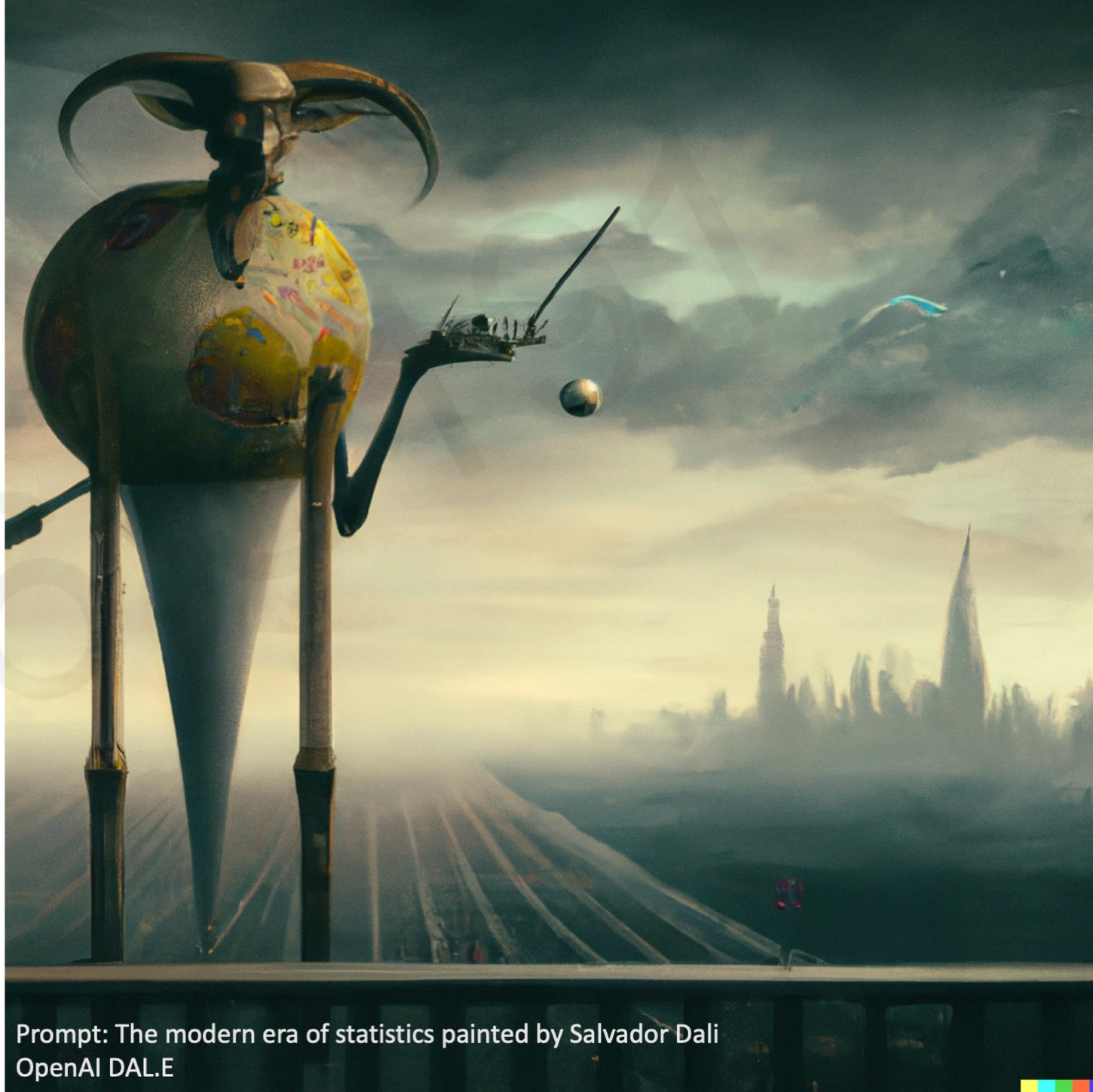
Vanguard®

Modern Era of Statistics

Ramin Hasani, Ph.D.

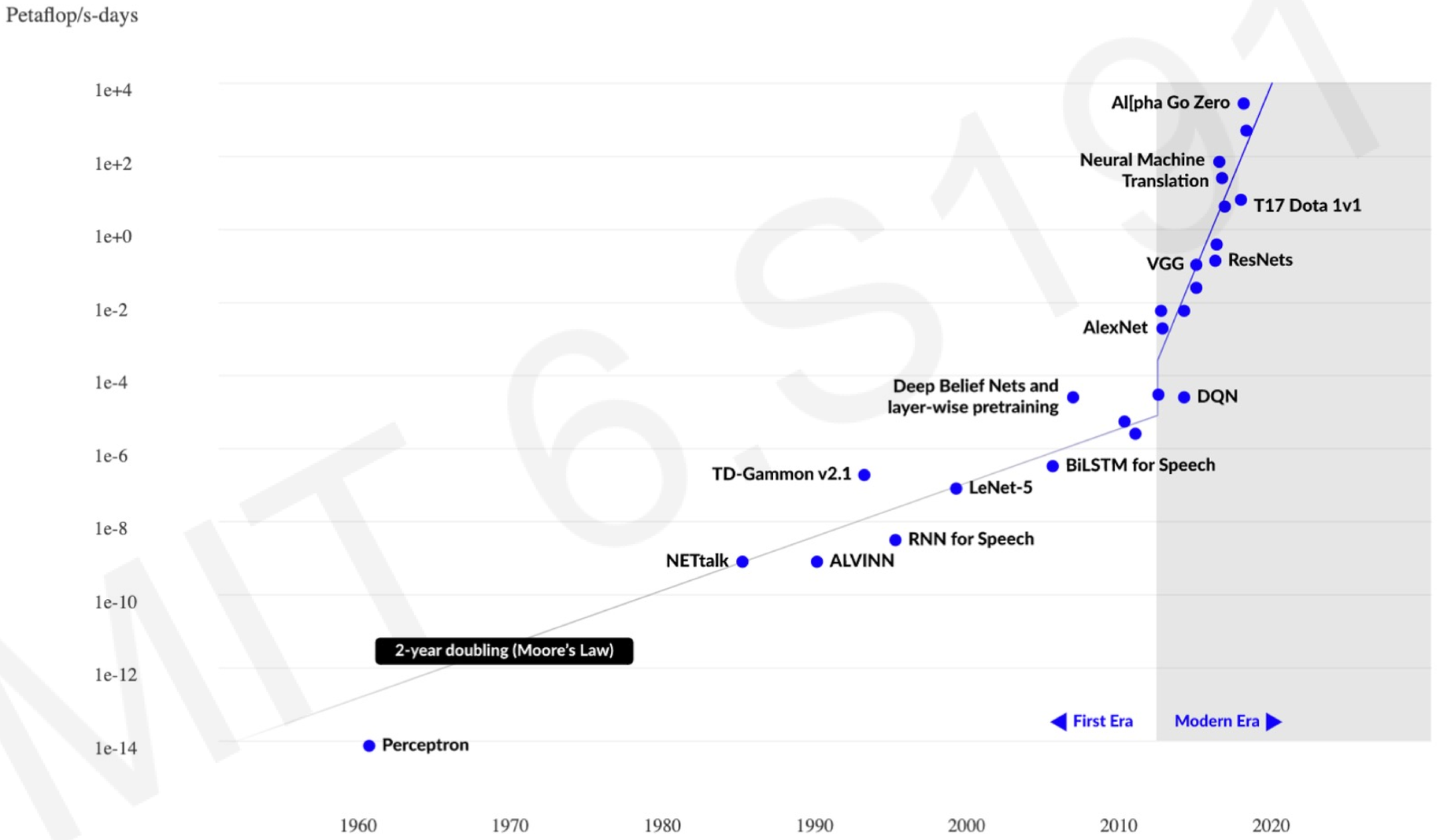
Principal AI Scientist, Vanguard
Research Affiliate, MIT

MIT Introduction to Deep Learning
January 12th 2023



Prompt: The modern era of statistics painted by Salvador Dali
OpenAI DAL.E

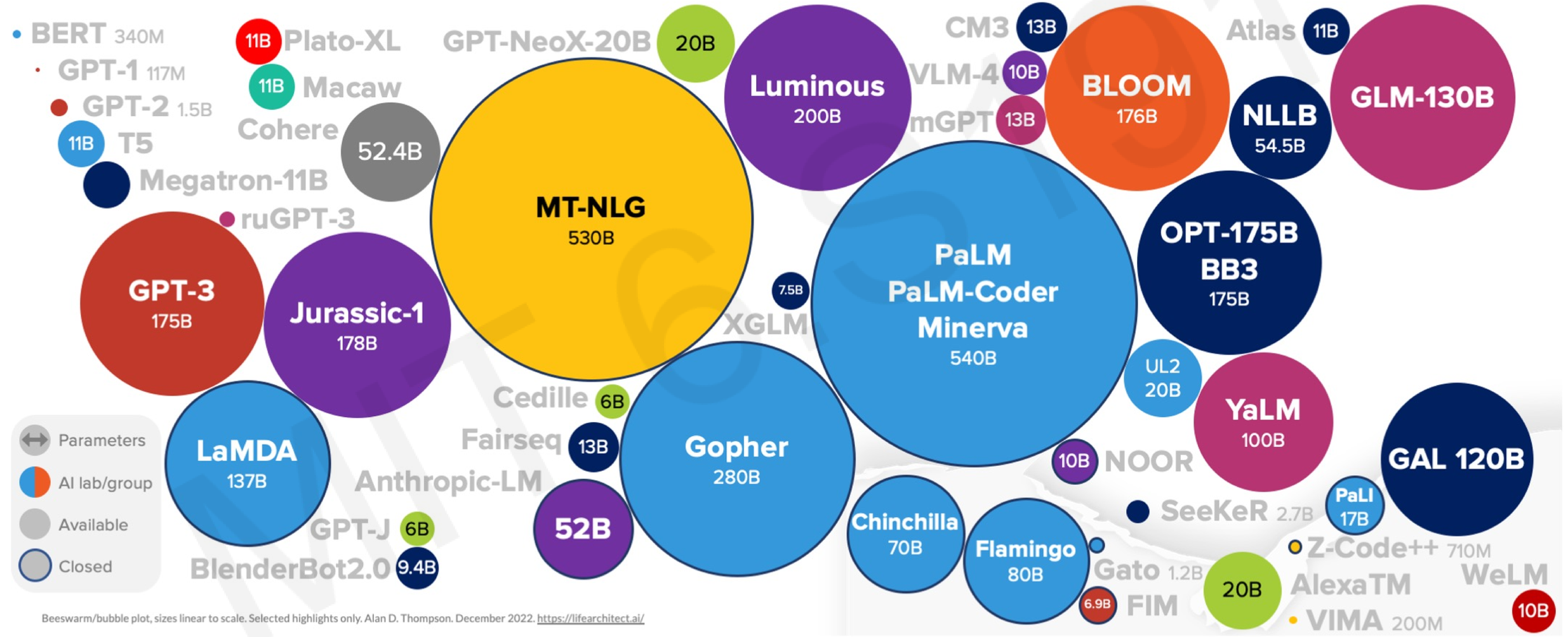
Modern Era of Statistics



Credit: https://morioh.com/p/9c545ba9416b

Modern Era of Statistics

Language Models size – up to Dec, 2022

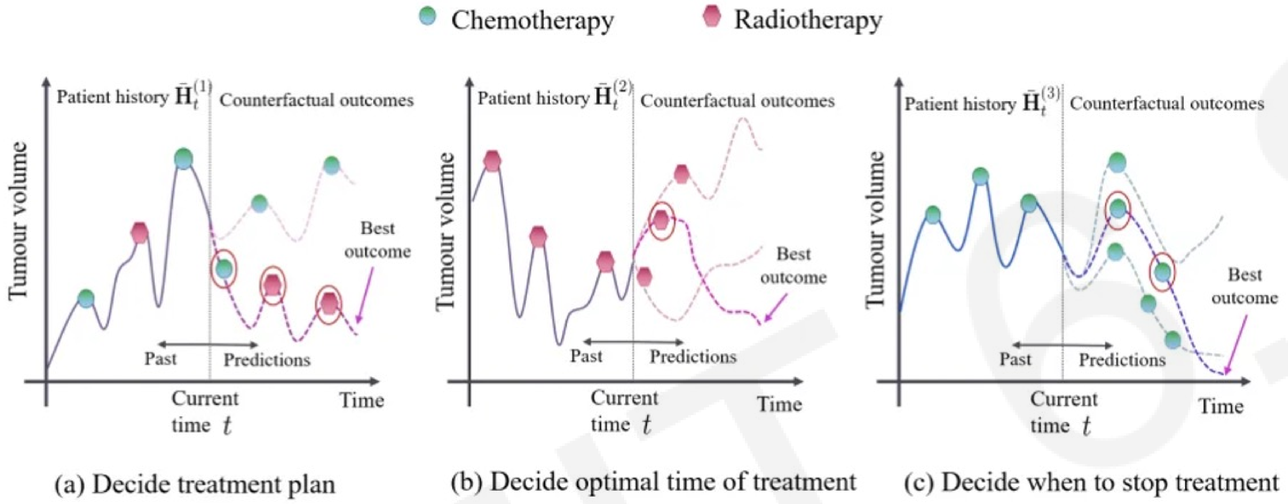


Credit: LifeArchitect.ai/models

Modern Era of Statistics

Time Series

Medical Diagnoses



(a) Decide treatment plan (b) Decide optimal time of treatment (c) Decide when to stop treatment

https://www.vanderschaar-lab.com/individualized-treatment-effect-inference/

Financial Time Series

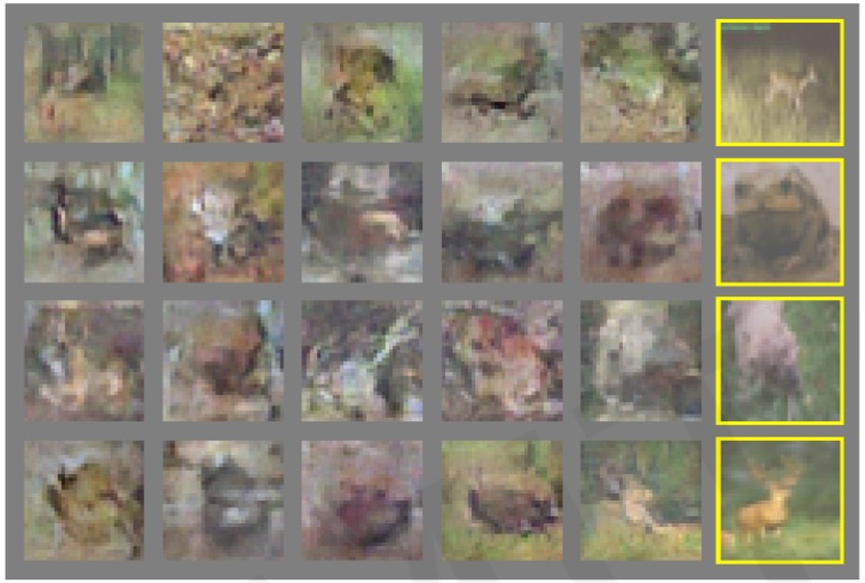
Modeling time series of 90k time steps long,
with Liquid Structural State-Space Models (Liquid-S4)

