Liquid Time-Constant Network

Ramin Hasani
CSAIL, Massachusetts Institute of Technology, USA

Simons SMS Workshop, UC Berkeley
March 22th, 2021

What is a time-continuous neural network?

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

Number of layers

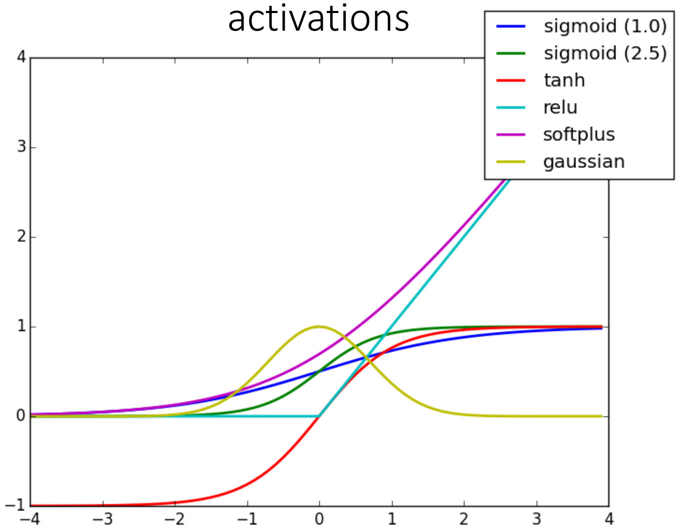
Width

activations

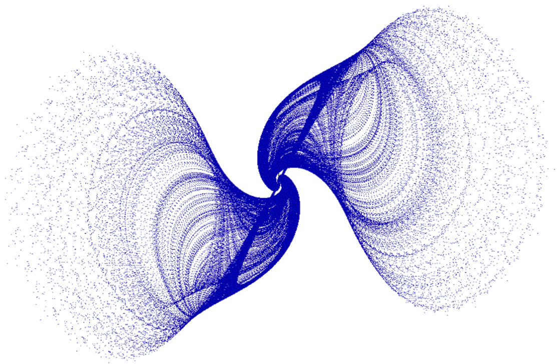
Hidden state

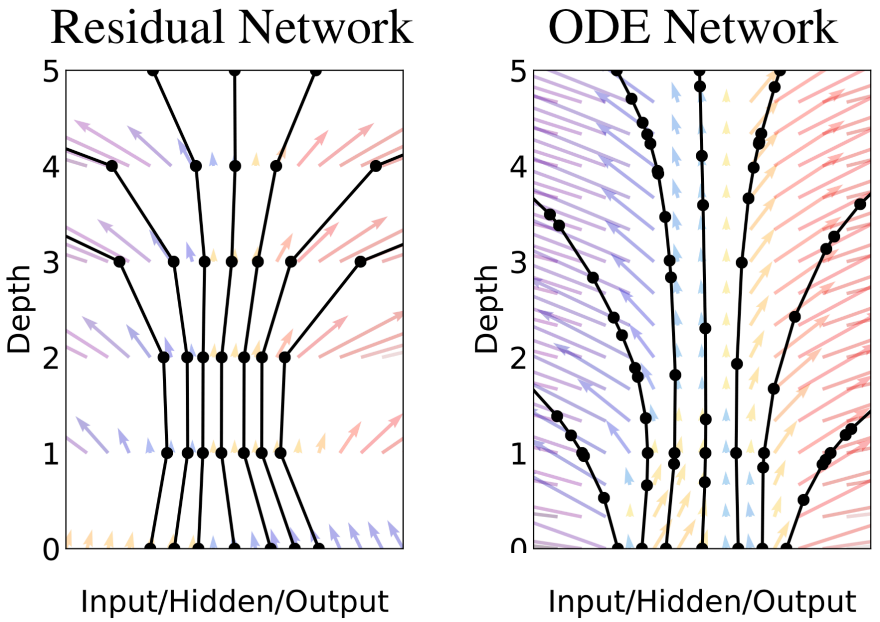
Inputs

Model parameters



Dynamical systems





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

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

He et al.
CVPR 2016

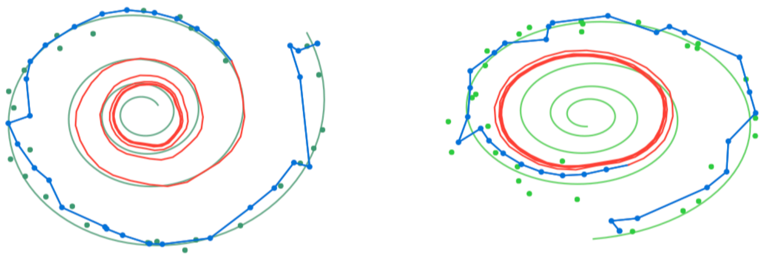
Chen et al.
NeurIPS 2018

Figure Credit: Chen et al. NeurIPS 2018

What is a time-continuous neural network?

Standard Recurrent Neural Network (RNN)
Hopfield 1982

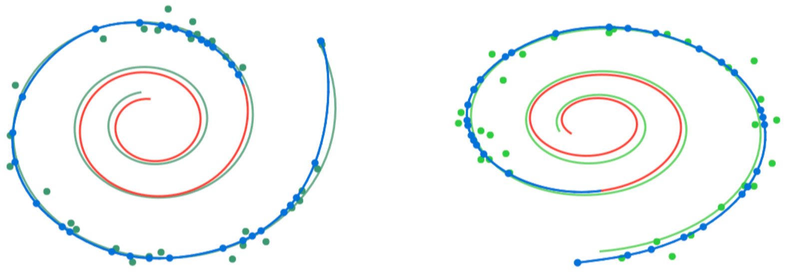
$$x(t+1) = f(x(t), I(t), t\,;\theta)$$



(a) Recurrent Neural Network

Neural ODE
Chen et al. NeurIPS, 2018

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



(b) Latent Neural Ordinary Differential Equation

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

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

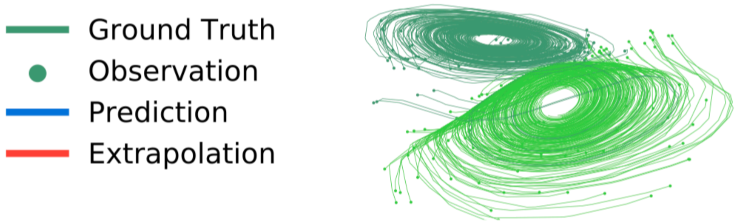


Figure Credit: Chen et al. NeurIPS 2018

Time-continuous neural networks
How to implement them?

$$d\mathbf{x}(t)/dt = f(\mathbf{x}(t), t, \theta)$$

Numerical ODE solvers

$$\frac{d\mathbf{x}(t)}{dt} \approx \frac{\mathbf{x}(t+\delta t) - \mathbf{x}(t)}{\delta t} \approx f(\mathbf{x}(t), t, \theta)$$

Forward-pass $$\mathbf{x}(t+\delta t) = \mathbf{x}(t) + \delta t\, f(\mathbf{x}(t), t, \theta)$$

Choice of the way we do an integration step determines forward pass complexity

Time-continuous neural networks
How to train them?

Memory Complexity $O(1)$ Per layer of f

Adjoint Sensitivity Method

[Pontryagin et al. 1962, Chen et al. NeurIPS, 2018]

Loss function $$L\big(\boldsymbol{x}(t_1)\big) = L(\mathrm{ODESolve}(\boldsymbol{x}(t_0), f, t_0, t_1, \theta))$$

Neural ODE $$\frac{d\boldsymbol{x}(t)}{dt} = f(\boldsymbol{x}(t), t, \theta)$$

Adjoint State $$\boldsymbol{a}(t) = \frac{\partial L}{\partial x(t)}$$

$$\frac{d\boldsymbol{a}(t)}{dt} = -\boldsymbol{a}(t)^{\mathrm{T}} \frac{\partial f(\boldsymbol{x}(t), t, \theta)}{\partial \boldsymbol{x}}$$

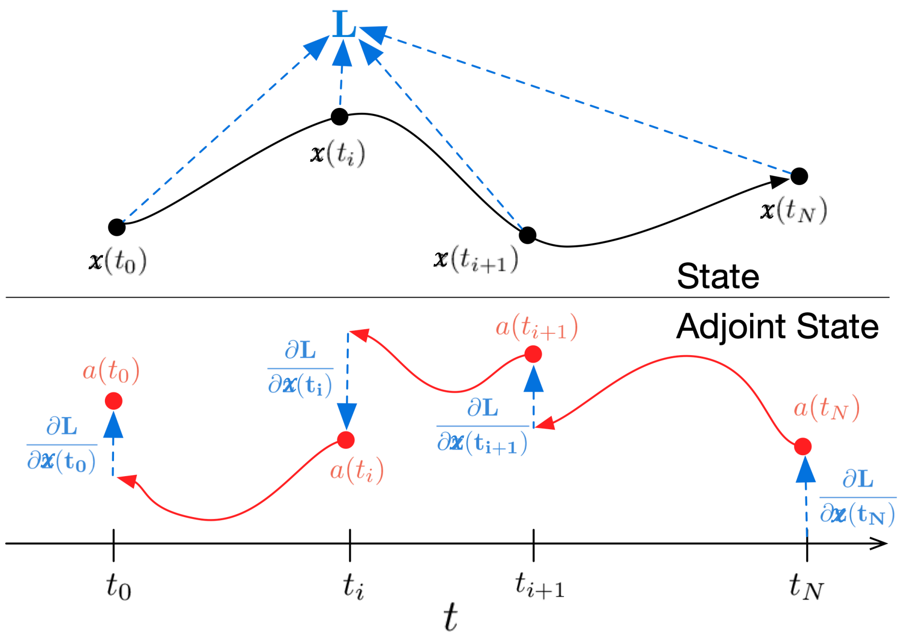


Figure Credit: Chen et al. NeurIPS 2018

Time-continuous neural networks
How to train them?

