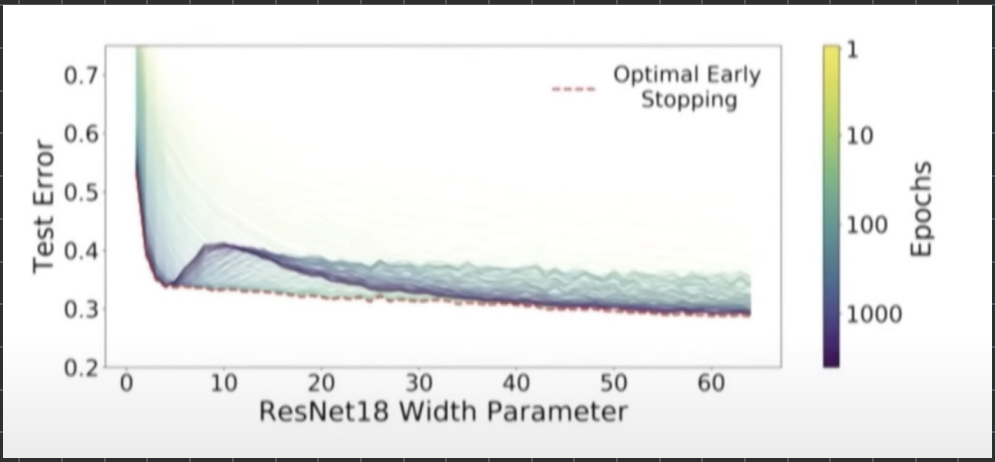
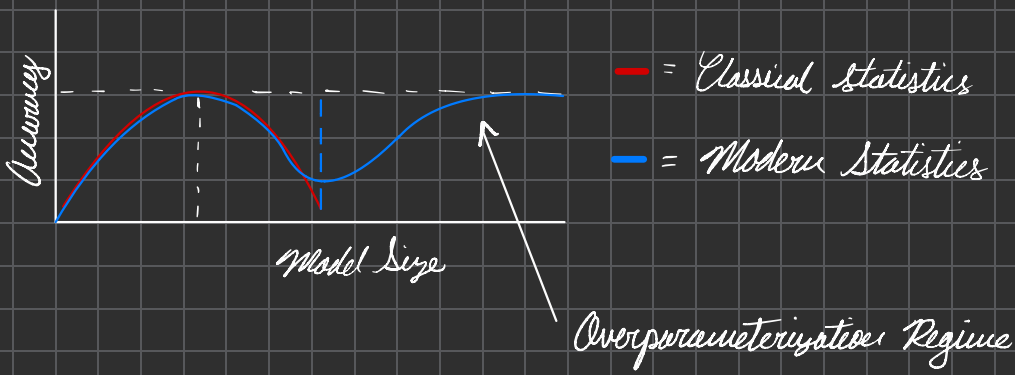
The Modern Era of Statistics

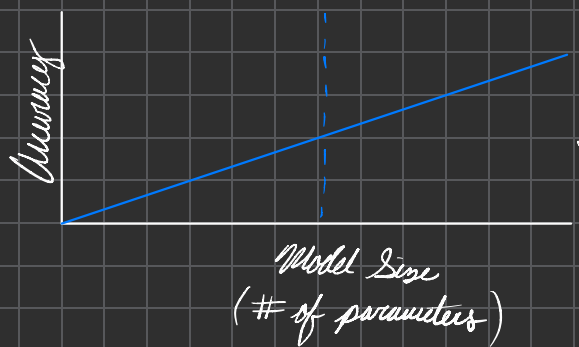
- Bigger seems to be better, but why?
- Solving n equations requires n unknowns
- MNIST: 60K data points $d = 28 \times 28$ images
  - Today, models with millions of parameters are trained on MNIST
  - Performance improves with increasing the number of parameters
  - Generalization bound $\alpha = \sqrt{\frac{\#\text{ of params}}{\text{dataset size}}}$
- ImageNet is 1.4M images of size $256 \times 256 \times 3$ and models can be hundreds of millions parameters
- NLP: Datasets are billions

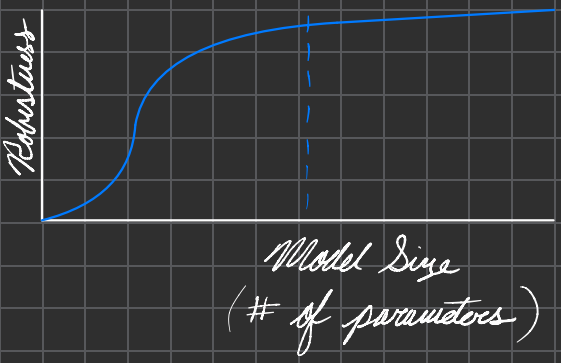
Benign Overfitting and Double Descent



Scaling does help generalization a bit!