`https://github.com/raminmh/liquid-s4`

Modern Era of Statistics

Generative modeling

Generative Adversarial Networks



Goodfellow et al. 2014

Stable Diffusion

Credit: https://www.jousefmurad.com/ai/a-primer-on-stable-diffusion/

Modern Era of Statistics

Bigger seems to be better? But why?

Solving n equations requires n unknowns

$$2x + 3y = 20$$
$$4x - 2y = 12$$

But then deep learning:

Choose excessively more unknowns to learn from n data (equations)!

Scale in Modern Machine Learning

MNIST n = 60k points, d = 28 x 28 images

Today models with millions of parameters are trained on MNIST
The performance improves with increasing the number of parameters!

How does this make sense? What are we learning?

Generalization bound $\alpha \sqrt{\frac{\#\ of\ param}{dataset\ size}}$

ImageNet: is 1.4M images of size 256 x 256 x 3 and models can be hundreds of millions of parameters.

NLP: datapoints of few billions, and models are hundreds of billions!

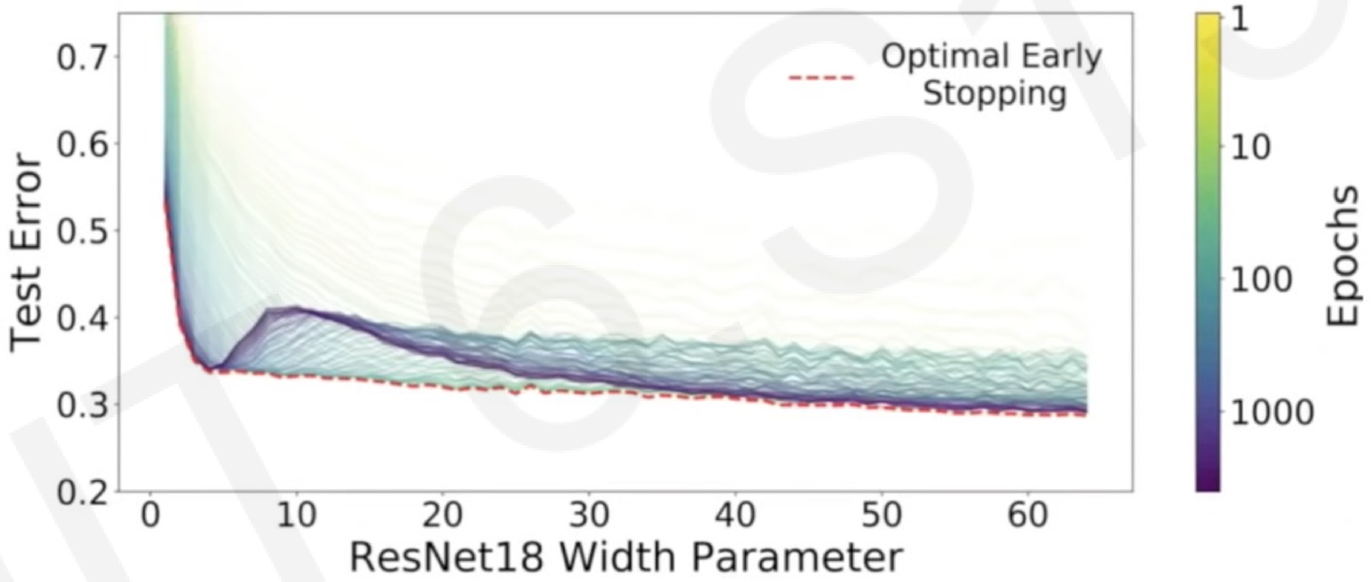
Bubeck and Sellke NeurIPS 2021 https://youtu.be/OzGguadEHOU

Prompt: A portrait photo of a kangaroo wearing an orange hoodie and blue sunglasses standing on the grass in front of the Sydney Opera House holding a sign on the chest that says Welcome Friends!

Credit: parti.research.google

Benign Overfitting & Double Descent

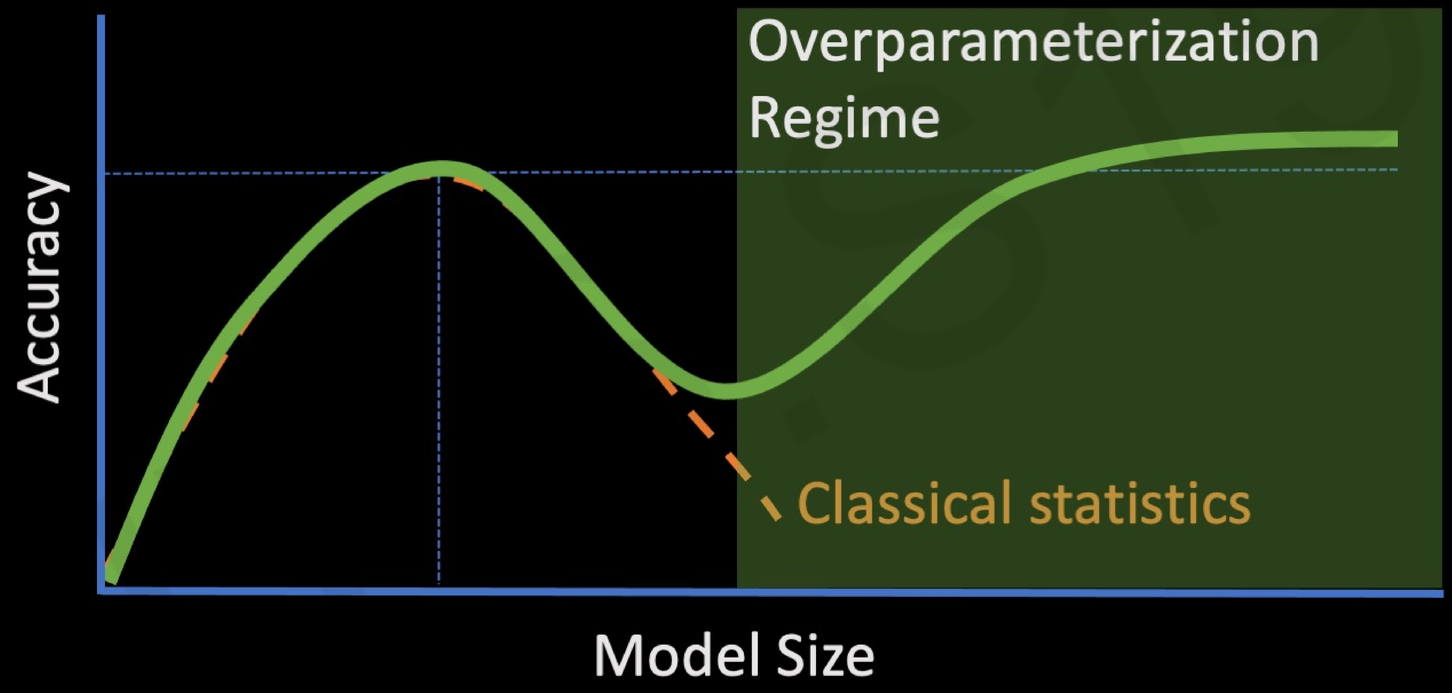
Experiment from [Nakkiran et al., 2019] on CIFAR-10 with 15% label noise:



Scaling does help generalization a bit!

Frei et al. 2022. Benign Overfitting without Linearity

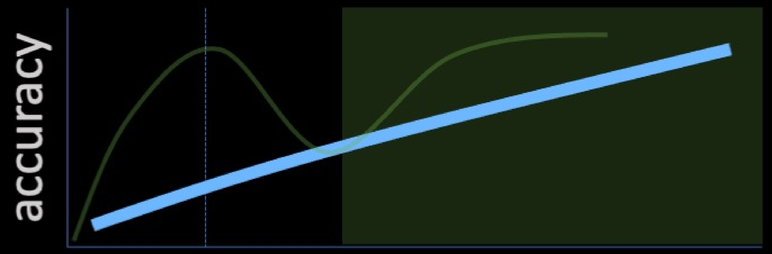
Modern Era of Statistics



Belkin et al. 2020 & Nakkiran et al. 2021

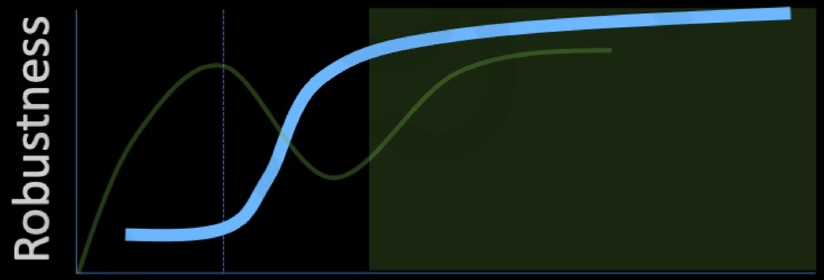
Modern Era of Statistics

Generalization across many tasks and domains



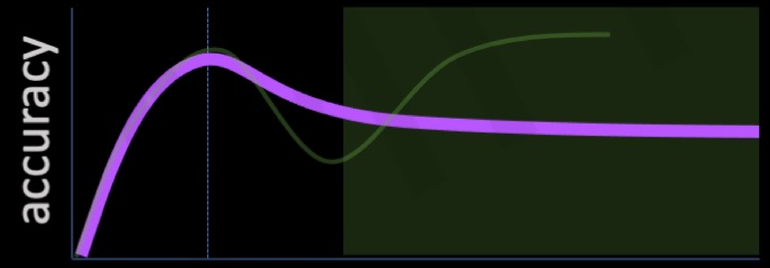
[Reed et al. DeepMind 2022]

Scale improves Robustness



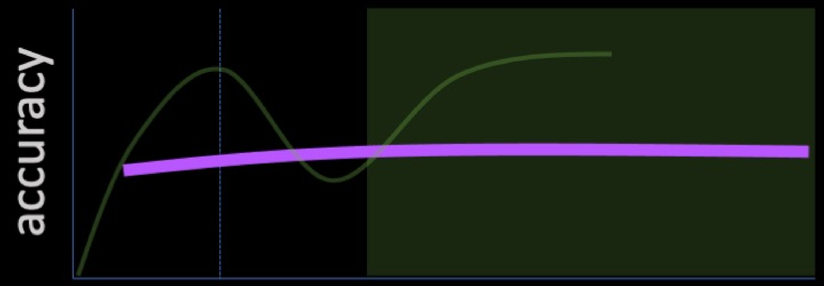
[Madry et al. ICLR 2018, Bubeck & Selke NeurIPS 2021]

Worsen accuracy on minority samples



[Sagawa et al. ICML 2020]

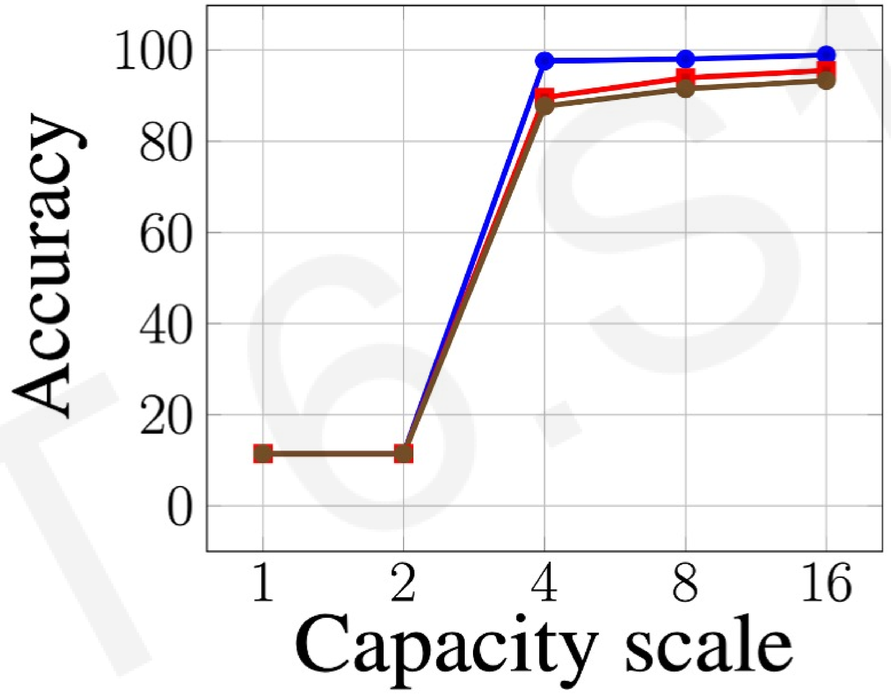
Reasoning stays unchanged



[Liu et al. Google Research 2022]

Scale Improves Robustness

Experiment from [Madry et al., 2018] on MNIST:



They use adversarial training against Projected Gradient Decent (PGD) attacks on various number of params.