Memory Complexity $O(L * T)$ Per layer of f

Depth sequence length

Backpropagation through-time (BPTT)

[Werbos, 1990, Gholami et. al, 2019, Lechner et al. 2019, Lechner et al. 2020, Hasani et al. 2020]

Perform a forward-pass

$$\mathbf{x}(t + \delta t) = \mathbf{x}(t) + \delta t\, f(\mathbf{x}(t), t, \theta)$$

Compute gradients through the ODE solver

$$\mathrm{d}\Theta = \left[\frac{dL}{dx(t+\delta t)}, \frac{dx(t+\delta t)}{dx(t)}, \frac{dx(t+\delta t)}{df}, \frac{df}{dx(t)}, \frac{df}{dt}, \frac{df}{d\theta}\right]$$

Update parameters

$$\Theta_{new} \leftarrow \Theta_{old} + \gamma\, \mathrm{d}\Theta$$

Time-continuous neural networks
Better Stay with BPTT

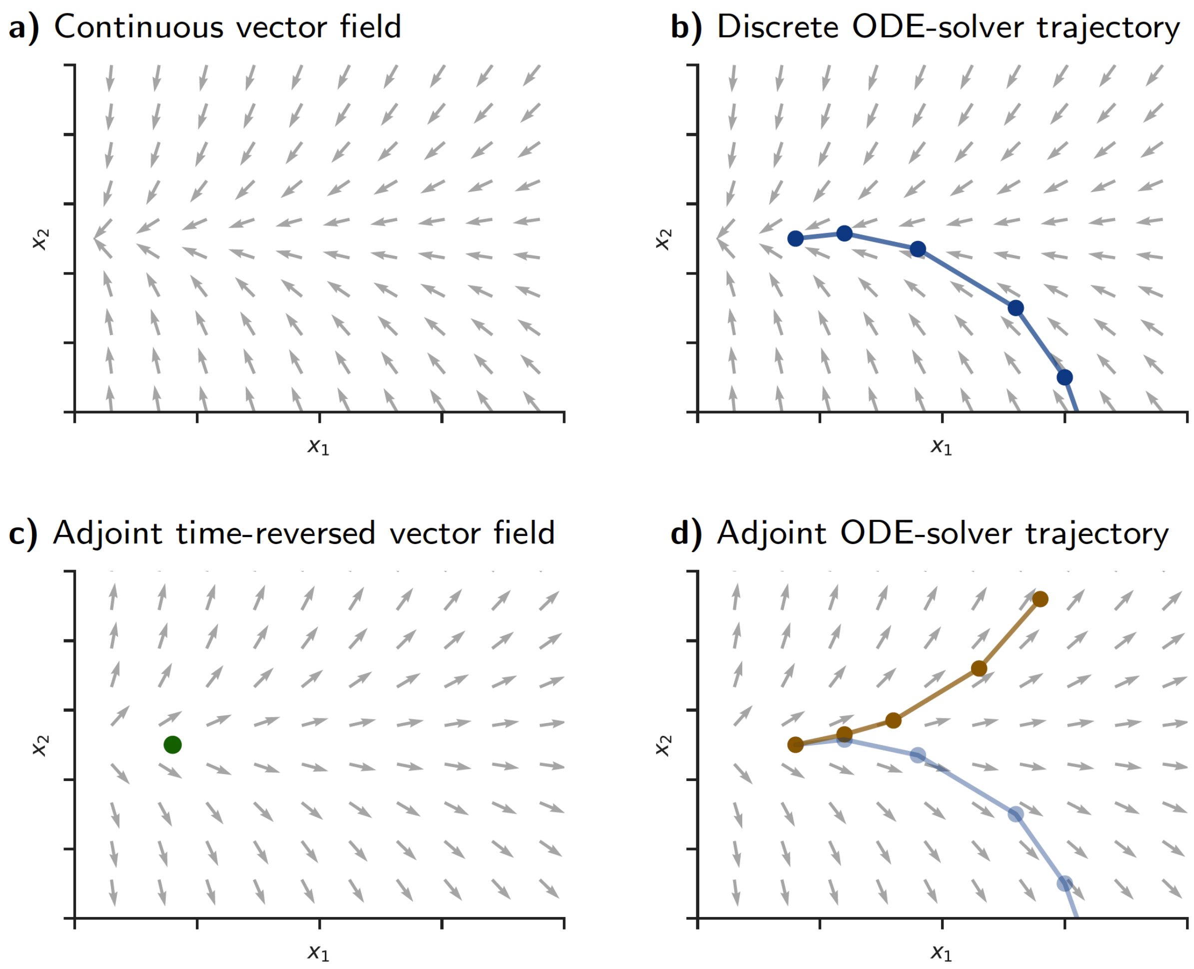


Table 1: Complexity of the vanilla BPTT algorithm compared to the adjoint method, for a single layer neural network f

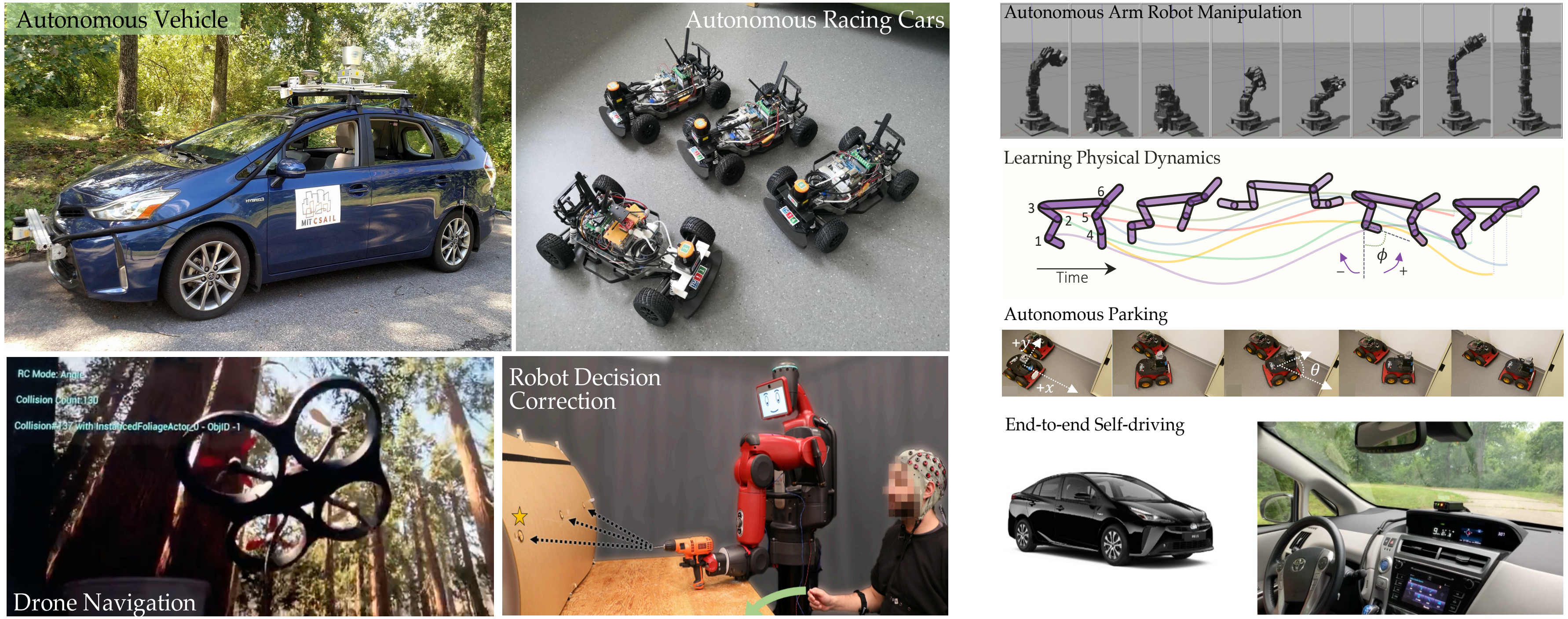
	Vanilla BPTT	**Adjoint**
Time	$O(L \times T \times 2)$	$O((L_f + L_b) \times T)$
Memory	$O(L \times T)$	$\mathbf{O(1)}$
Depth	$O(L)$	$O(L_b)$
FWD acc	High	High
BWD acc	**High**	Low

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.

Lechner & Hasani 2020
Zhuang et al. 2020

Neural ODE
Chen et al. 2018

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



Can Neural ODEs be as expressive as advanced RNNs in real-world applications?