Scaling

- Minority bias comes with scale
- Reasoning stays unchanged

Scale improves Robustness

- Using adversarial training against Projected Gradient Descent (PGD) attacks on various number of params
- Accuracy jumps from 2-4 Capacity Scale

Scale is a law of Robustness

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

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

the non-smoothness of the inner functions and their "wild behavior" has limited the practical use of the representation

Scale is a law of Robustness (cont.)

- Sample n data points from a "truly high dimensional" distribution. Add label noise
- Then, to memorize this dataset (i.e. optimize training error below the label noise level), and to do so robustly, one must necessarily have dramatic overparameterization
- $p \gtrsim nd$
- Lipschitzness (Lipschitzness): If I want to move my inputs by ϵ then I would ideally want the output also moved by ϵ

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

Why is $p \gtrsim nd$ called "Dramatic Overparameterization"?

- Intuitively, memorizing n data points is about satisfying n equations, so order n parameters should be enough
- Theorem: Two layer neural networks with threshold activation function only need ($p \approx O(n)$) to memorize binary labels.

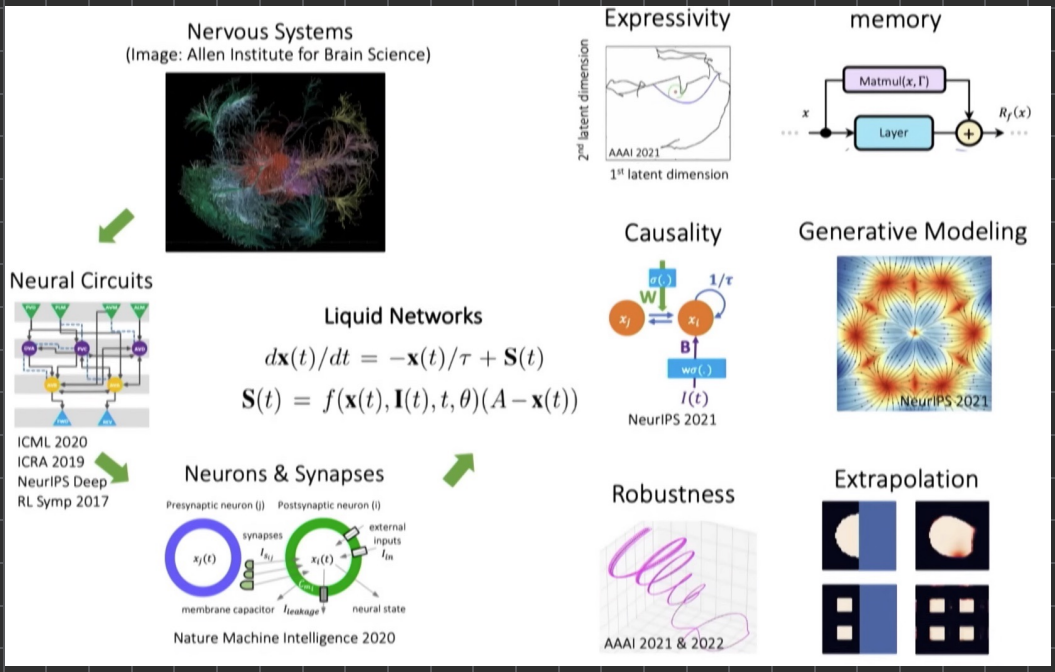
Examples in real world data

- MNIST has $n \approx 10^5$ and $d \approx 10^3$
  - Transition accuracy shift around $p \approx 10^6$
- Notion of Robustness does not match law of robustness
- $10^6 < 10^5 \cdot 10^3$ (Effective dimension: $d_{eff} \approx 10^1 \rightarrow p \approx nd_{eff}$)
- What is "noisy labels" in real data? Measuring "difficult" part of the training.