Scale is a law of robustness

A Universal law of robustness [Bubeck and Sellke 2021]:
https://youtu.be/OzGguadEHOU

Fix any **"reasonable"** function class with p parameters (e.g., deep nets with poly-size parameters and NOT Kolmogorov-Arnold type networks).

Sample n data points from a **"truly high dimensional"** distribution (e.g., a mixture of Gaussians, or ImageNet with a properly defined notion of "dimension"). **Add label noise.**

Then, to **memorize** this dataset (i.e., optimize the training error below the label noise level), and to do so **robustly** (in the sense of being Lipschitz), one must necessarily have **dramatic overparameterization**:

$$p \gtrsim n\, d$$

Kolmogorov-Arnold Representation Theorem

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right)$$

the non-smoothness of the inner functions and their "wild behavior" has limited the practical use of the representation [Girosi & Poggio 1989]

Lipschitzness: If I move my inputs by ϵ then I would ideally want the output also moves by ϵ

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2)$$

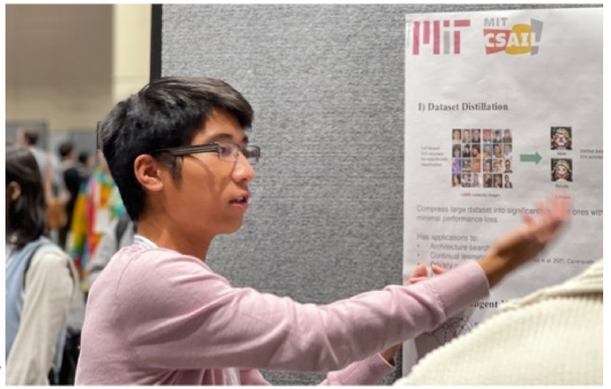
Why is $p \gg nd$ called "dramatic Overparameterization" ?

Intuitively, memorizing n data points is about satisfying n equations, so order n parameters should be enough.

Theorem (Baum 1988)
Two layer neural net with threshold activation function only need $\left(p \approx O(n)\right)$ to memorize binary labels.

[Yun, Sra, Jadbabaie 2019; Bubeck, Eldan, Lee, Mikulincer 2020]
In fact the same is true with ReLU on real labels.

[Bubeck, Eldan, Lee, Mikulincer 2020] In fact even **Neural Tangent Kernels** can do it.



Noel Loo

Bubeck and Sellke NeurIPS 2021 https://youtu.be/OzGguadEHOU

Examples in real-world data:

MNIST. It has around $n \approx 10^5$ and $d \approx 10^3$. [Madry et al., 2018] show a transition in robust accuracy at around $p \approx 10^6$

Note 1: their *notion of robustness (PGD)* does not exactly match the law of robustness (Lipschitz constant).

Note 2: the law seems to be contradicted since $10^6 \ll 10^5 \times 10^3$?
No. MNIST is NOT truly "$10^3 - \text{dimensional}$". "**Effective dimension**": $d_{eff} \approx 10^1 \rightarrow p \sim n\, d_{eff}$

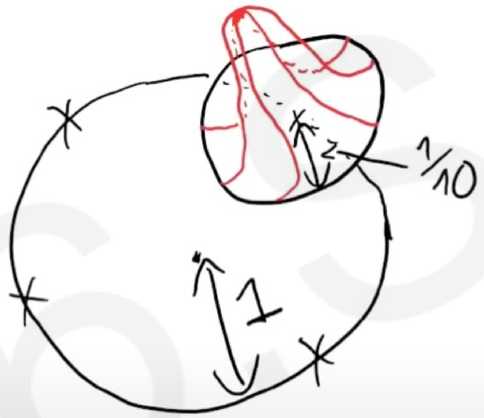
Note 3: what is "noisy labels" in real data? Measuring the "difficult" part of a learning problem. For MNIST it should be 2-5% gain in accuracy.

What about ImageNet? It's $n \approx 10^7$ and $d \approx 10^5$ ($d_{eff} \approx 10^3$?), hence we predict that at least 10^{10} parameters are needed. **Current models are too small?!??? (Less than 10^9 parameters)**

Bubeck and Sellke NeurIPS 2021 https://youtu.be/OzGguadEHOU

What about Smoothness?

All these constructions are $\Omega(\sqrt{d})$ Lipschitz even for well-dispersed data (e.g., i.i.d. on the sphere), but in principle one can memorize such data with $O(1)$ – Lipschitz functions (We assume n = poly(d)):

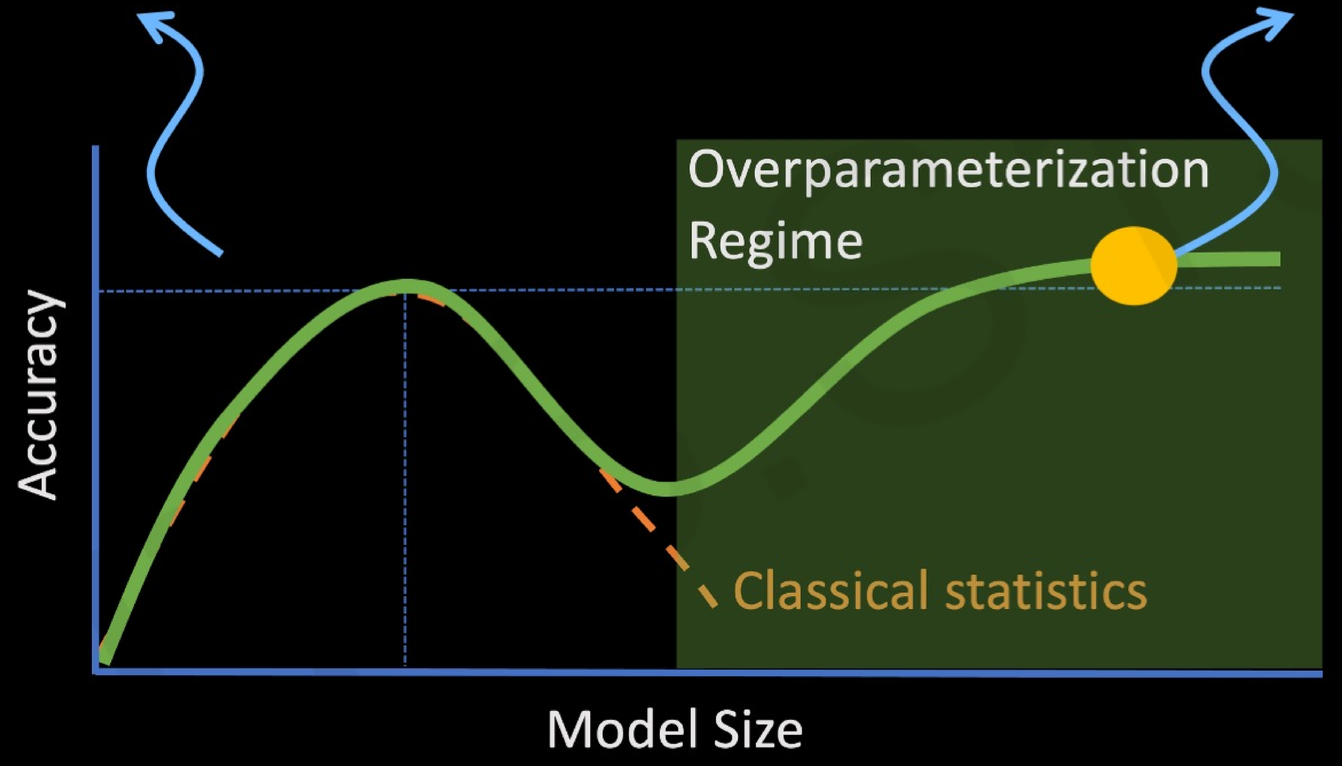


Picture can easily be realized with k = O(n) neurons (p = nd). So we have two options: either small model (p = n) but nonrobust (Lip = $\sqrt{d}$), or very large (p= nd) and very robust (Lip = 1).

Is this tradeoff real? Can we do better than $Lip \leq O(\frac{\sqrt{nd}}{p})$?

The law of robustness says this is tight!

Great Generalization
More Robust
Reasoning
Bias and Fairness
Energy
Accountability

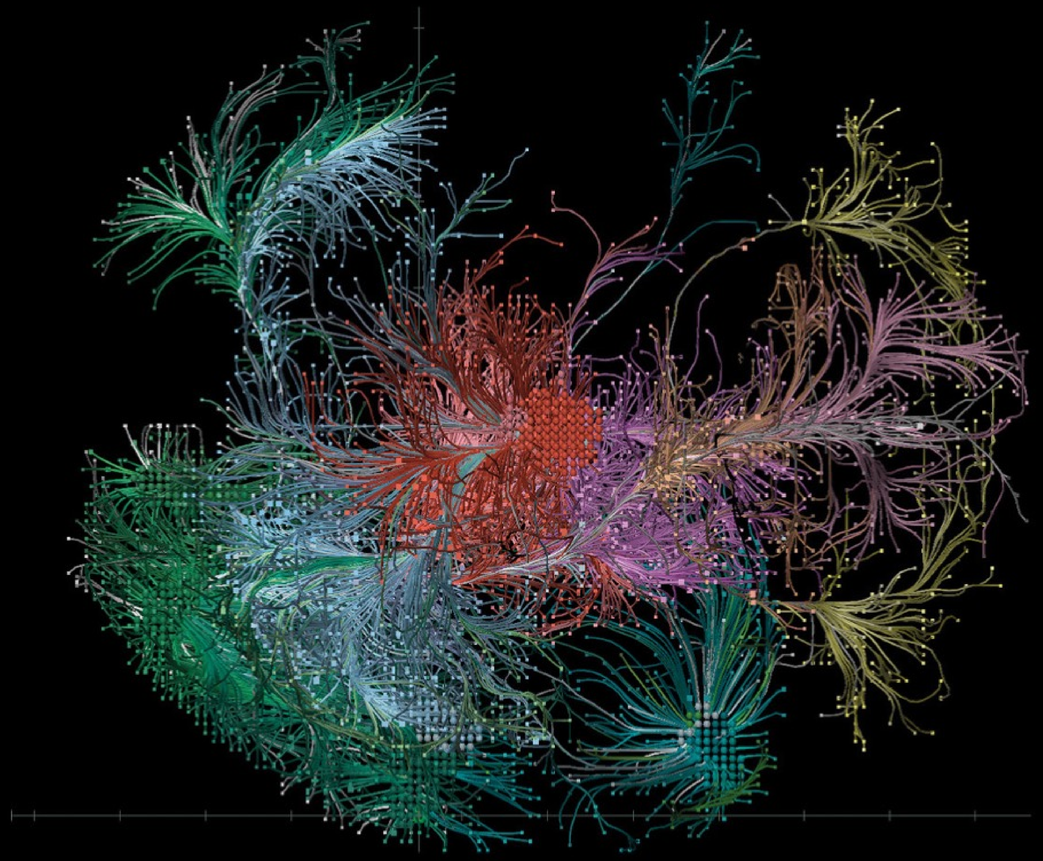


Great Generalization
More Robust
Reasoning?
Bias and Fairness?
Energy?
Accountability?

HOW?