Liquid Time-Constant Networks

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

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

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

"Liquid" = variable

LTCs have stable state and time-constant

System time-constant

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

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

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

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

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

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

Liquid Time-Constant Networks are Universal Approximators

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

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

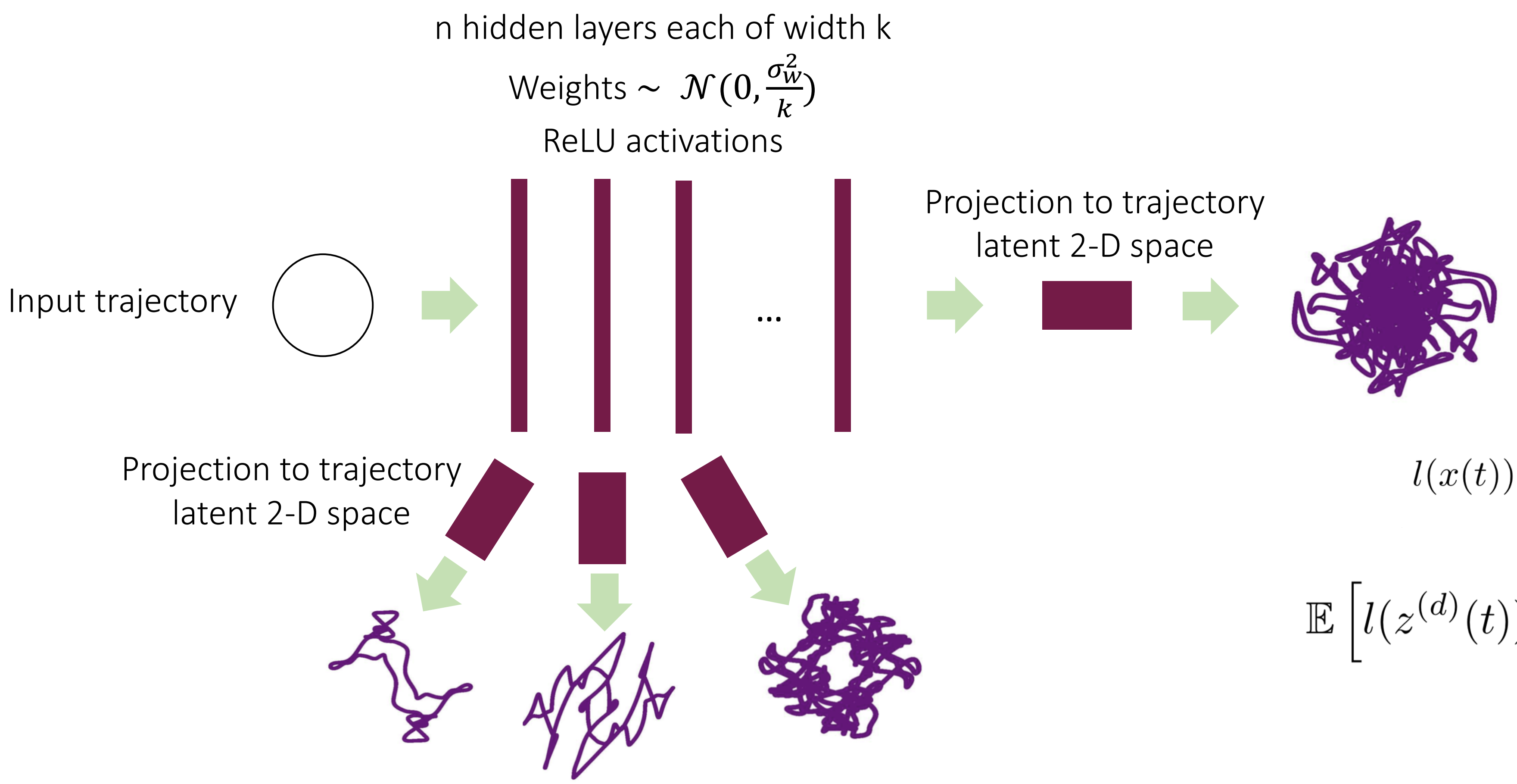
$$max_{t \in I}|\boldsymbol{x}(t) - \boldsymbol{u}(t)| < \epsilon \qquad (6)$$

Expressivity

Defining a better measure

Trajectory Length as a measure of expressivity of Deep networks

[Raghu et al. ICML 2017]



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

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

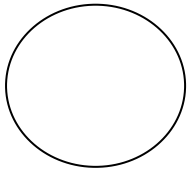
[Raghu et al. ICML 2017]

Expressivity

Trajectory length as a measure of expressivity

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

Input trajectory



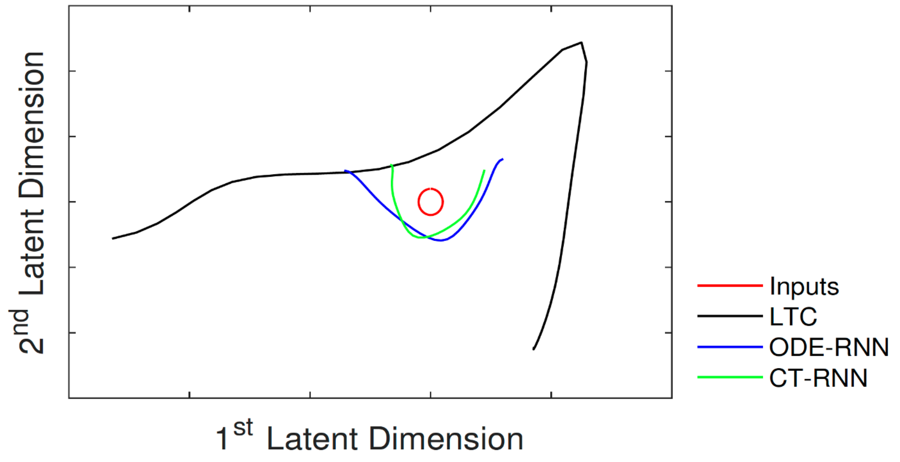
LTC
CT-RNN
Neural ODE

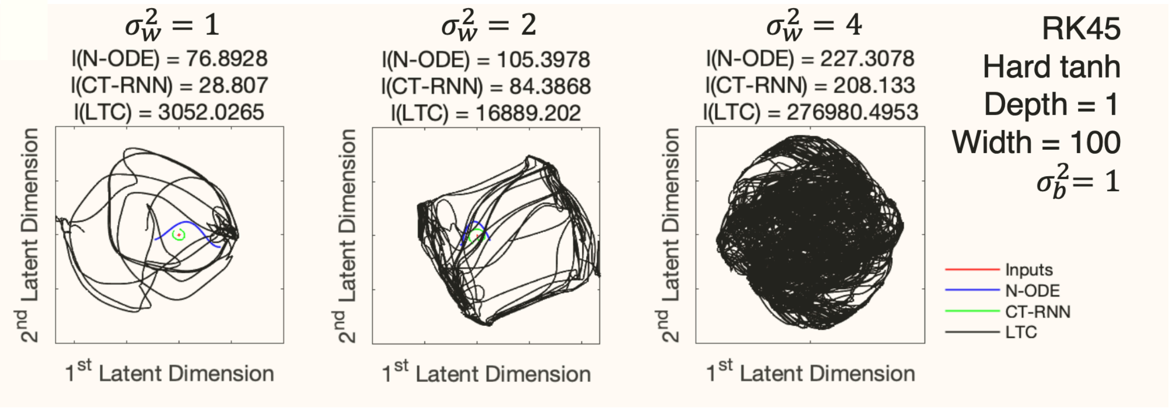
Weights $\sim \mathcal{N}(0, \frac{\sigma_w^2}{k})$

