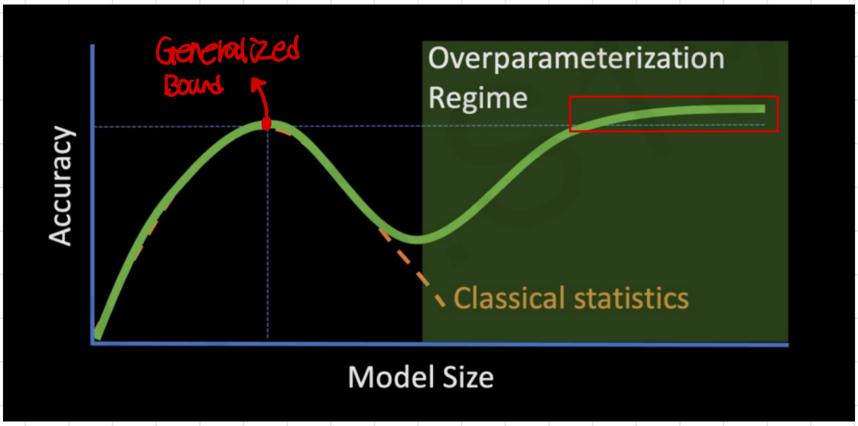
Generalized Bound

$$\propto \sqrt{\frac{\#\text{ of Parameters}}{\text{Dataset size}}}$$

← 이론적으로 이 값이 높아지면 overfitting 위험이 증가

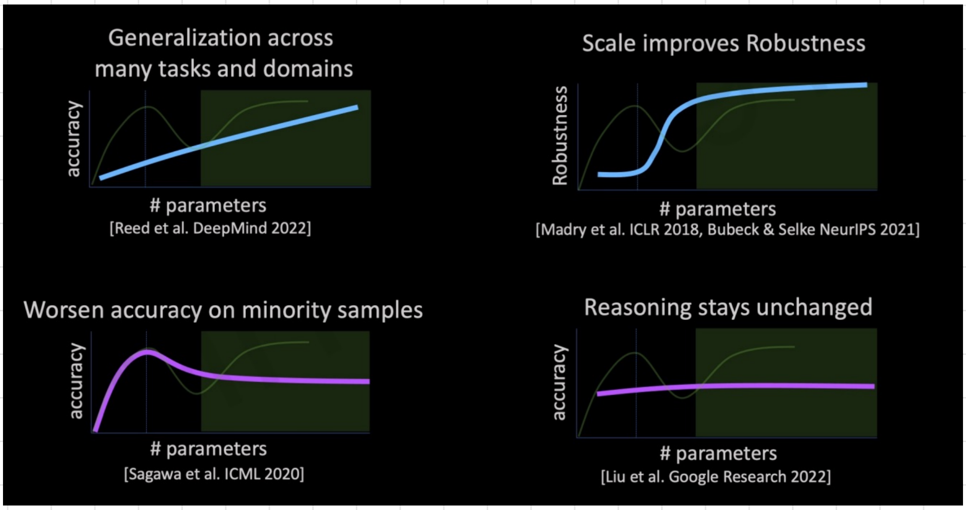
- 전통적인 머신러닝에 대해서는 잘 맞는다.
- 최근 딥러닝에서는 단순하게 Parameter 수 ↑ → 성능 ↑의 경우가 많다.



딥러닝의 경우, 이론적인 값보다 매우 큰 경우에 성능이 증가하기도 한다.

- # of parameters ∝ Model Capacity

Advantages of increasing # of parameters



- Robustness : Bias 등에 의해 변하는 정도
- Parameter numbers ↑ : Generalization, Robustness, worsen accuracy 모두 증가

Reasoning은 model 자체가 학습되지 않는 한 변하지 않는다.

Scale is a law of robustness

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

(일반적인 딥러닝 모델)

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

(극단적인 모델 제외)

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.**

(Robust를 위해 Noise 추가)

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**:

Robustness를 위해 필요한

$$p \gtrsim n\,d$$ Parameter의 수

d: 데이터 차원수, n: 데이터 샘플 개수

모든 연속함수는 단변수함수의 합으로 표현 가능하다

Kolmogorov-Arnold Representation Theorem

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

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

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

출력의 변화 ≤ 입력의 변화

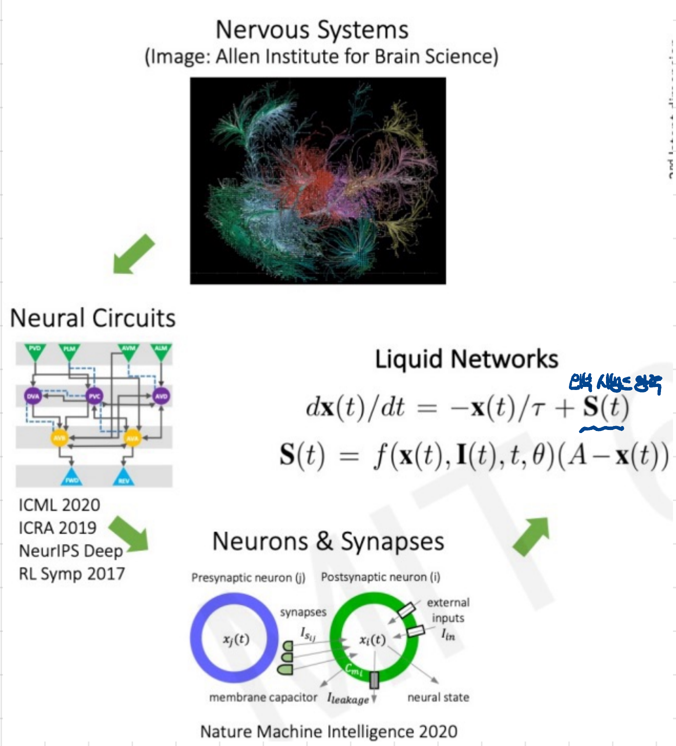
→ K가 작을수록 안정적이다. 입력의 작은 변화에 흔들리지 않음

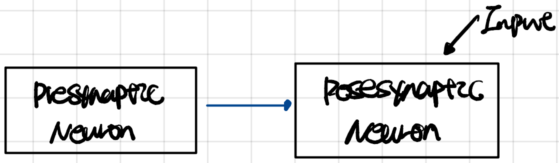
# of parameters ↑ → Robustness ↑는 증명되었는데, # of parameter ↑가 다른 것 (Energy, memory 등)을 증가시켜준다는 보장이 없다.

→ 다양한 인지적 기능을 동시에 학습시킬 수 있는 방법이 없을까?

→ Liquid Time Constant (LTC) Networks

Liquid Time Constant (LTC) Network





Liquid Time-Constant (LTC) networks

1. Linear state-space model

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

2. Non-linear synapse Model

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

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

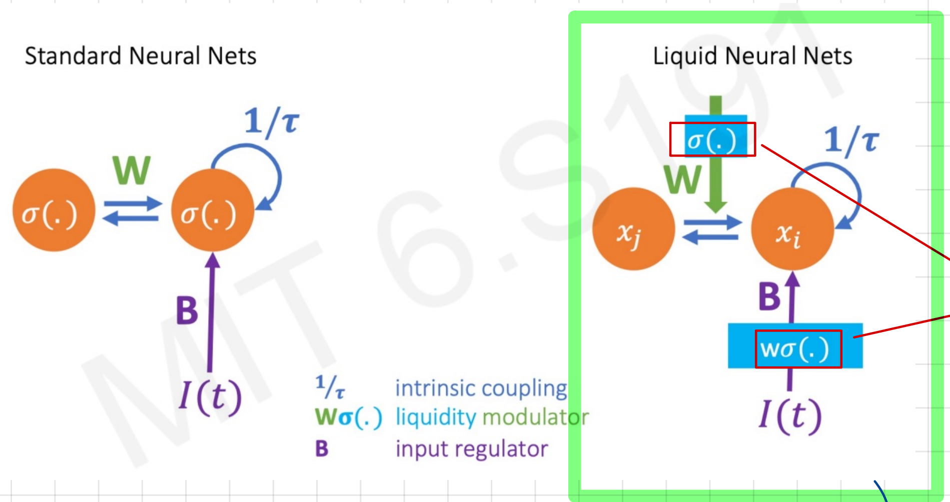
"Liquid" = variable

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

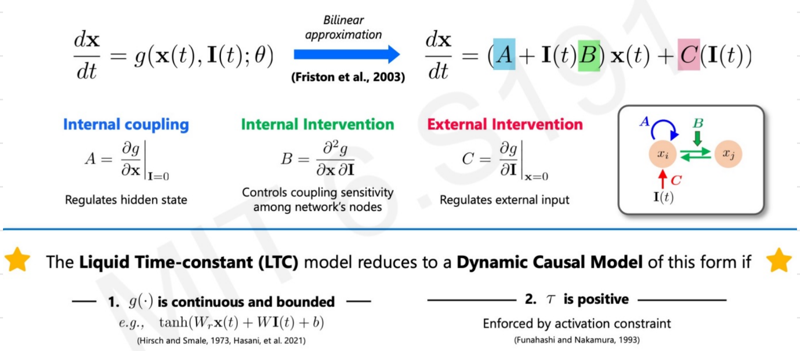
- $x(t)$: 현재 숨은 상태
- $I(t)$: 현재 Input
- A : 목표 상태
- $f(.)$: Activation function

Why LTC?

→ 연속적인 미분방정식으로 상태가 부드럽게 변함, 기억력 ↑, Long-term에 유리



Input, weight 모두 시간에 따라 변한다.
→ 유동적 ↑

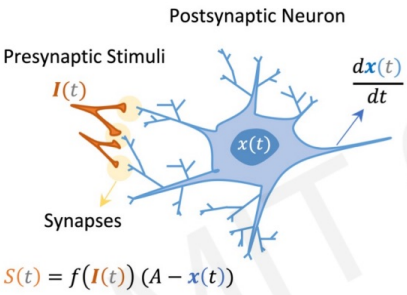


실제 Causal Model과 유사하다
→ LTC가 인과관계 학습에 강한 이유

Closed-form Solution of Liquid Networks

https://github.com/raminmh/CfC

Closed-form Continuous-time Neural Networks



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

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

→ Closed-form solution을 얻을 수 있다.

→ 해석적