Neuroscience inspiration as inductive bias

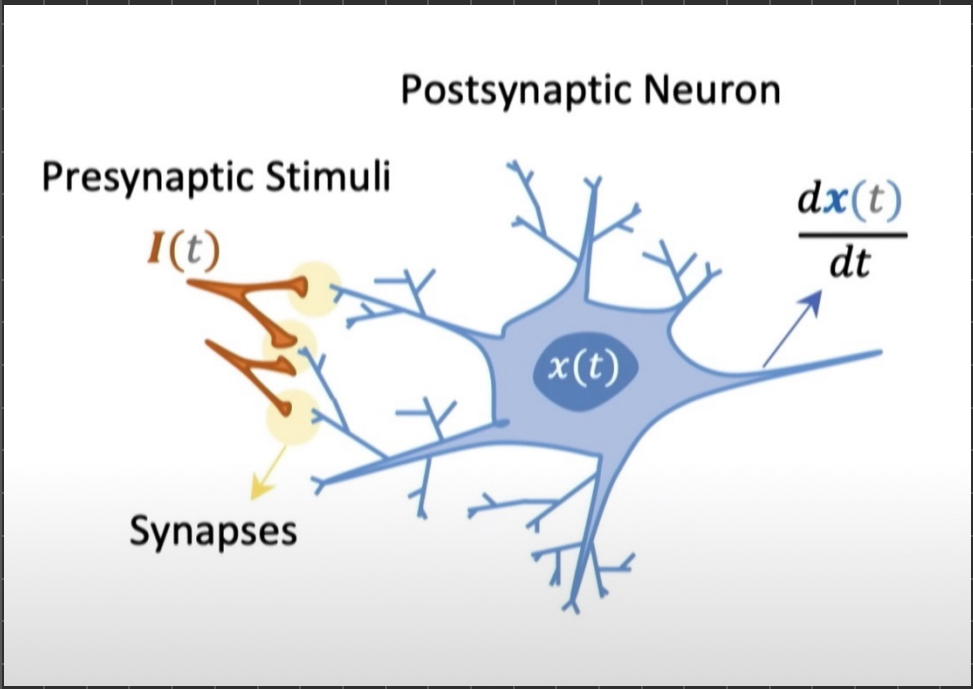
- Study the brain!

- Liquid Neural Networks



What are the building blocks of Liquid Neural Nets

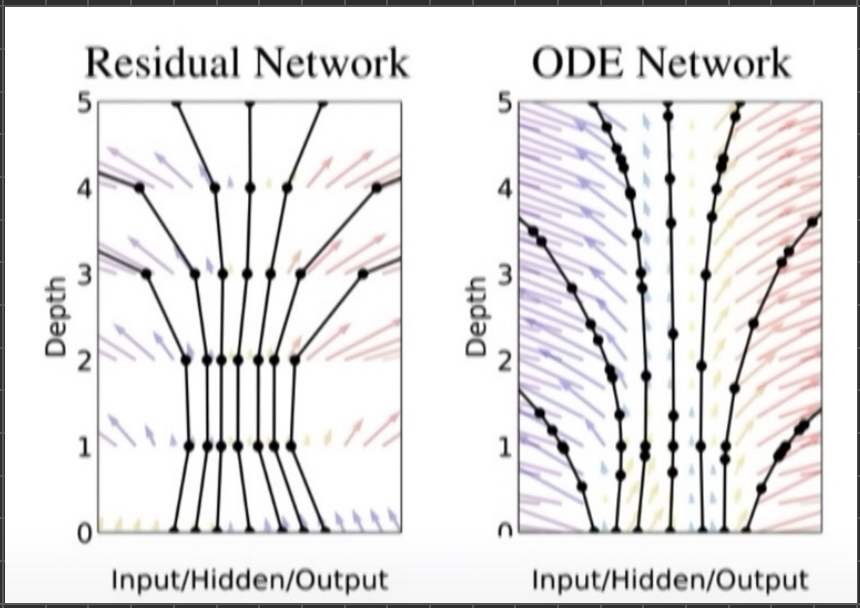
- Neural dynamics are typically continuous processes and are described by differential equations
- Synapse release is much more than scalar weights
- Recurrence, memory, and sparsity



What is a continuous-time / depth neural network?

$$\frac{dx}{dt} = f_{(n,k,\ldots)_{type}}\left(x(t), I(t), \theta\right)$$

- Number of layers
- Width
- Activations
- Hidden state
- Inputs
- Model parameters



$$h_{t+1} = h_t + f(h_t, \theta_t) \qquad \frac{dh(t)}{dt} = f(h(t), t, \theta)$$

What is a continuous-time/depth neural networks

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

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

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

*LSTM network outperforms CT-RNNs still!