activation functions:
ReLU, tanh, logistic sigmoid

Projection to a latent trajectory 2-D space

PCA

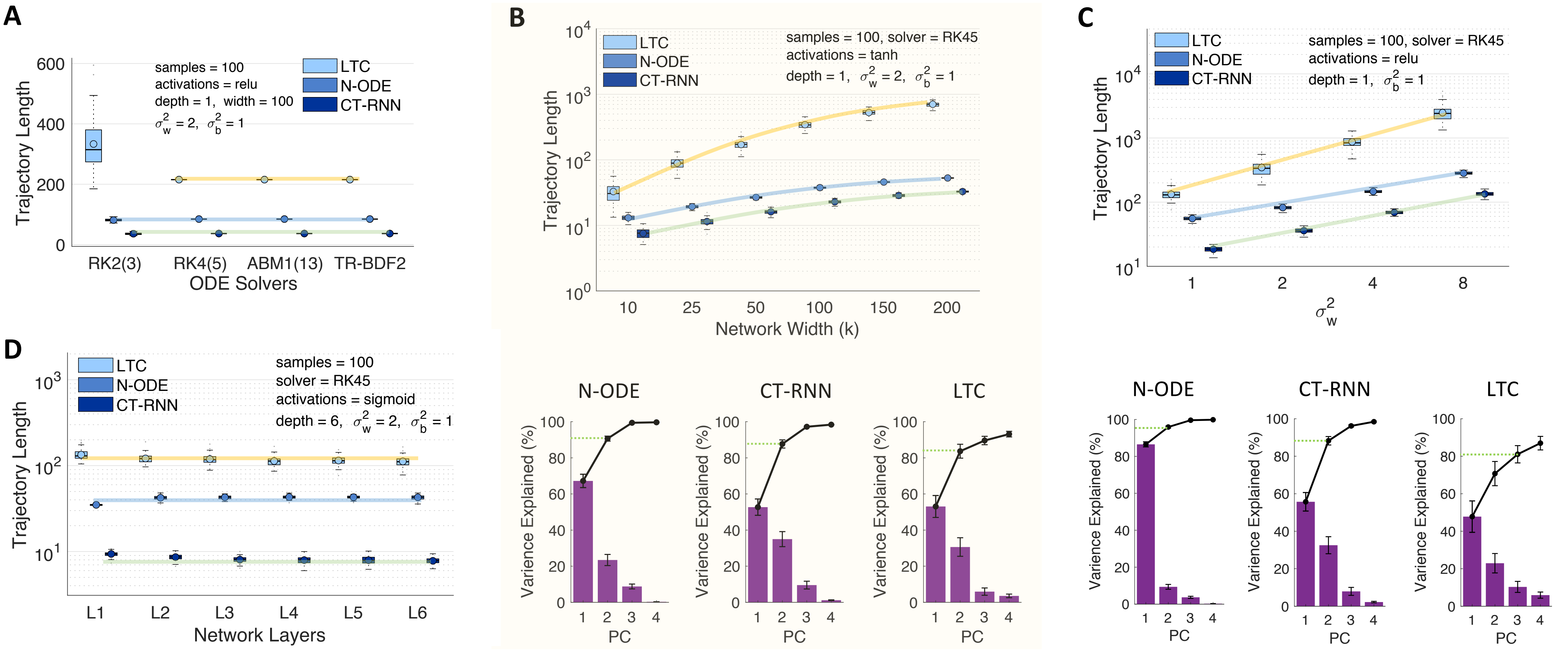




PCA = Principle Component Analysis

Expressivity

Trajectory length as a measure of expressivity



Expressivity

Trajectory length lower bound

Bias scale

Depth

Weight scale

Width

Number of discretization steps

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

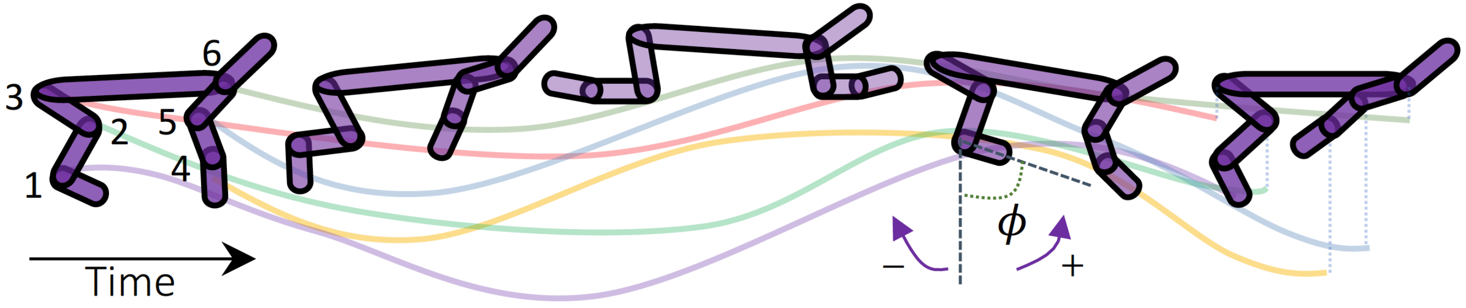
CT-RNN: $$\mathbb{E}\left[l(z^{(d)}(t))\right] \geq O\left(\frac{(\sigma_w-\sigma_b)\sqrt{k}}{\sqrt{\sigma_w^2+\sigma_b^2+k\sqrt{\sigma_w^2+\sigma_b^2}}}\right)^{d\times L} l(I(t))$$

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

System's dynamic time-scale

Performance

LTCs in modeling physical dynamics



17 input observations | 6 control outputs | ϕ = joint angle

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

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

Performance

LTCs in modeling irregularly sampled data

Table 5: Person activity, 2nd setting

Algorithm	Accuracy
RNN Δ_t * [47]	0.797± 0.003
RNN-Decay* [38]	0.800± 0.010
RNN GRU-D* [5]	0.806± 0.007
RNN-VAE* [47]	0.343± 0.040
Latent ODE (D enc.)*	0.835± 0.010
ODE-RNN *	0.829 ± 0.016
Latent ODE(C enc.)*	0.846 ± 0.013
LTC (ours)	**0.882** ± **0.005**

Note: Accuracy values for algorithms indicated by $*$, are taken directly from [47]. RNN Δ_t = classic RNN + input delays. RNN-Decay = RNN with exponential decay on the hidden states. GRU-D = gated recurrent unit + exponential decay + input imputation. D-enc. = RNN encoder. C-enc = ODE encoder. n=5

[5] Che et al. Nature Scientific Reports, 2018
[38] Moser et al. Arxiv, 2017
[47] Rubanova et al. NeurIPS 2019

Performance

LTCs in modeling real-life time series data

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

Dataset	Metric	LSTM [28]	CT-RNN [47]	Neural ODE [6]	CT-GRU [38]	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

[28] Hochreiter et al. 1997
[6] Chen et al. NeurIPS, 2018
[38] Moser et al. Arxiv, 2017
[47] Rubanova et al. NeurIPS 2019

Summary

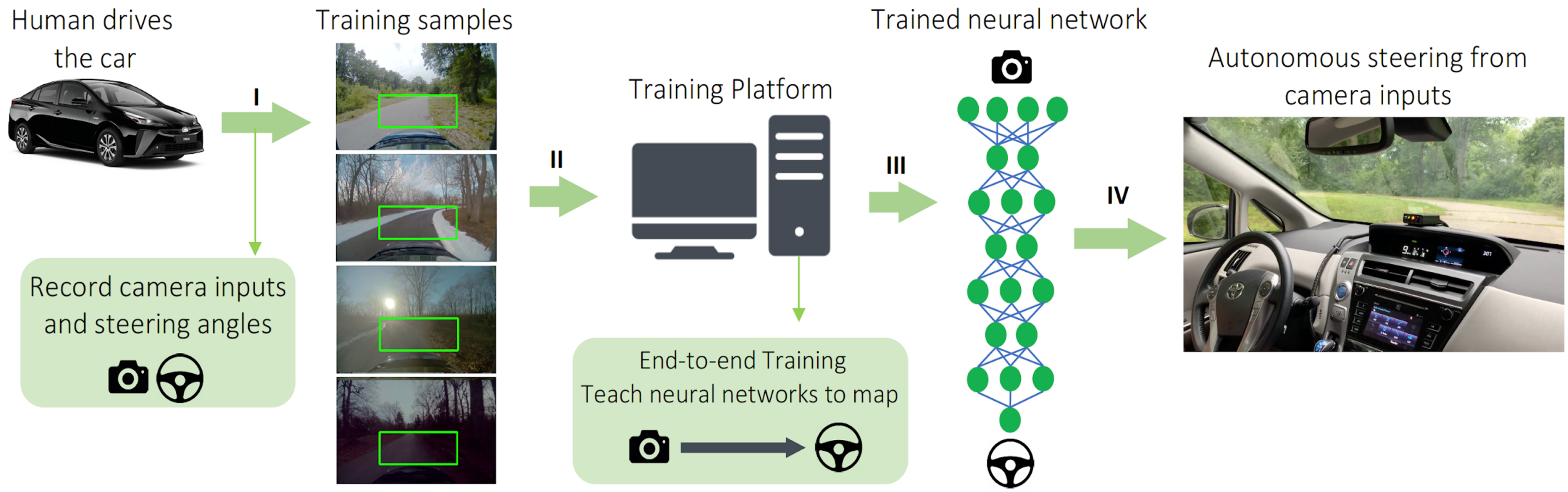
✓ A novel time-continuous neural networks for efficient time-series modelling

- LTCs are universal approximators
- LTCs are stable dynamical systems
- LTCs show better degrees of expressivity
- They can vary their behavior even post-training
- Learning irregularly-sampled data
- Their effectiveness in modeling continuous-time processes.

What can we do with LTCs in real world?

