrow \quad y = \overline{\mathbf{K}} * u + \overline{\mathbf{K}}_{\text{liquid}} * u_{\text{correlations}}$$

$$\overline{\mathbf{K}}_{\text{liquid}} \in \mathbb{R}^{\tilde{L}} := \mathcal{K}_L(\overline{\mathbf{C}}, \overline{\mathbf{A}}, \overline{\mathbf{B}}) := \left(\overline{\mathbf{CA}}^{(\tilde{L}-i-p)} \overline{\mathbf{B}}^p\right)_{i \in [\tilde{L}],\ p \in [\mathcal{P}]} = \left(\overline{\mathbf{CA}}^{\tilde{L}-2} \overline{\mathbf{B}}^2, \ldots, \overline{\mathbf{CB}}^p\right)$$

Hasani et al. ICLR 2023

Liquid Structural State-Space Models

Liquid-S4 Kernel

Algorithm 1 LIQUID-S4 KERNEL - The S4 convolution kernel (highlighted in black) is used from Gu et al. (2022a) and Gu et al. (2022b). Liquid kernel computation is highlighted in purple.

Input: S4 parameters $\boldsymbol{\Lambda}, \boldsymbol{P}, \boldsymbol{B}, \boldsymbol{C} \in \mathbb{C}^N$, step size Δ, liquid kernel order $\mathcal{P}$, inputs seq length L, liquid kernel sequence length $\tilde{L}$

Output: SSM convolution kernel $\overline{\boldsymbol{K}} = \mathcal{K}_L(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ and SSM liquid kernel $\overline{\boldsymbol{K}}_{liquid} = \mathcal{K}_{\tilde{L}}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ for $\boldsymbol{A} = \boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{P}^*$ (Eq. 6)

$\widetilde{\boldsymbol{C}} \leftarrow \left(\boldsymbol{I} - \overline{\boldsymbol{A}}^L\right)^* \overline{\boldsymbol{C}}$ ▷ Truncate SSM generating function (SSMGF) to length L

$\begin{bmatrix} k_{00}(\omega) & k_{01}(\omega) \\ k_{10}(\omega) & k_{11}(\omega) \end{bmatrix} \leftarrow \left[\widetilde{\boldsymbol{C}}\ \boldsymbol{P}\right]^* \left(\frac{2}{\Delta}\frac{1-\omega}{1+\omega} - \boldsymbol{\Lambda}\right)^{-1} [\boldsymbol{B}\ \boldsymbol{P}]$ ▷ Black-box Cauchy kernel

$\hat{\boldsymbol{K}}(\omega) \leftarrow \frac{2}{1+\omega}\left[k_{00}(\omega) - k_{01}(\omega)(1+k_{11}(\omega))^{-1}k_{10}(\omega)\right]$ ▷ Woodbury Identity

$\hat{\boldsymbol{K}} = \{\hat{\boldsymbol{K}}(\omega) : \omega = \exp(2\pi i \frac{k}{L})\}$ ▷ Evaluate SSMGF at all roots of unity $\omega \in \Omega_L$

$\overline{\boldsymbol{K}} \leftarrow \mathsf{iFFT}(\hat{\boldsymbol{K}})$ ▷ Inverse Fourier Transform

if Mode == KB **then** ▷ Liquid-S4 Kernel as shown in Eq. 14

  for p in $\{2, \ldots, \mathcal{P}\}$ **do**

    $\overline{\boldsymbol{K}}_{\text{liquid}=p} = \left[\overline{\boldsymbol{K}}_{(L-\tilde{L},L)} \odot \overline{\boldsymbol{B}}^{p-1}_{(L-\tilde{L},L)}\right] * \mathbf{J}_{\tilde{L}}$ ▷ $\mathbf{J}_{\tilde{L}}$ is a backward identity matrix

    $\overline{\boldsymbol{K}}_{\text{liquid}}.\text{append}(\overline{\boldsymbol{K}}_{\text{liquid}=p})$

  end for

else if Mode == PB **then** ▷ Liquid-S4 Kernel of Eq. 14 with $\overline{\boldsymbol{A}}$ reduced to Identity.

  for p in $\{2, \ldots, \mathcal{P}\}$ **do**

    $\overline{\boldsymbol{K}}_{\text{liquid}=p} = \overline{\boldsymbol{C}} \odot \overline{\boldsymbol{B}}^{p-1}_{(L-\tilde{L},L)}$

    $\overline{\boldsymbol{K}}_{\text{liquid}}.\text{append}(\overline{\boldsymbol{K}}_{\text{liquid}=p})$

  end for

end if