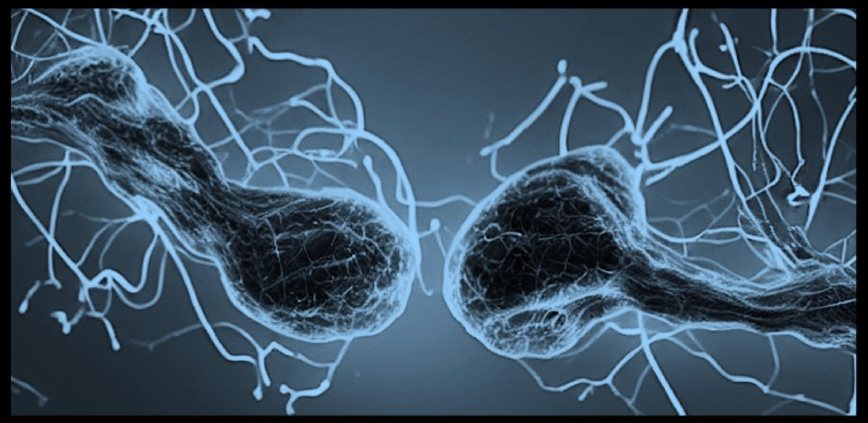
Neuroscience inspiration as inductive bias

Brain



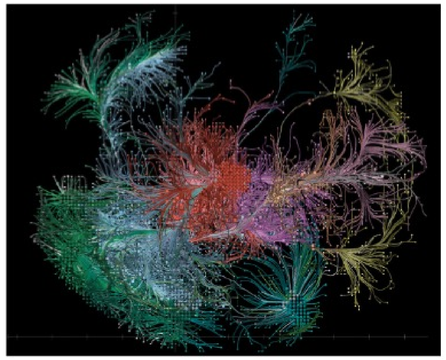
© Image: Allen Institute for Brain Science

Liquid Neural Networks

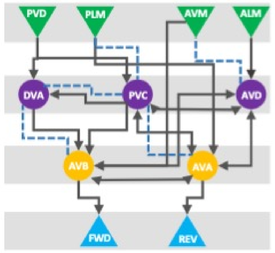


Lechner et al. Nature Machine Intelligence 2020
Hasani et al. Nature Machine Intelligence 2022

Nervous Systems
(Image: Allen Institute for Brain Science)



Neural Circuits



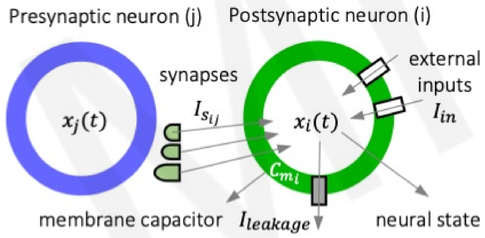
ICML 2020
ICRA 2019
NeurIPS Deep RL Symp 2017

Liquid Networks

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

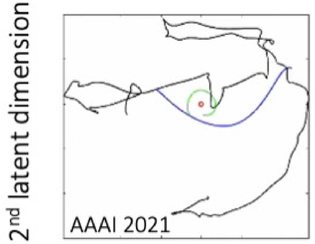
$$\mathbf{S}(t) = f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)(A - \mathbf{x}(t))$$

Neurons & Synapses

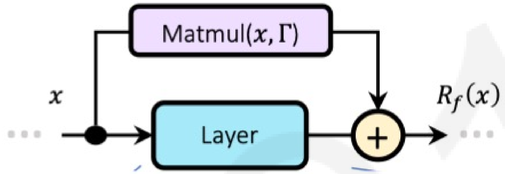


Nature Machine Intelligence 2020

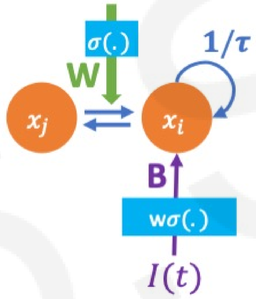
Expressivity



memory

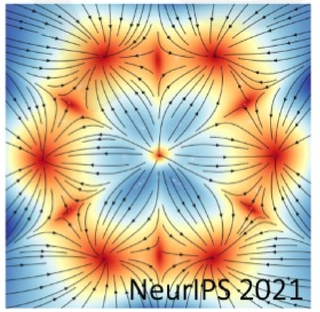


Causality

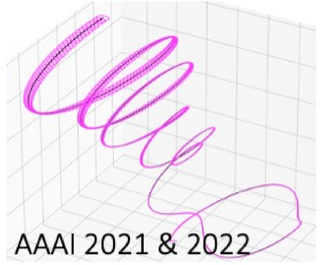


NeurIPS 2021

Generative Modeling

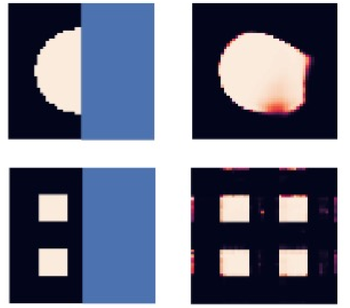


Robustness

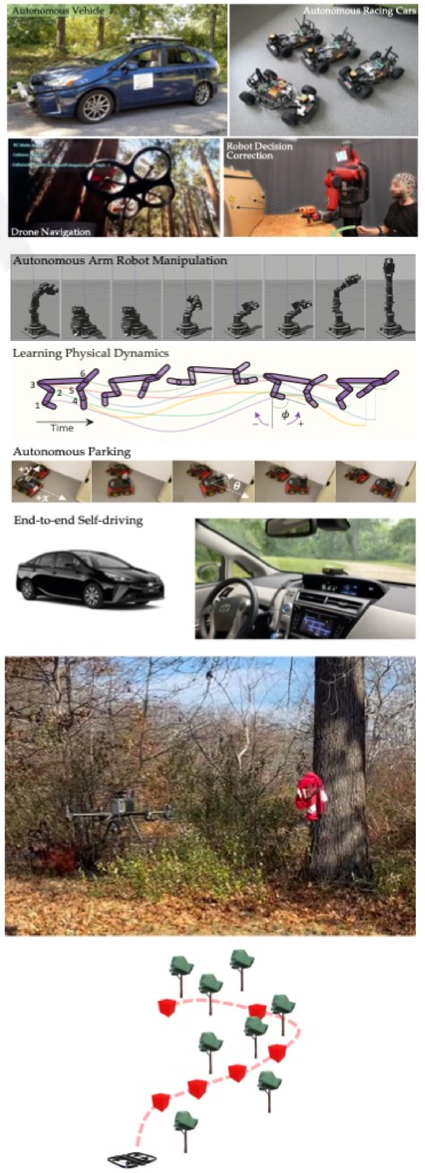


AAAI 2021 & 2022

Extrapolation

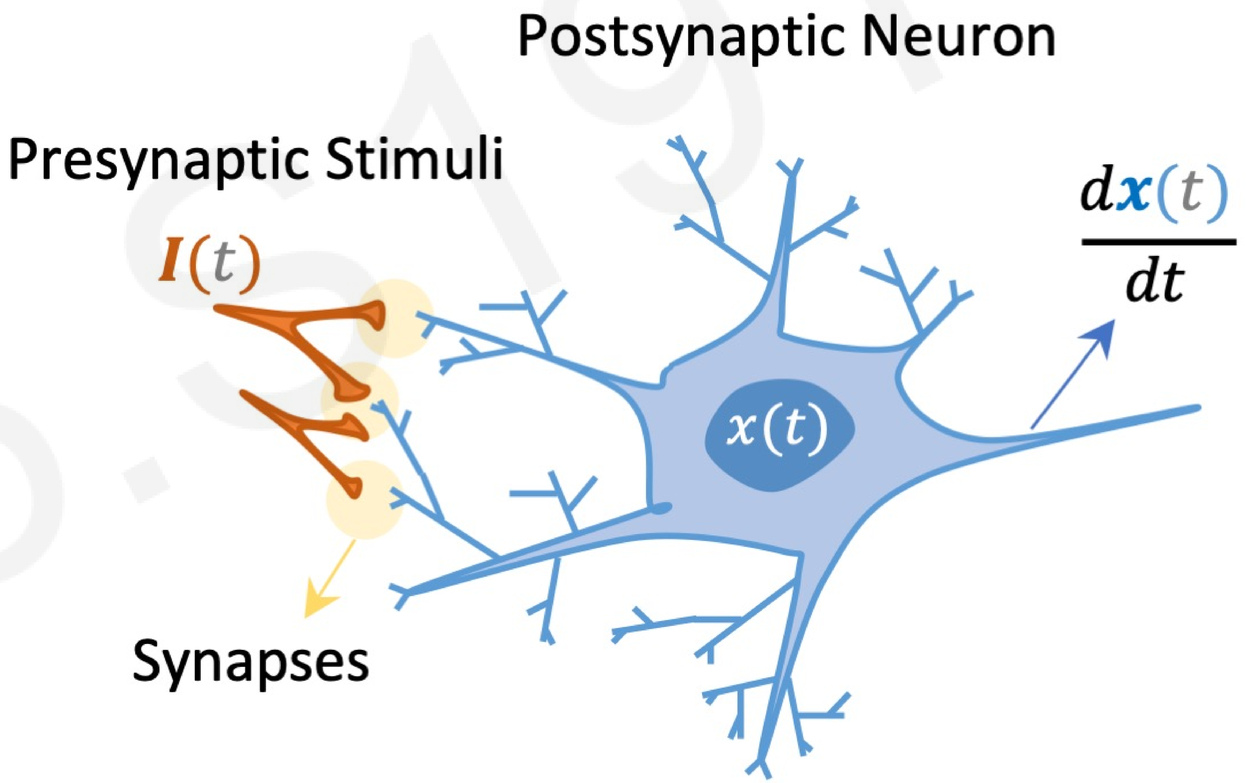


Mixed-horizon Decision-making



What are the building blocks' differences?

- ✓ Neural dynamics are typically continuous processes and are described by differential equations
- ✓ Synaptic release is much more than scalar weights
- ✓ Recurrence, memory, and sparsity



Let’s incorporate these building block differences to:

Improve representation learning
Improve robustness and flexibility of models
Improve models’ interpretability

Explore Continuous-time (depth) models

What is a continuous- time/depth neural network?

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

Number of layers

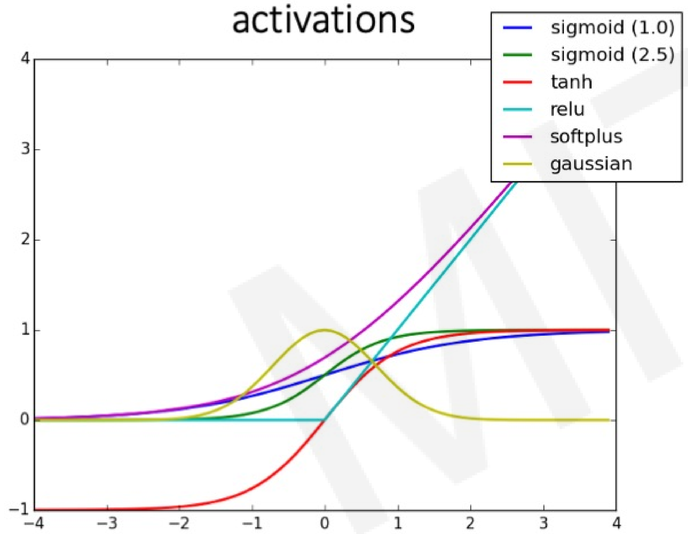
Width

Hidden state

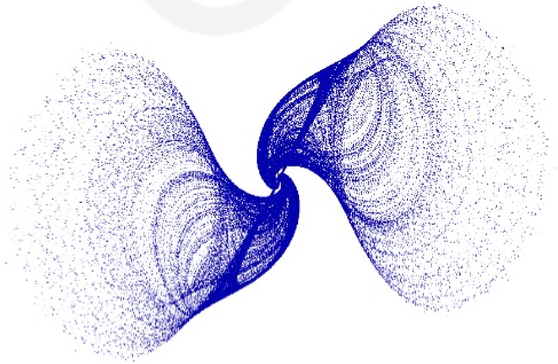
Inputs

Model parameters

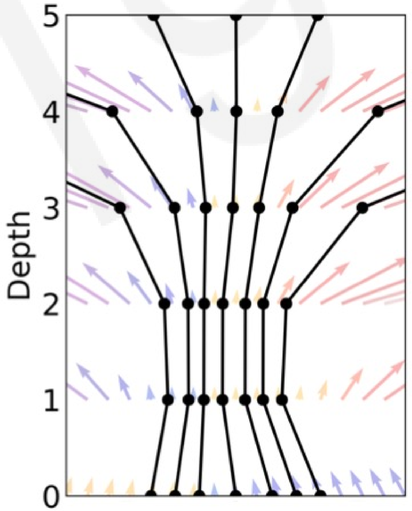
activations



Dynamical systems



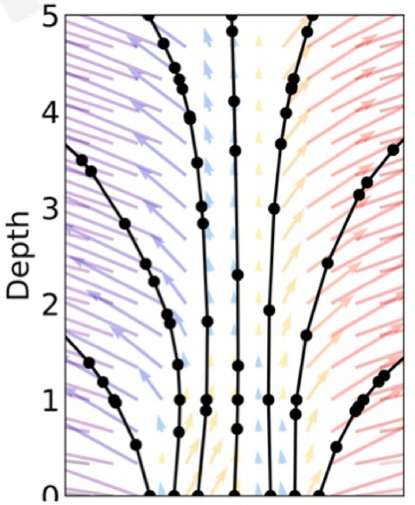
Residual Network



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

He et al.
CVPR 2016

ODE Network



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

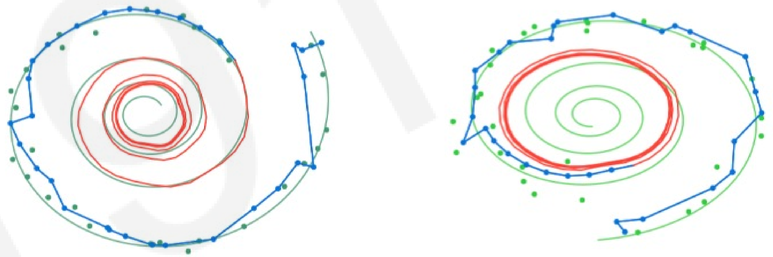
Chen et al.
NeurIPS 2018

Figure Credit: Chen et al. NeurIPS 2018

What is a continuous- time/depth neural network?