Performance

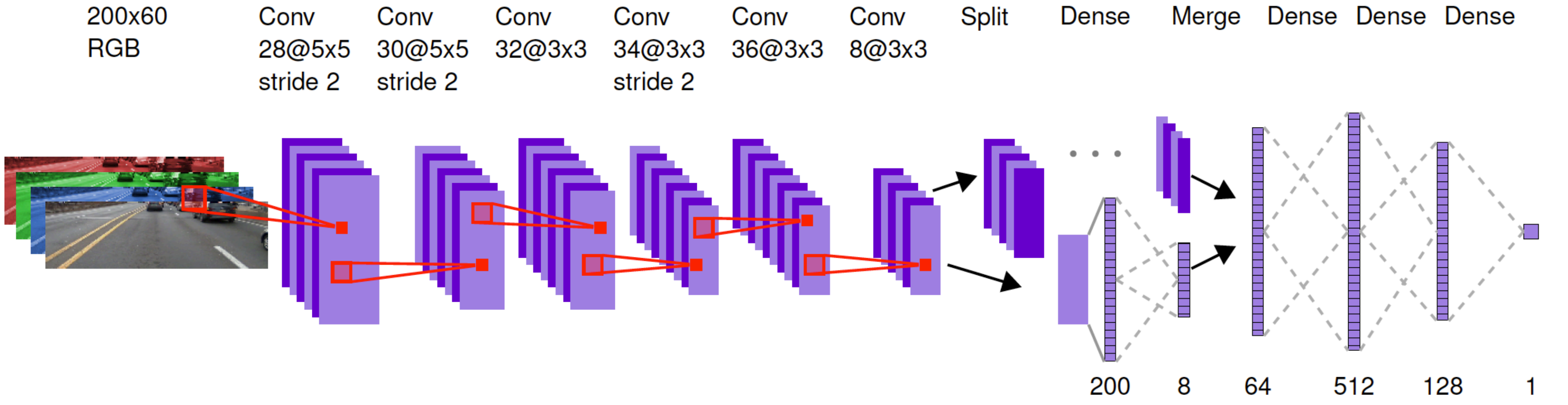
High-fidelity autonomy by LTCs
end-to-end learning



Lechner* & Hasani* et al. Nature Machine Intelligence, 2020

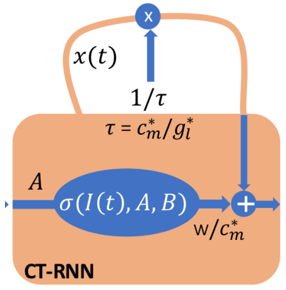
LTCs: Performance

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

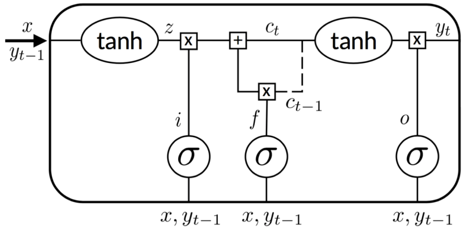


What if we replace the fully connected layers by a recurrent neural network?

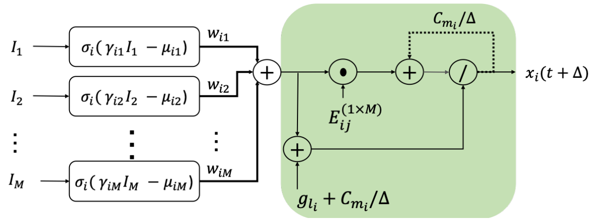
ODE-RNN



LSTMs



LTC-based Networks?



Lechner* & Hasani* et al. Nature Machine Intelligence, 2020

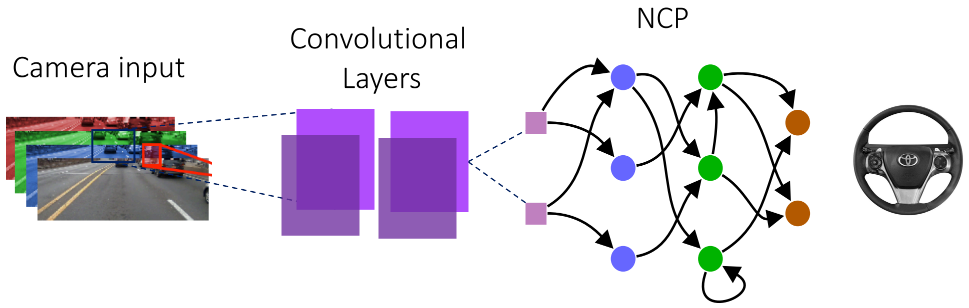
LTCs: Performance

High-fidelity autonomy by LTCs

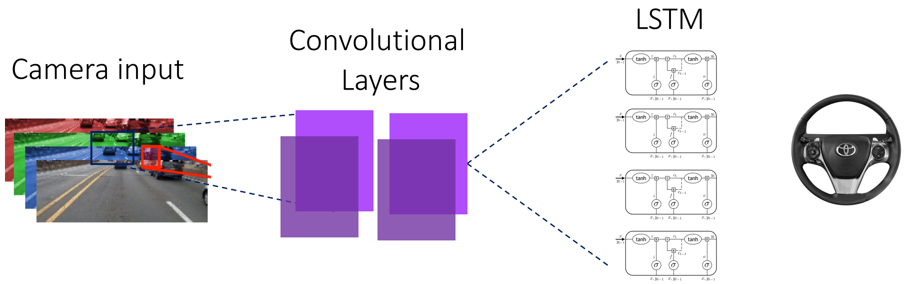
end-to-end learning of Neural Circuit Policies (NCP)

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

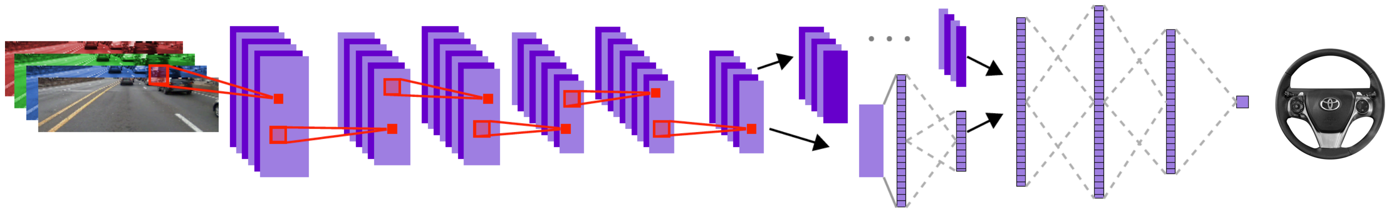
NCPs



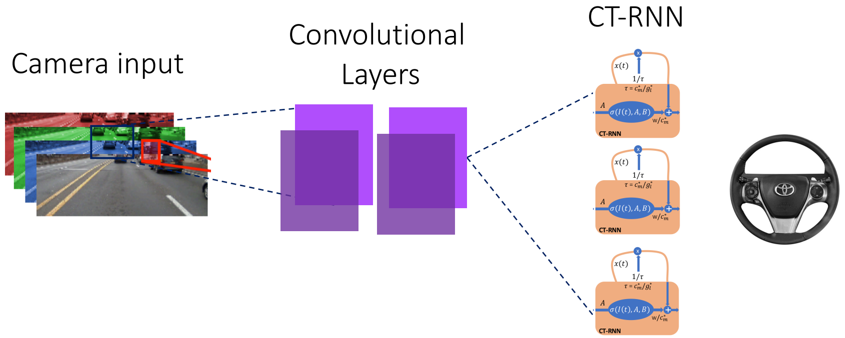
LSTMs



Convolutional Neural Networks (CNNs)