Liquid Time-constant (LTC) Networks

1.) Linear state-space model

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

2.) Non-linear Synapse model

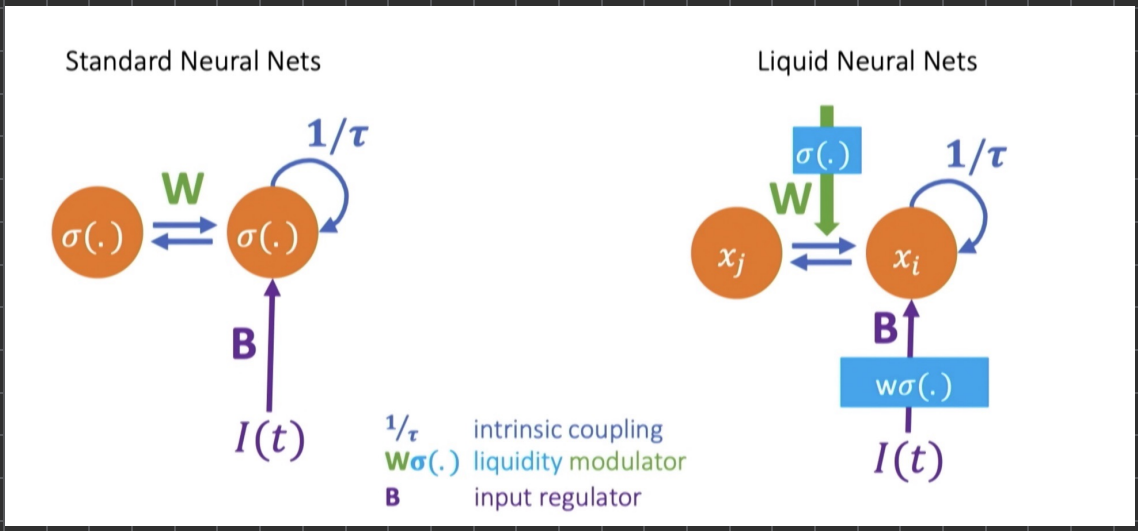
$$S(t) = f(x(t), I(t), t, \theta)(A - x(t))$$

So...

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

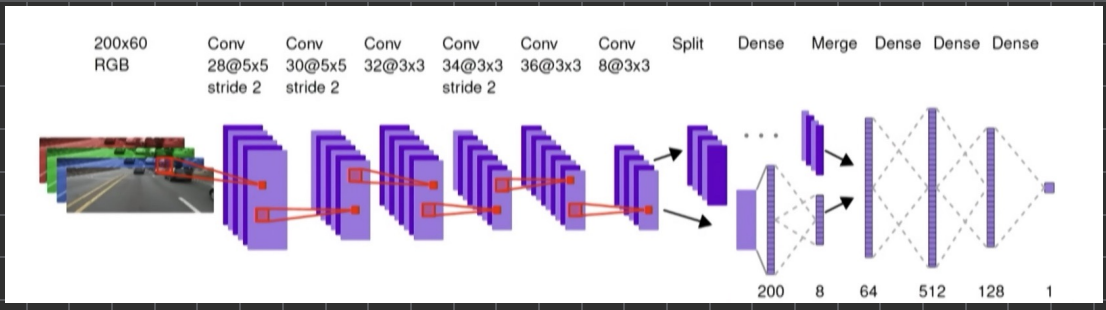
= "Liquid Variable"