Standard Recurrent Neural Network (RNN)
Hopfield 1982

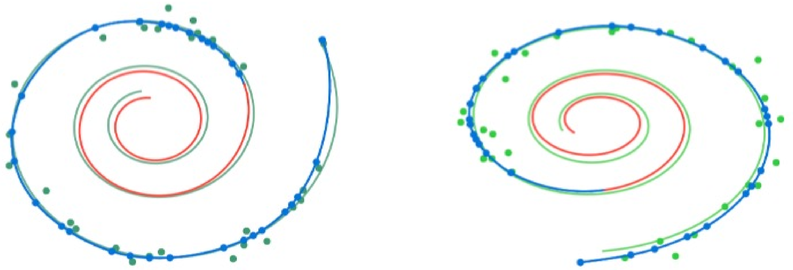
$$x(t+1) = f(x(t), I(t), t\,; \theta)$$



(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

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

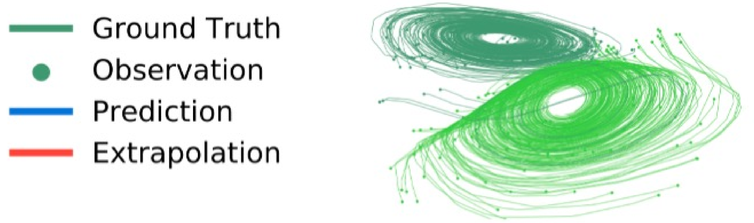


Figure Credit: Chen et al. NeurIPS 2018

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

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

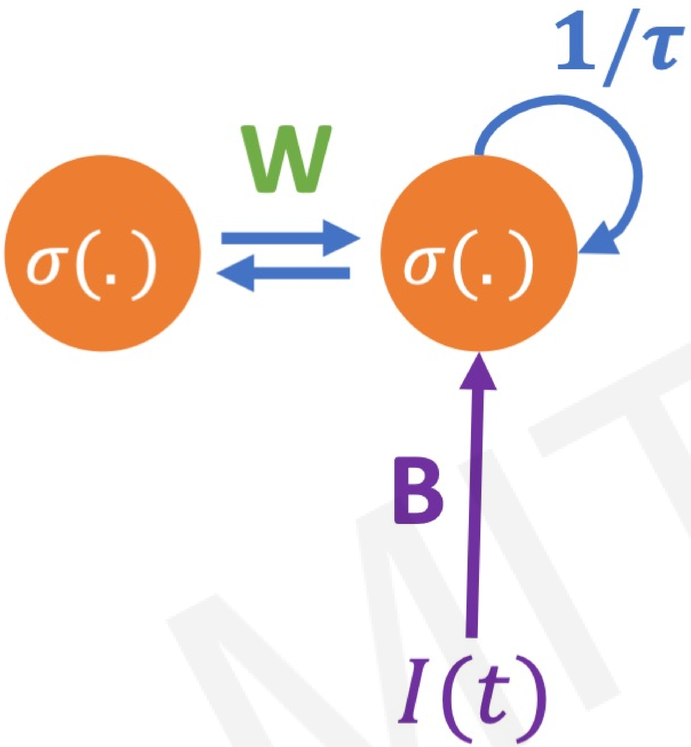
$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A$$

"Liquid" = variable

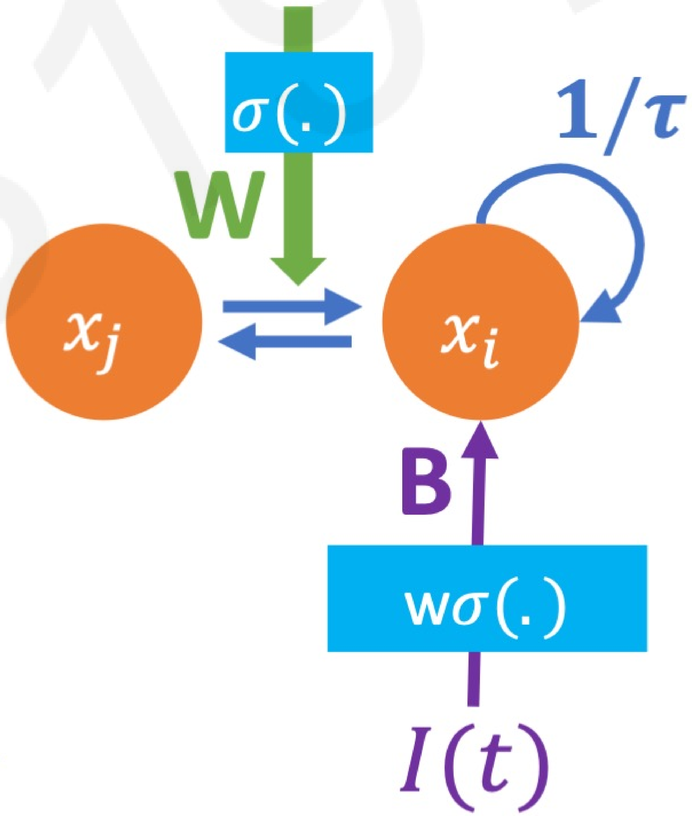
Hasani, Lechner, Amini, Grosu, Rus et al. AAAI 2021

Liquid Time-Constant Networks

Standard Neural Nets



Liquid Neural Nets



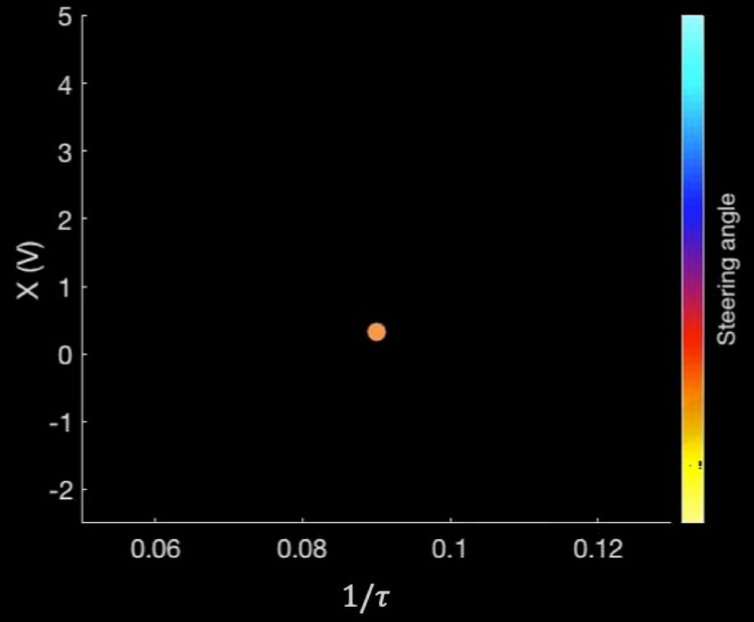
$1/\tau$ intrinsic coupling

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

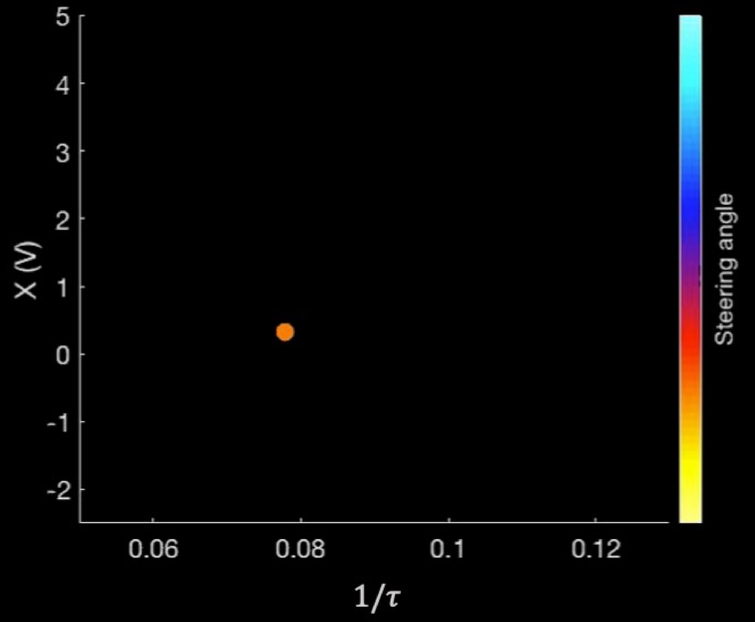
B input regulator

Hasani, Lechner, Amini, Grosu, Rus et al. AAAI 2021

Standard Neural Network



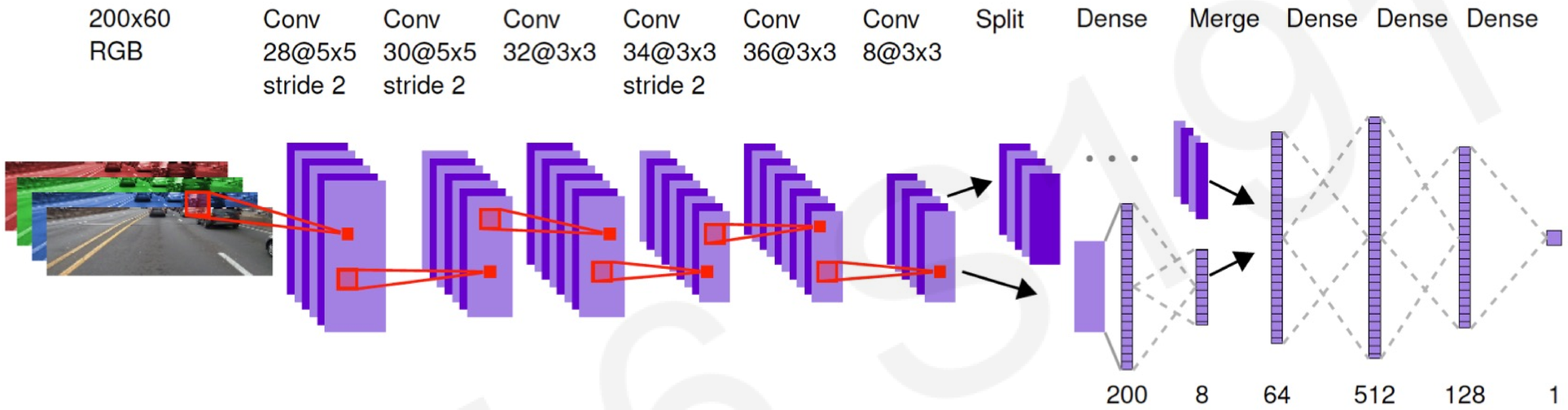
Liquid Neural Network



LTCs: Performance

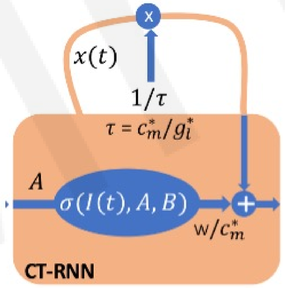
https://github.com/mlech26l/ncps

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

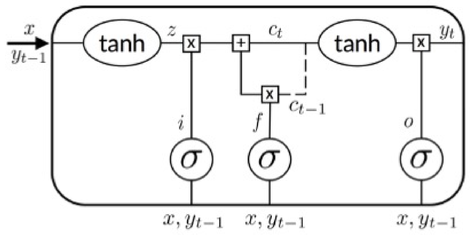


What if we replace the fully connected layers by a recurrent neural network?

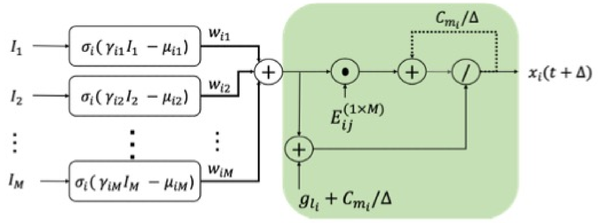
ODE-RNN



LSTMs



LTC-based Networks?