CT-RNNs



Lechner* & Hasani* et al. Nature Machine Intelligence, 2020

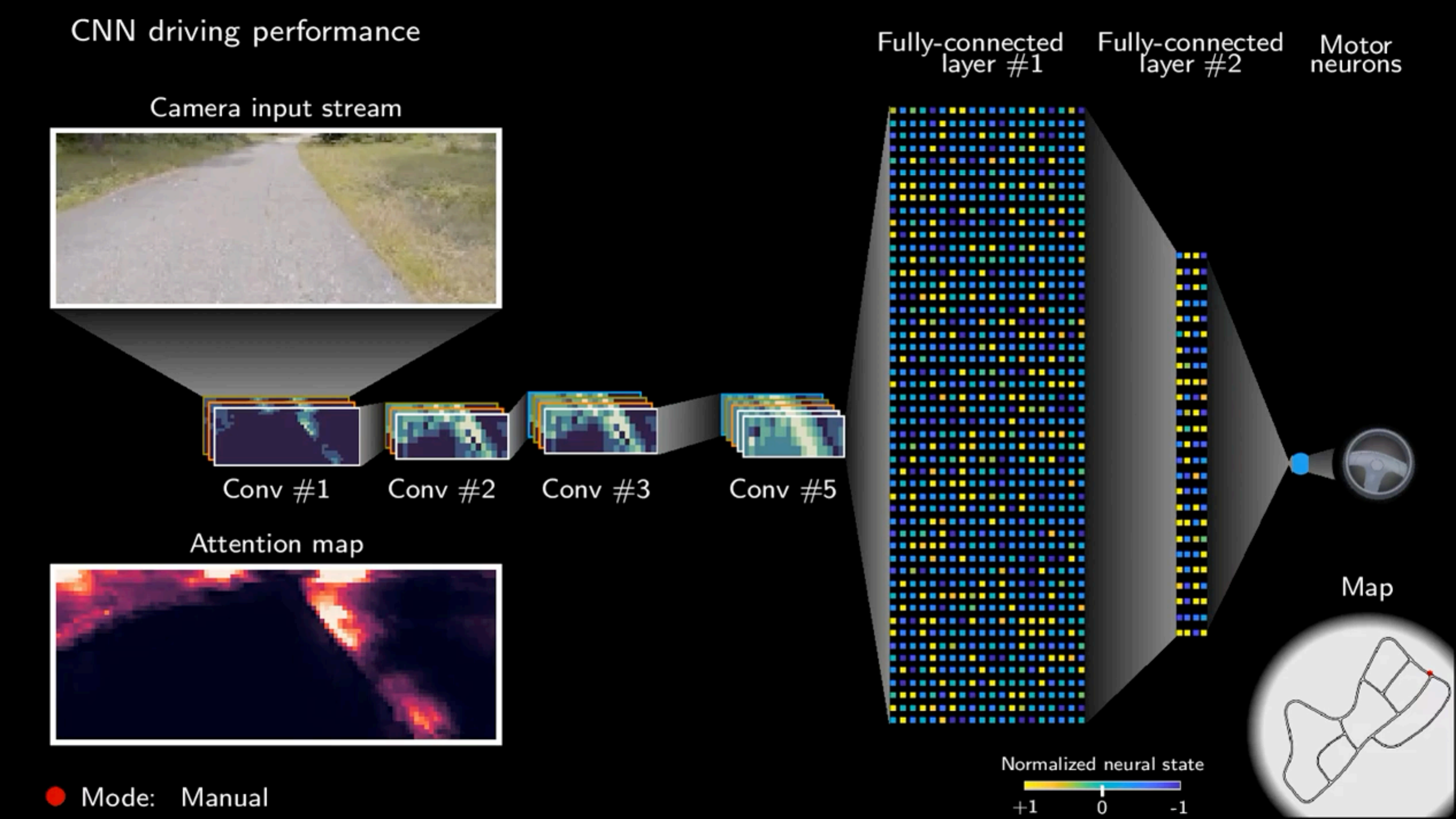
LTCs: Performance

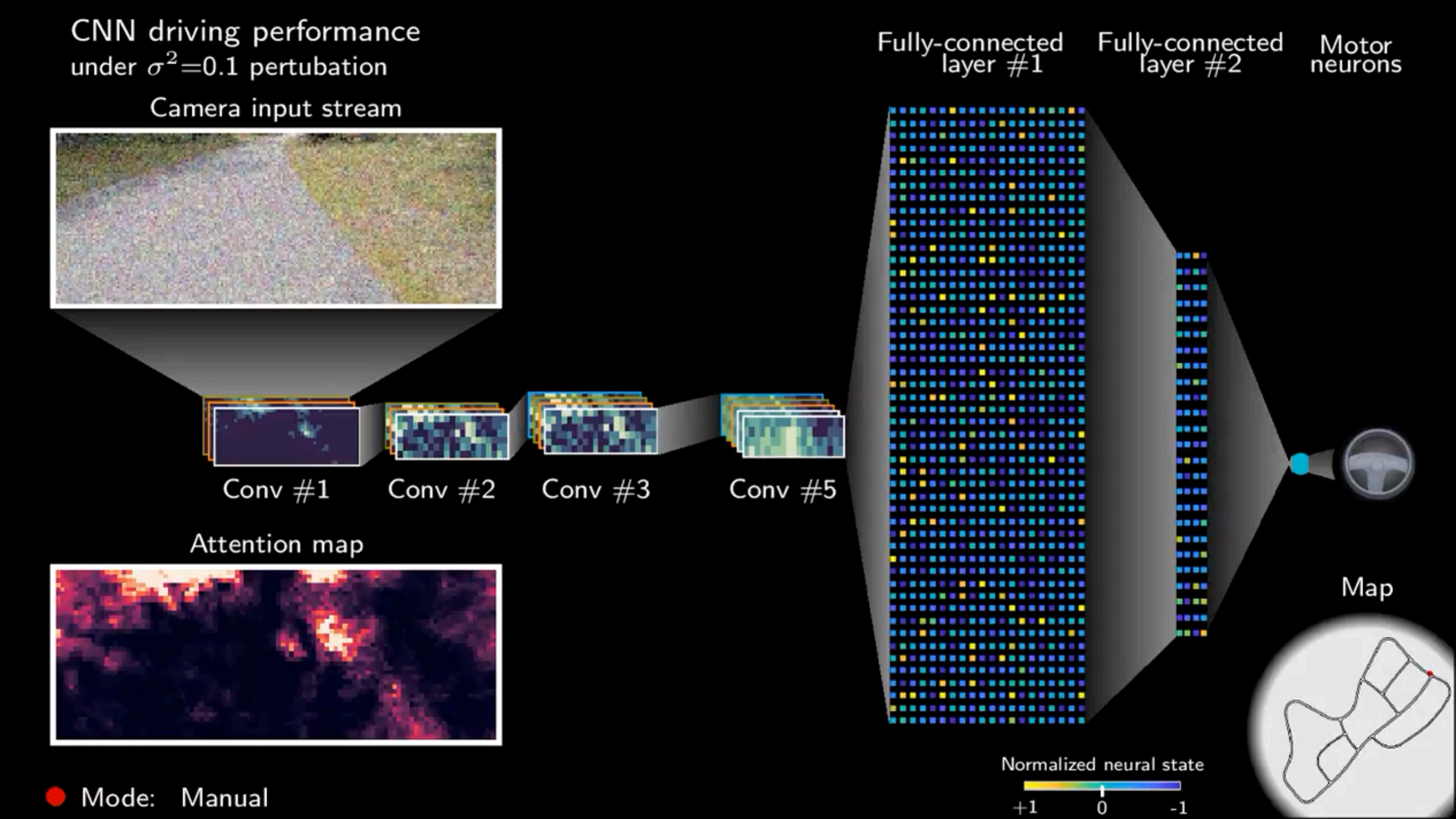
High-fidelity autonomy by LTCs

Parameter efficiency

Model	Conv layers Param	RNN neurons	RNN synapses	RNN trainable param
CNN	5,068,900	-	-	-
CT-RNN	79,420	64	6112	6273
LSTM	79,420	64	24640	24897
NCP	79,420	**19**	**253**	**1065**

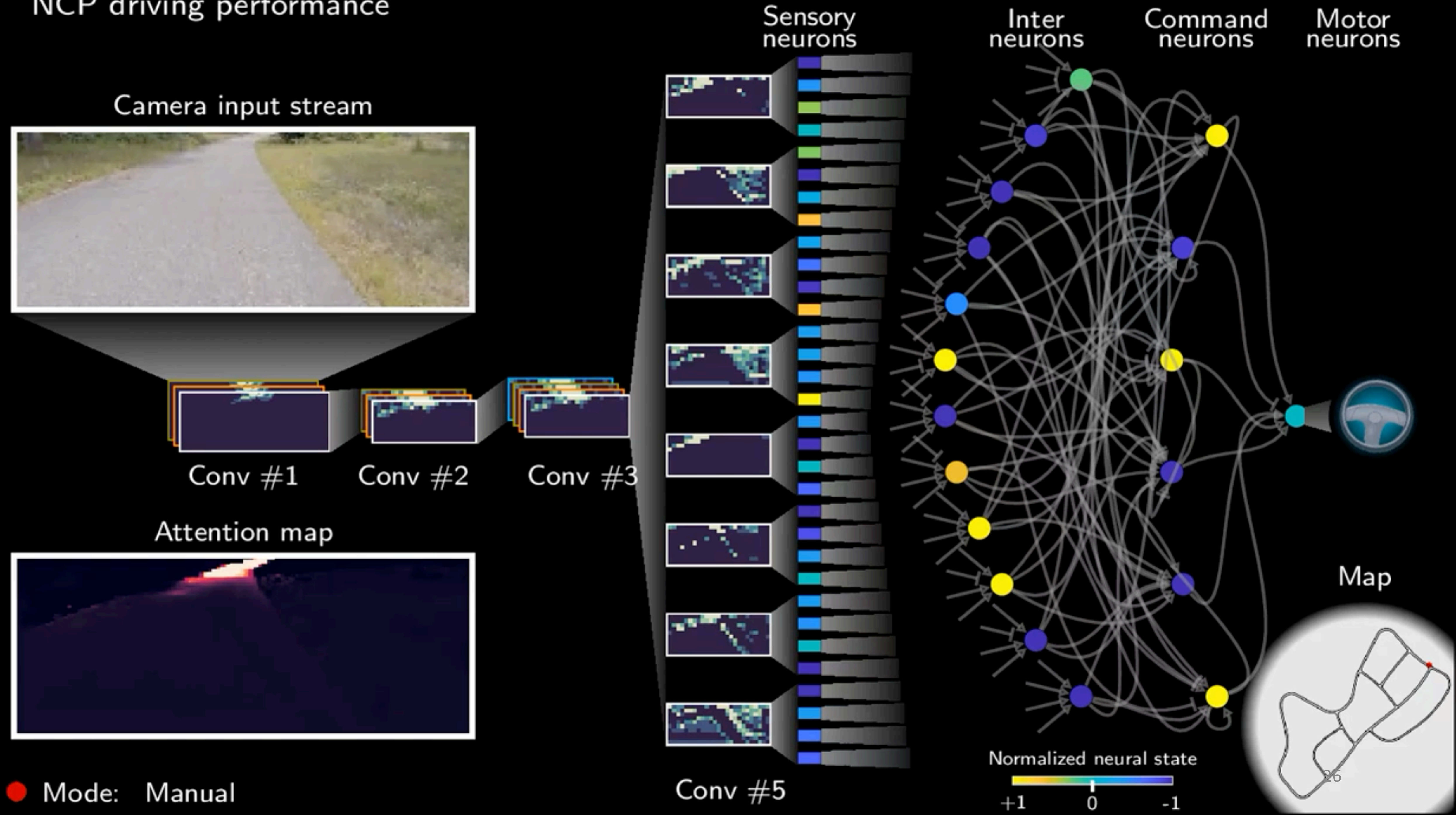
Lechner* & Hasani* et al. Nature Machine Intelligence, 2020