Connectivity Structures

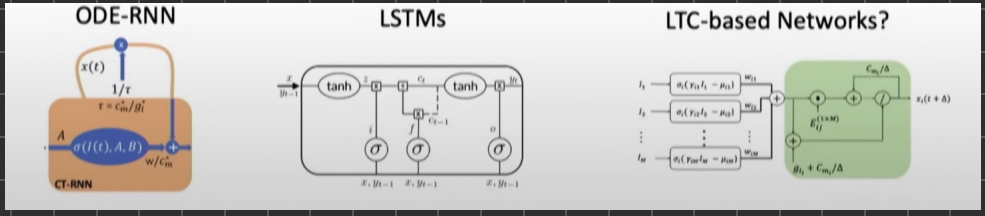


LTCs: Performance

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

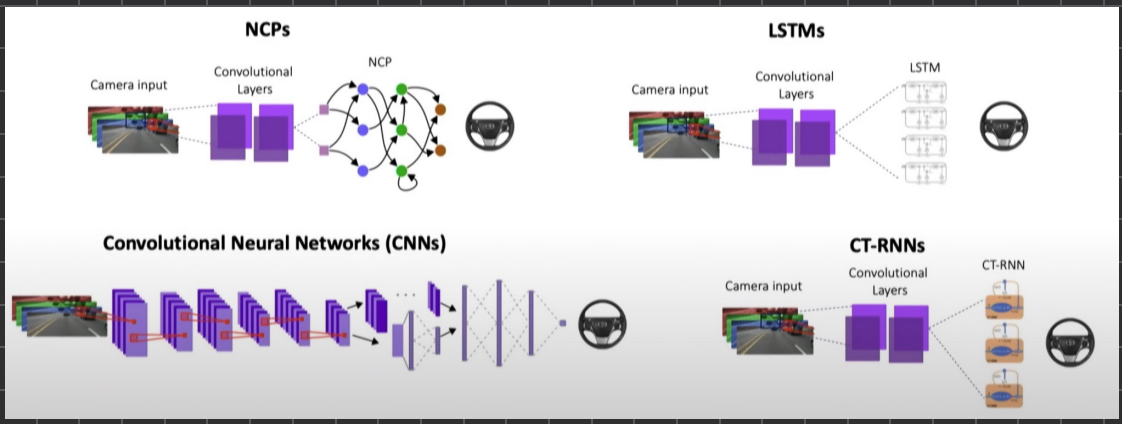


What if we replace FC layers with RNN



LTCs: Performance

Neural Circuit Policies (NCP)



Why can LTCs learn better causal relationships?

- Can focus on essence of task
- Best for physical models
- Dynamic Causal Models

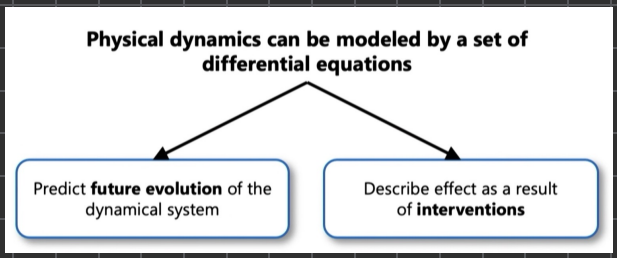
$$\frac{dx}{dt} = g(x(t), I(t); \theta)$$

$$(A + I(t)B)x(t) + C(I(t))$$

A = cluteral coupling
B = cluteral intervention
C = External intervention

- LTC reduces to dynamic causal model

Differential equations can form causal structures

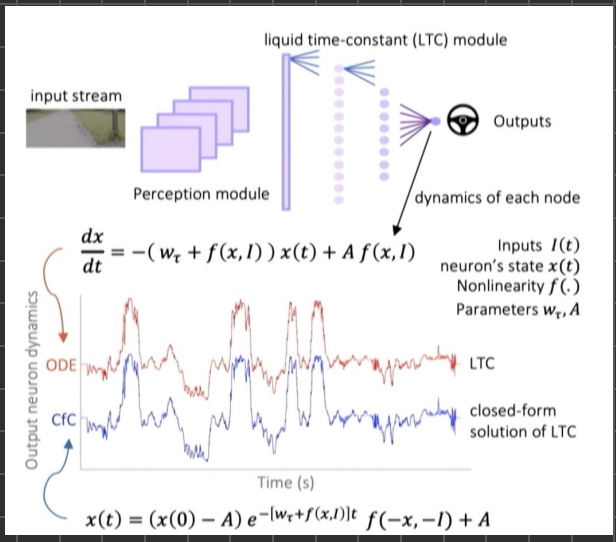


Closed form Solution of Liquid Networks

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

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

where $x(t)$ is post-synaptic neurons potential
A is synaptic reversal potential
$f(.)$ is synaptic release non-linearity
τ is post-synaptic neurons time constant



Summary

1.) Law of Robustness

- $p \geq nd$ when d is effective dimensionality

2.) Overparameterisation improves generalisation, and robustness, but does come with sociotechnical challenges (accountability, fairness, bias)

3.) Architecture inductive biases, and dynamic processes in neural network architectures (LNNs) could alleviate many of the challenges

4.) Liquid networks enable robust representation learning outside of overparameterisation regime, as they have causal mechanisms that dramatically reduces a network's perceived effective dimensionality

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`