Lechner, Hasani, Amini, Henzinger, Grosu, Rus, Nature Machine Intelligence, 2020

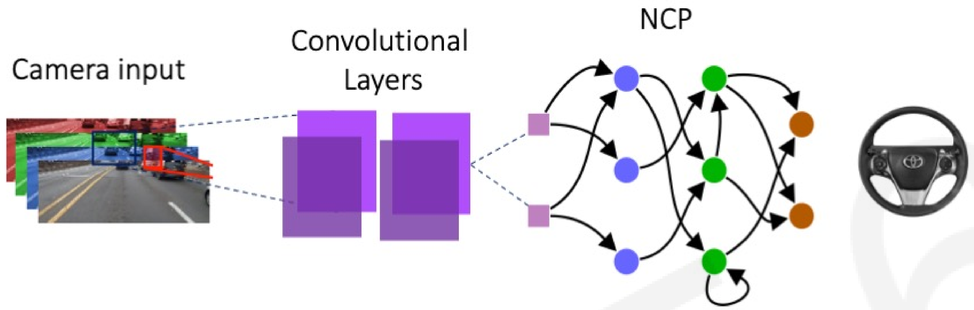
LTCs: Performance

https://github.com/mlech26l/ncps

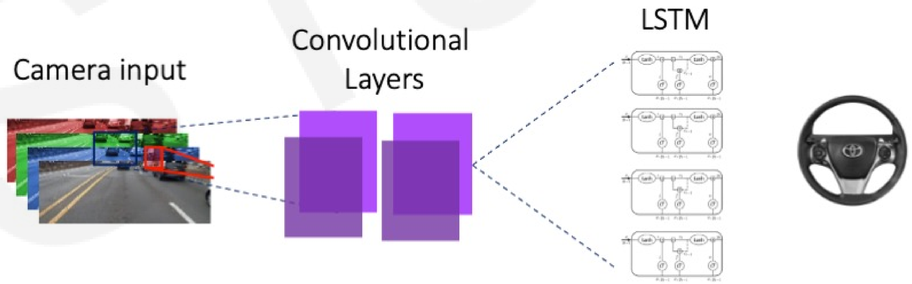
High-fidelity autonomy by LTCs
end-to-end learning of Neural Circuit Policies (NCP)

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

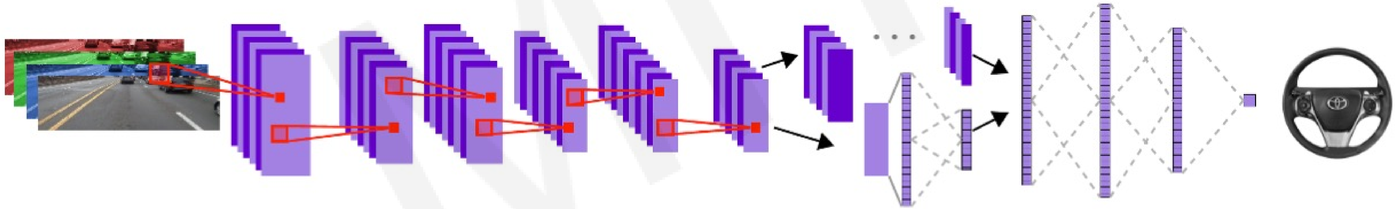
NCPs



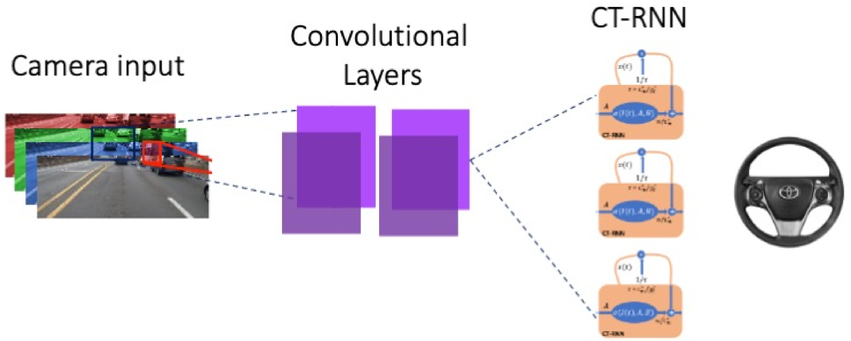
LSTMs



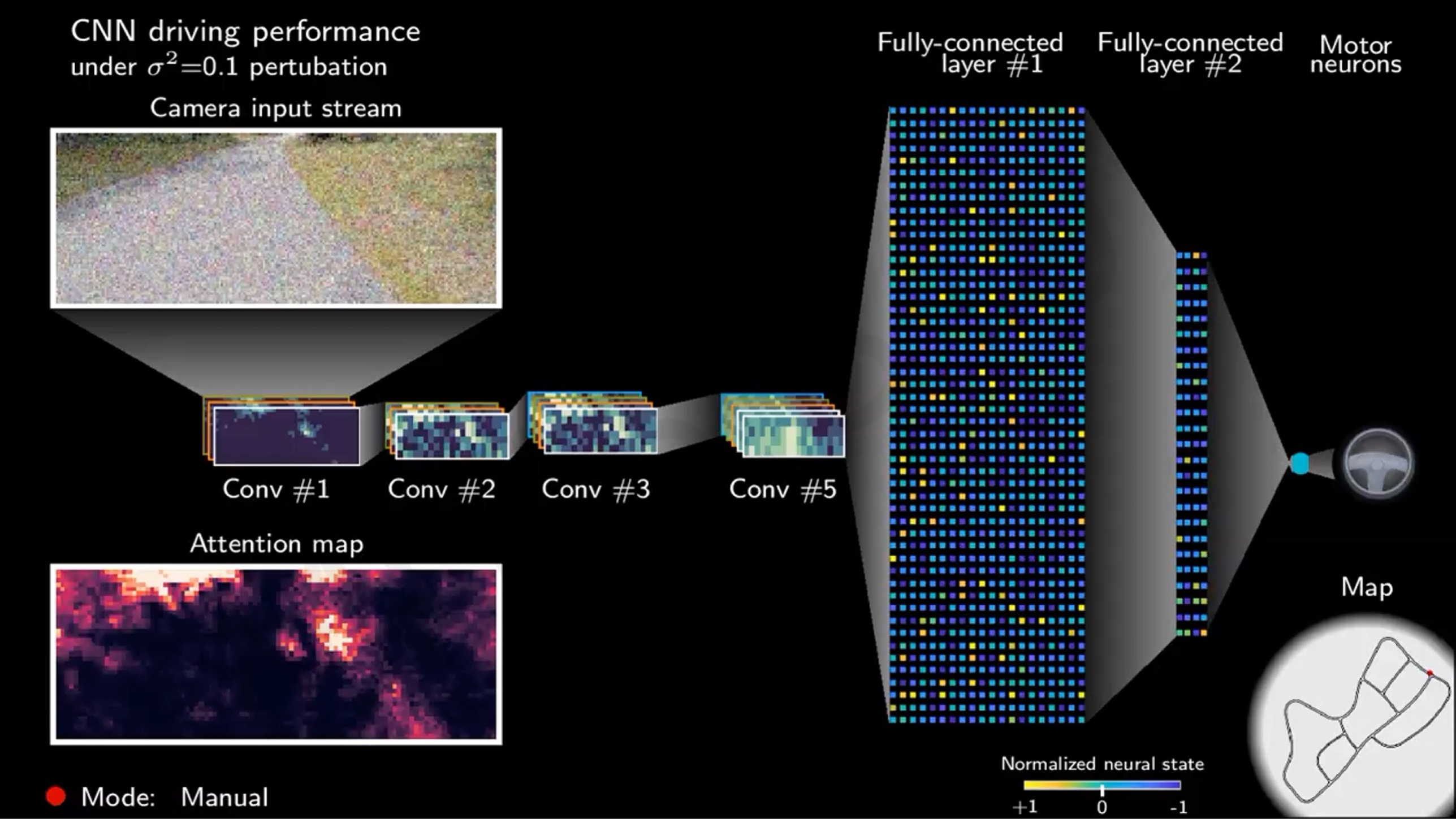
Convolutional Neural Networks (CNNs)

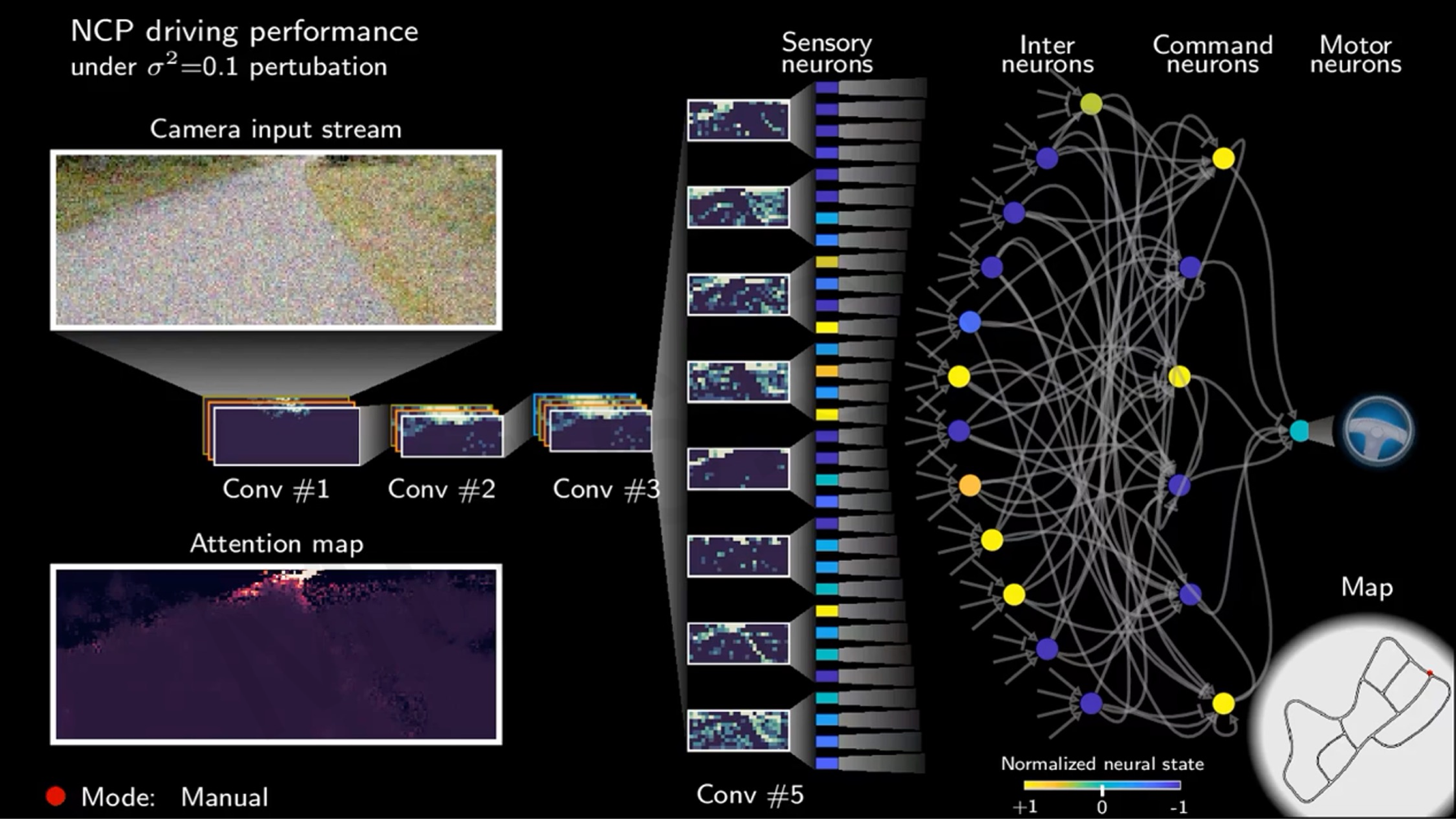


CT-RNNs



Lechner, Hasani, Amini, Henzinger, Grosu, Rus, Nature Machine Intelligence, 2020


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

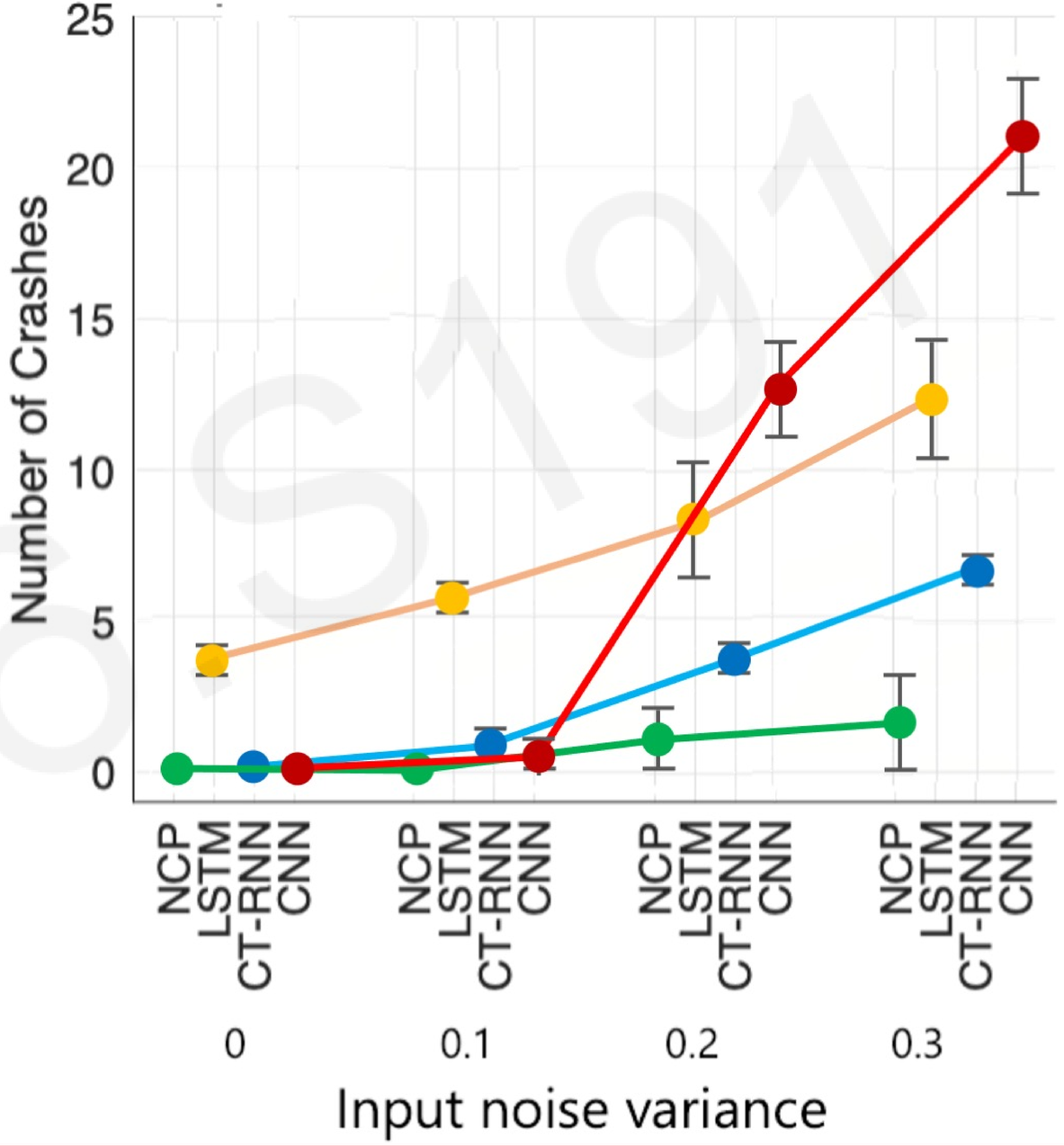

NCP driving performance
under σ^2=0.1 pertubation
Camera input stream
Conv #1
Conv #2
Conv #3
Attention map
Sensory neurons
Conv #5
Inter neurons
Command neurons
Motor neurons
Normalized neural state
+1
0
-1
Map
Mode: Manual