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

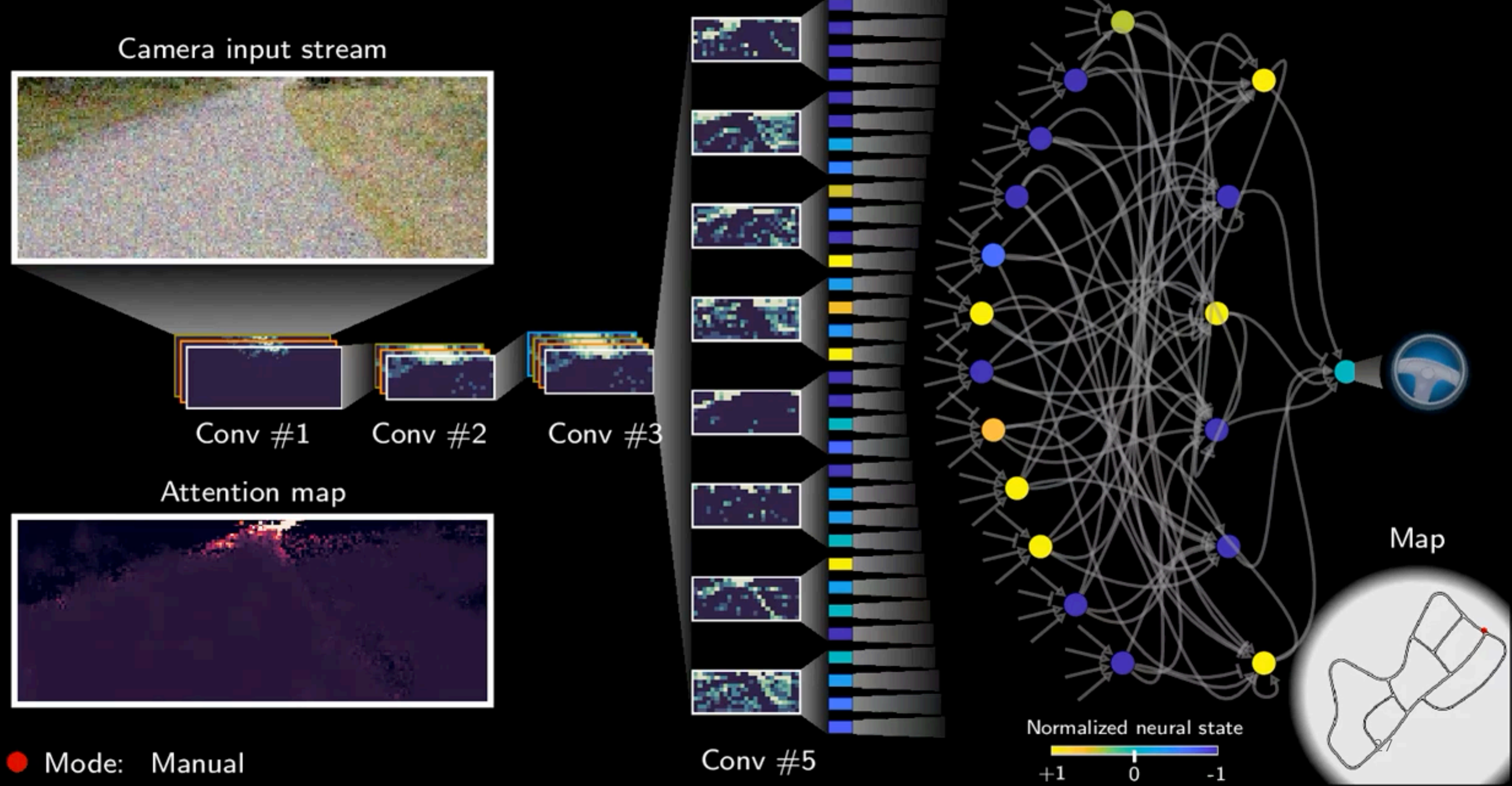


CNN driving performance
under σ^2=0.1 pertubation
Camera input stream
Conv #1
Conv #2
Conv #3
Conv #5
Attention map
Fully-connected layer #1
Fully-connected layer #2
Motor neurons
Map
Normalized neural state
+1
0
-1
Mode: Manual

NCP driving performance

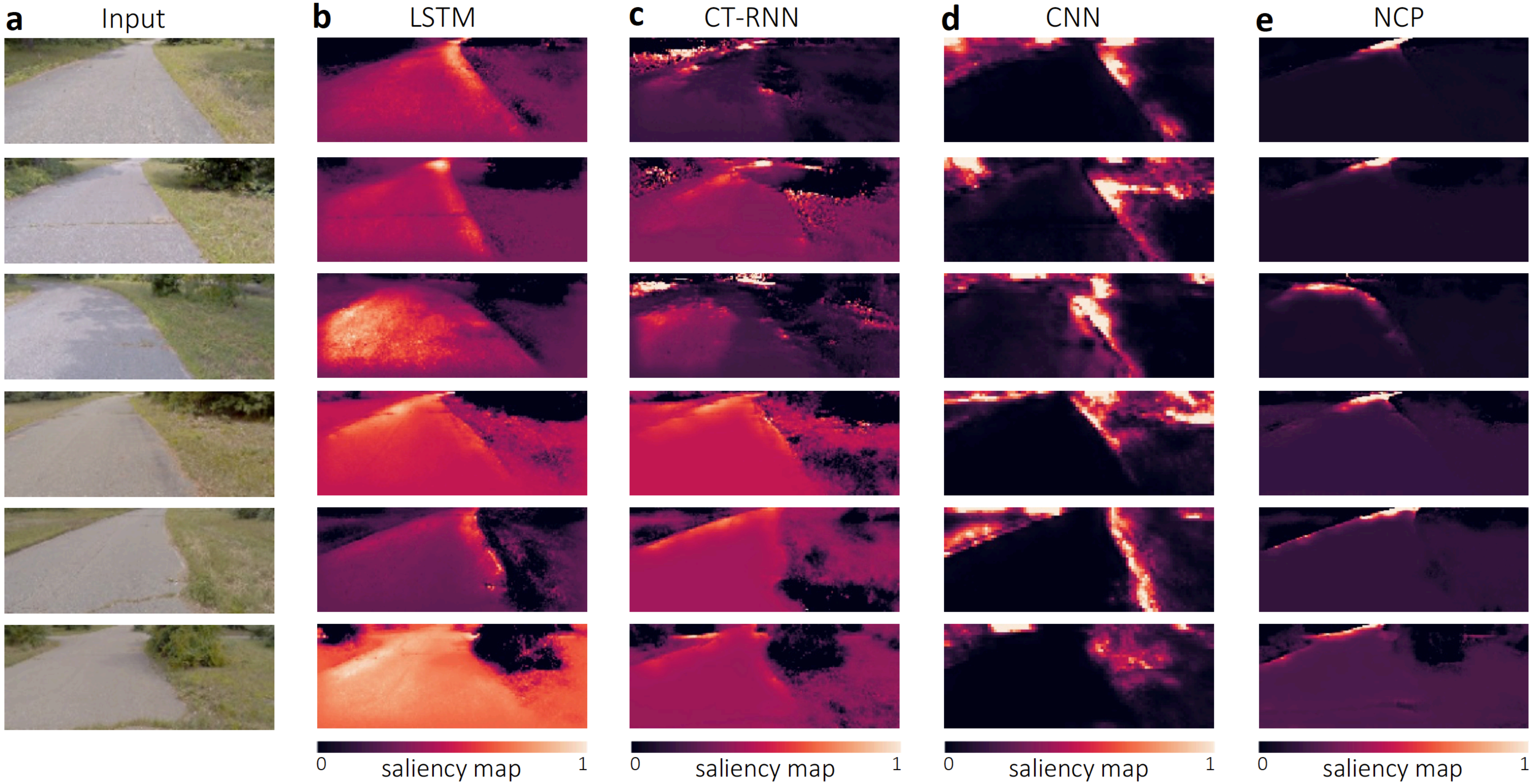


NCP driving performance
under σ^2=0.1 pertubation



LTCs: Performance

High-fidelity autonomy by LTCs



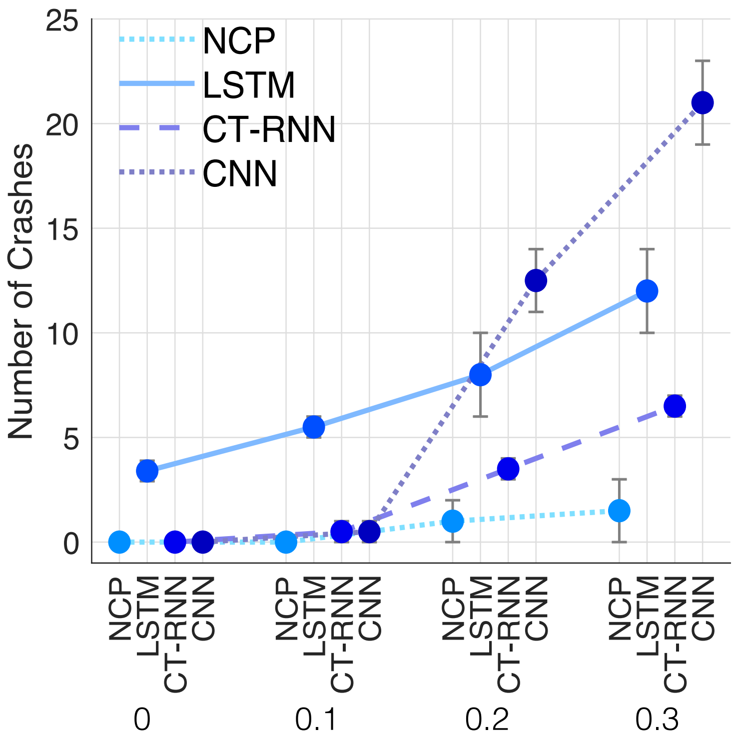
Saliency maps show where each network is learned to attend while driving

Lechner* & Hasani* et al. Nature Machine Intelligence, 2020

LTCs: Performance

High-fidelity autonomy by LTCs – Robustness

Noise Robustness (Interventions)

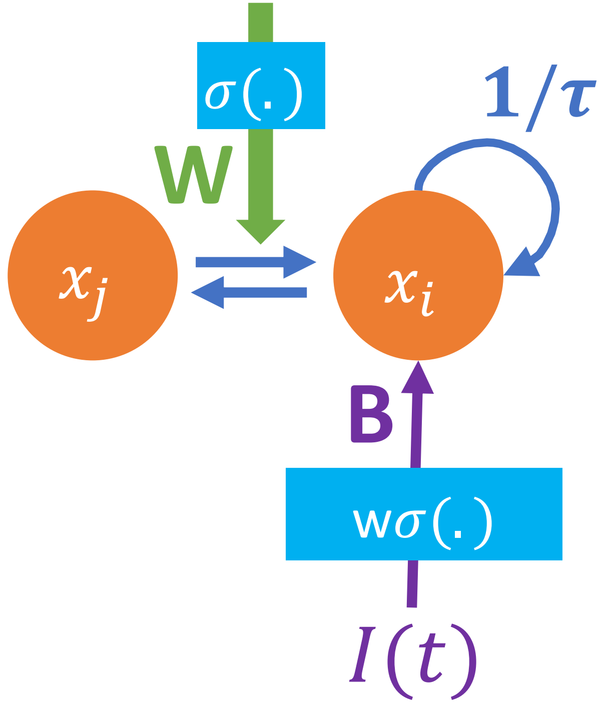


Input noise variance

Lechner* & Hasani* et al. Nature Machine Intelligence, 2020

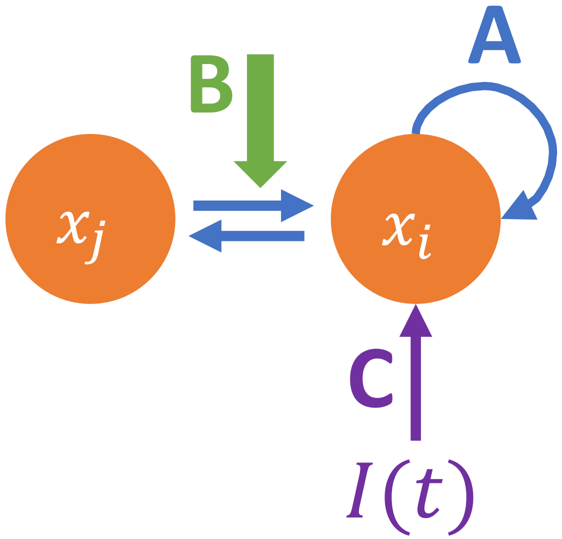
Neural Circuit Policies are Dynamic Causal Models

Liquid time-constant RNN



$^1/_\tau$ intrinsic coupling
$\mathbf{W\sigma}(.)$ liquidity modulator
B input regulator

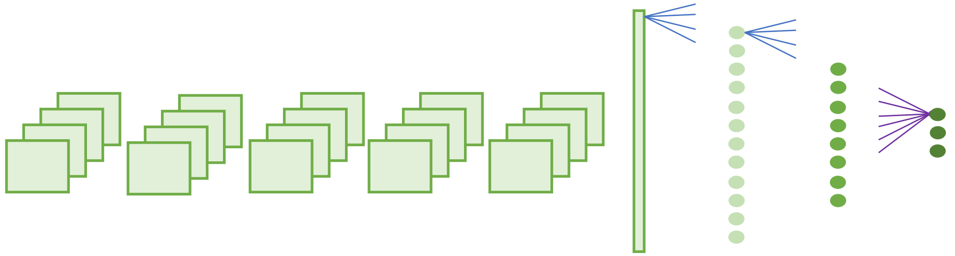
Dynamic causal model



A intrinsic coupling
B dynamic modulator
C input regulator

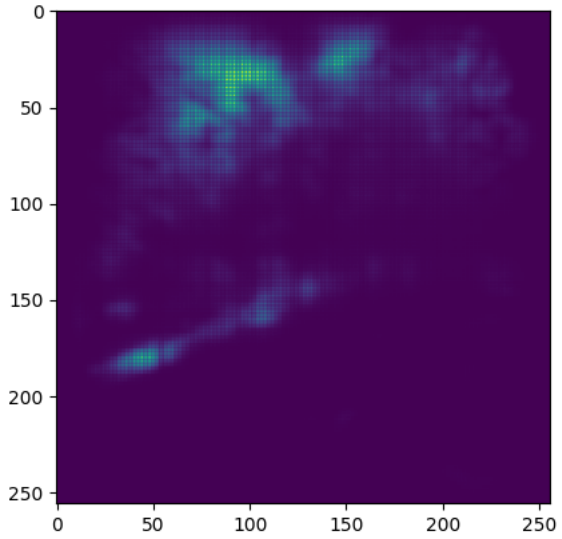
Neural Circuit Policies

Performance – Attention



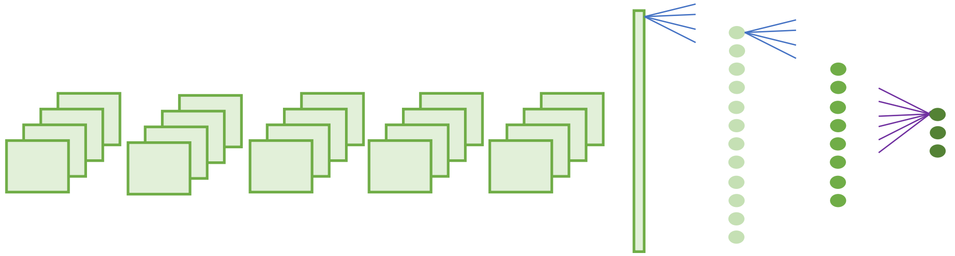
Flying Performance

Visual Backprop Attention Map



Neural Circuit Policies

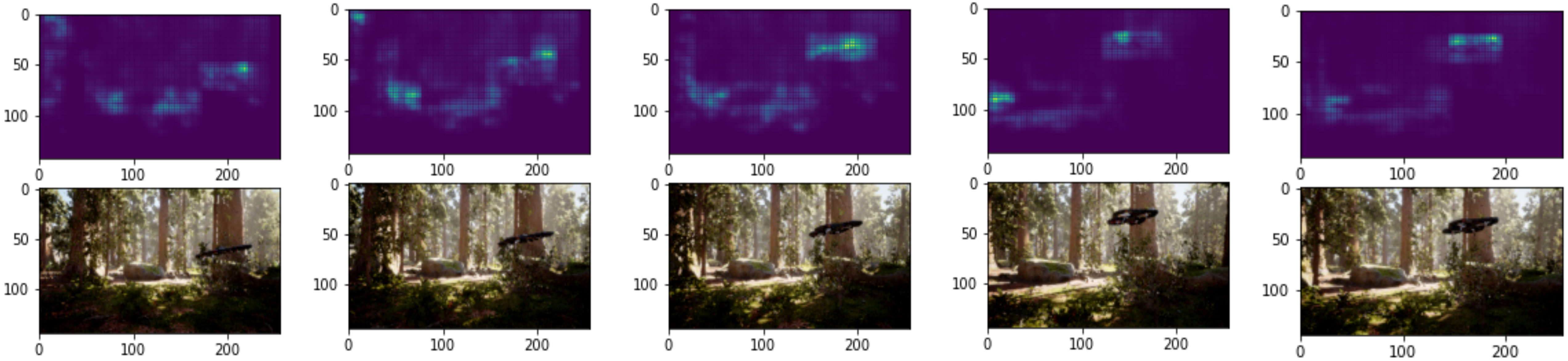
Leader Following task



Vorbach* & Hasani* et al. 2021, Under Revision

Neural Circuit Policies

Leader Following task



Vorbach* & Hasani* et al. 2021, Under Revision

Thank you!


Mathias Lechner


Alexander Amini


Daniela Rus


Radu Grosu

Feel free to

Check out our latest repositories

https://github.com/raminmh/liquid_time_constant_networks

https://github.com/mlech261/keras-ncp

https://github.com/mlech261/ode-lstms

and to reach out:
rhasani@mit.edu