LTCs: Performance

Robustness to perturbations

CNN
LSTM
CT-RNN
LTC (ours)

Liquid time-constant neuron resilience to sensory noise

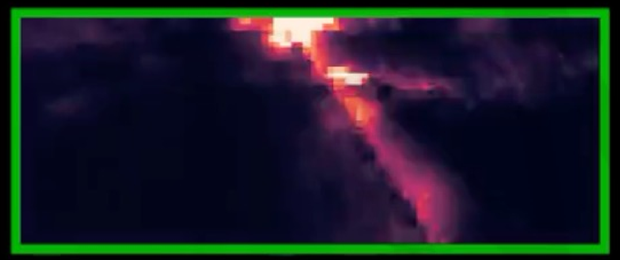


Lechner, Hasani, Amini, Henzinger, Grosu, Rus, Nature Machine Intelligence, 2020

Camera input stream

Map

CNN

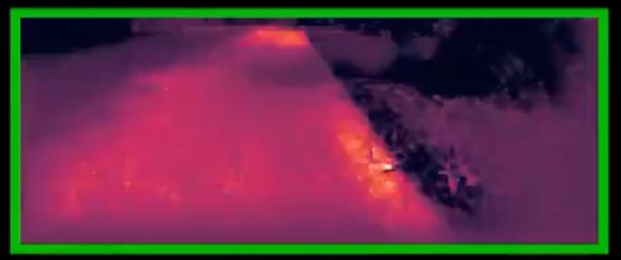


Mode: Autonomous

CT-RNN

Mode: Autonomous

LSTM



Mode: Autonomous

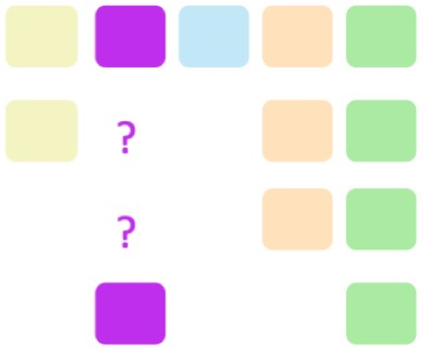
Our solution

Mode: Autonomous

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

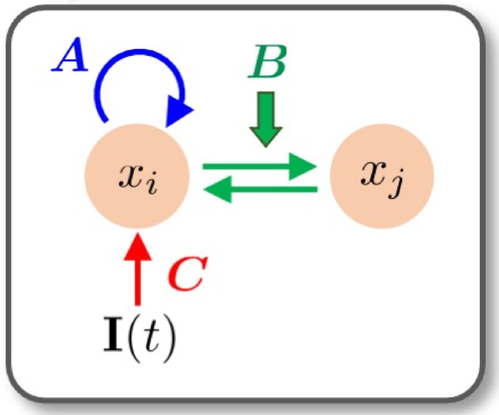
Insights about system
Learn from data
Answer counter factual
Predict in IID
Interventions

Dynamic Causal Models

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

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

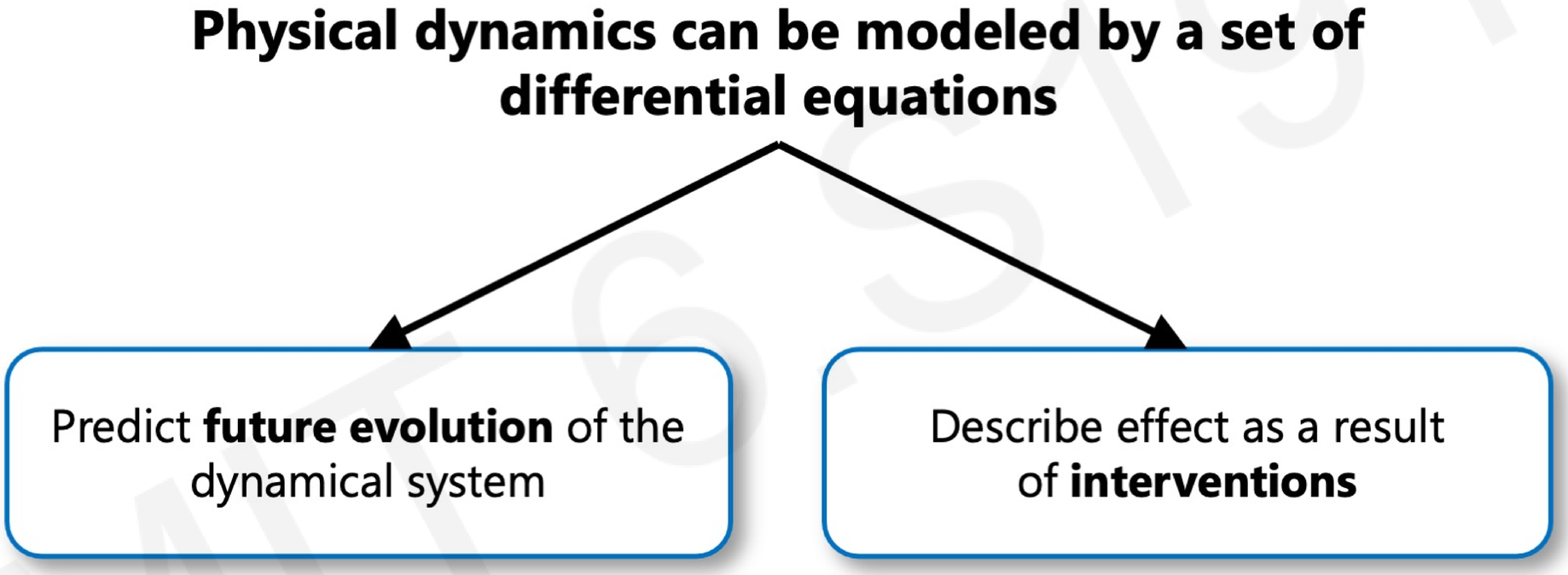
Internal coupling
Internal Intervention
External Intervention



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

Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

Differential equations can form causal structures



(Friston et al., 2003)

Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

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**

Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

Dynamic Causal Models (DCMs)

$$\frac{d\mathbf{x}}{dt} = g(\mathbf{x}(t), \mathbf{I}(t); \theta) \xrightarrow[\text{(Friston et al., 2003)}]{\textit{Bilinear approximation}} \frac{d\mathbf{x}}{dt} = (A + \mathbf{I}(t)B)\,\mathbf{x}(t) + C(\mathbf{I}(t))$$

Internal coupling

$$A = \frac{\partial g}{\partial \mathbf{x}}\bigg|_{\mathbf{I}=0}$$

Regulates hidden state

Internal Intervention

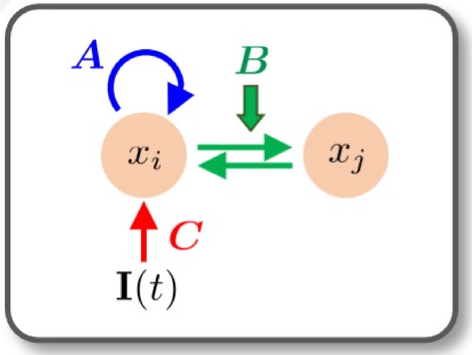
$$B = \frac{\partial^2 g}{\partial \mathbf{x}\,\partial \mathbf{I}}$$

Controls coupling sensitivity among network's nodes

External Intervention

$$C = \frac{\partial g}{\partial \mathbf{I}}\bigg|_{\mathbf{x}=0}$$

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**

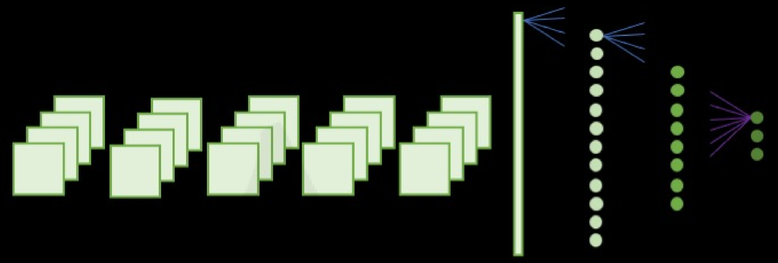
Enforced by activation constraint

(Funahashi and Nakamura, 1993)

Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

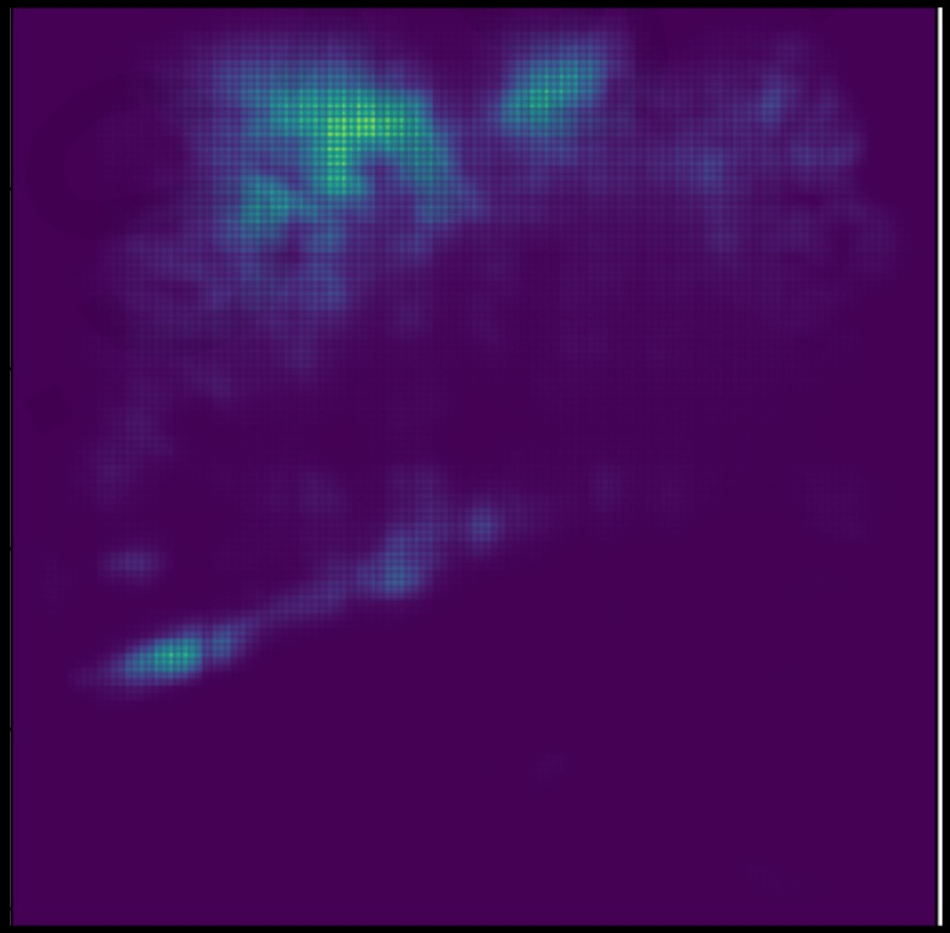
LTC-based: Neural Circuit Policies

Performance – Attention



Flying Performance

Visual Backprop Attention Map

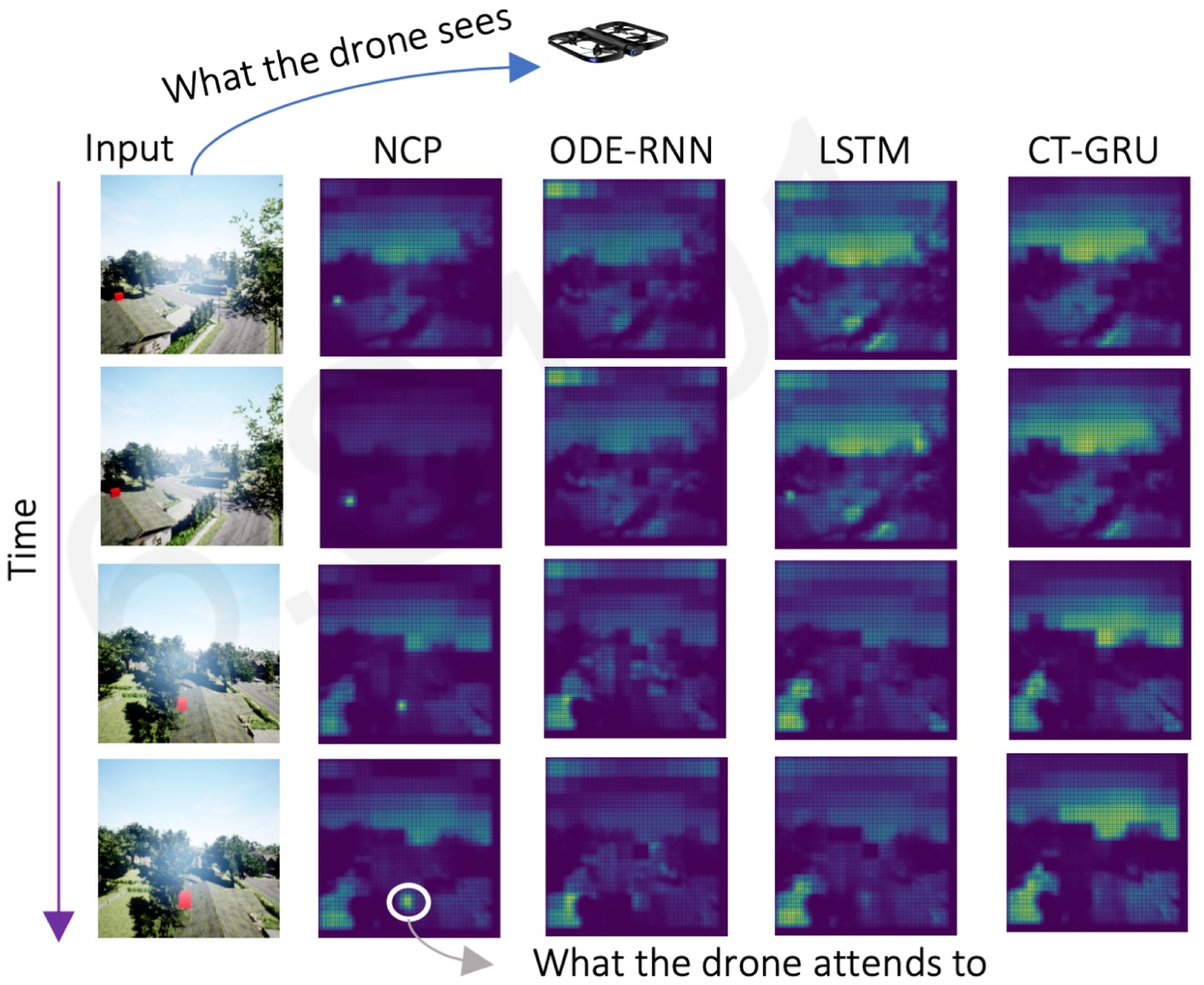


Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

Liquid Neural Networks

Performance – Attention

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

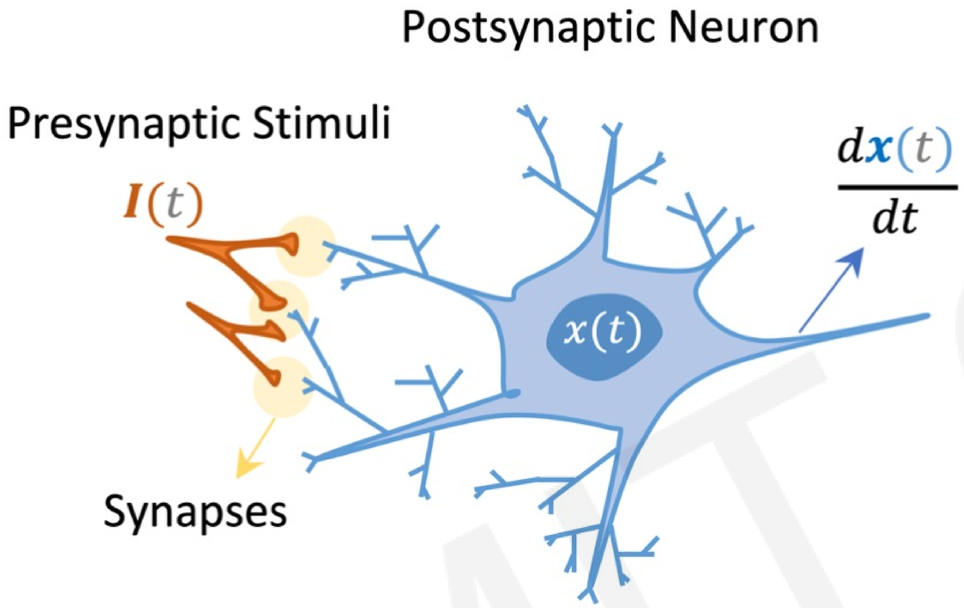


Vorbach*, Hasani*, Amini, Lechner, Rus. NeurIPS 2021

Closed-form Solution of Liquid Networks

Closed-form Continuous-time Neural Networks

https://github.com/raminmh/CfC



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

this is a liquid time-constant differential equation instance

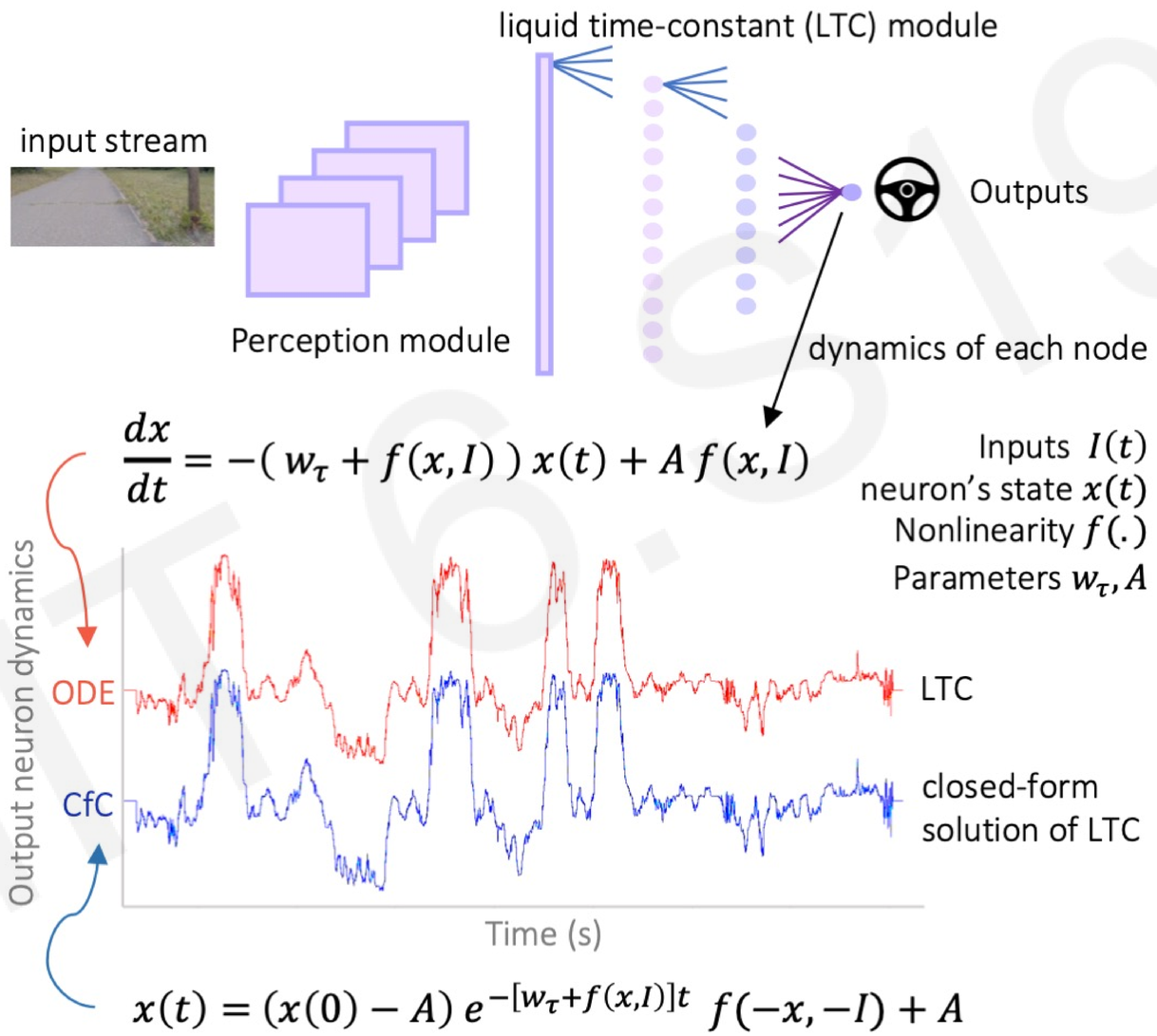
we solve this in closed-form

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

$$S(t) = f(I(t))\,(A - x(t))$$

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

Tightness of the Closed-form Solution in Practice



Hasani et al. Nature MI 2022

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

Hasani et al. Nature MI 2022

Standard Deep Neural Network

Task: Identify and navigate to target

Deploy: Closed-loop testing

Agent

Target

LTC-based Network

Task: Identify and navigate to target

Deploy: Closed-loop testing

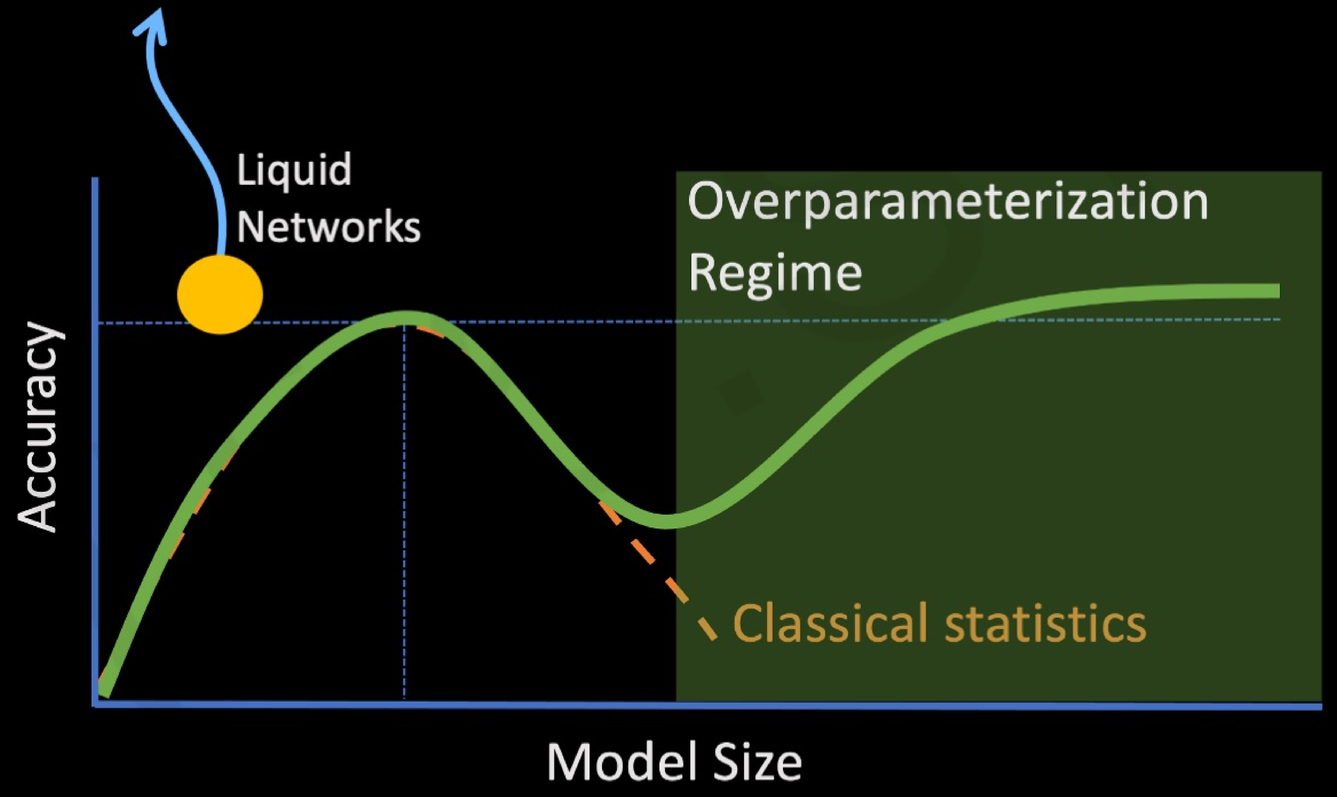
Agent

Target

4

Brain-inspired inductive biases could break the scaling law of neural networks

Generalization | More Robust | Better Reasoner | Fairer | Energy efficient | Accountable



The Modern Era of Statistics
Summary

- ✓ The law of robustness is real! $p \geq n\, d$ where d is the **effective dimensionality**
- ✓ Overparameterization improves generalization, and robustness, but does come with sociotechnical challenges (e.g., accountability, fairness and bias, energy)
- ✓ Architectural Inductive biases, and dynamic processes in neural network architectures (Liquid Neural Networks) could alleviate many of the challenges
- ✓ Liquid networks enable robust representation learning outside of overparameterization regime, as they have causal mechanisms that dramatically reduces a network's perceived effective dimensionality.

Some Resources

Get hands-on with LTC-based networks:

`github.com/mlech26l/ncps`

Get hands-on with the closed-form liquid networks:

`github.com/raminmh/CfC`

Get hands-on with Liquid-S4:

`github.com/raminmh/liquid-s4`

Get in touch:

`rhasani@mit.edu`
`ramin_hasani@vanguard.com`