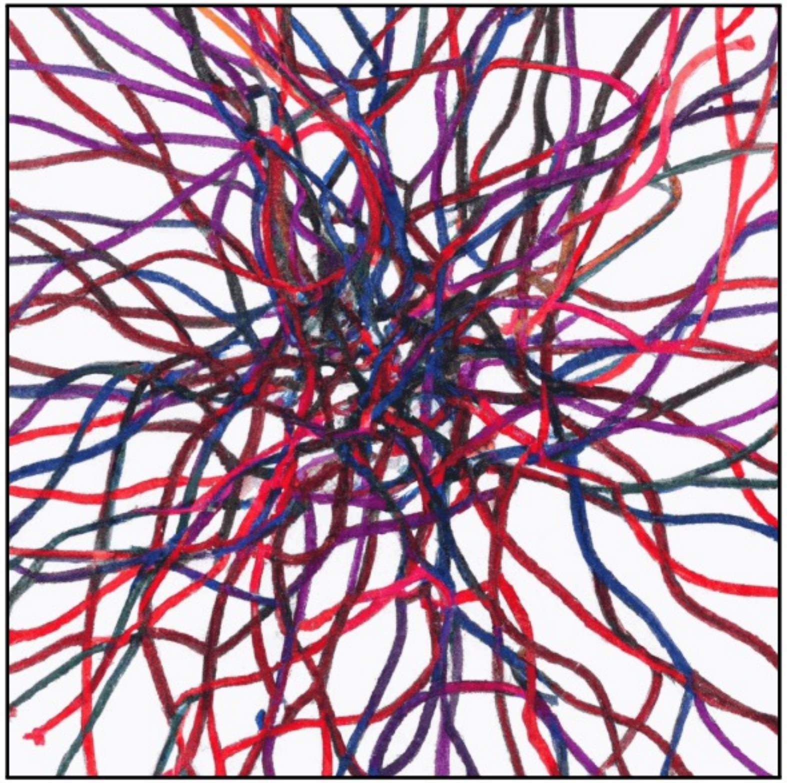
Controlling the Structure of Inference and Learning in Neural Networks

Yisong Yue

Caltech

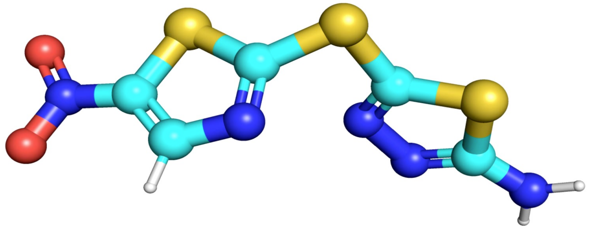


https://arxiv.org/abs/2210.10101

Machine learning is transforming science

CRISPR ML

https://www.microsoft.com/en-us/research/project/crispr/

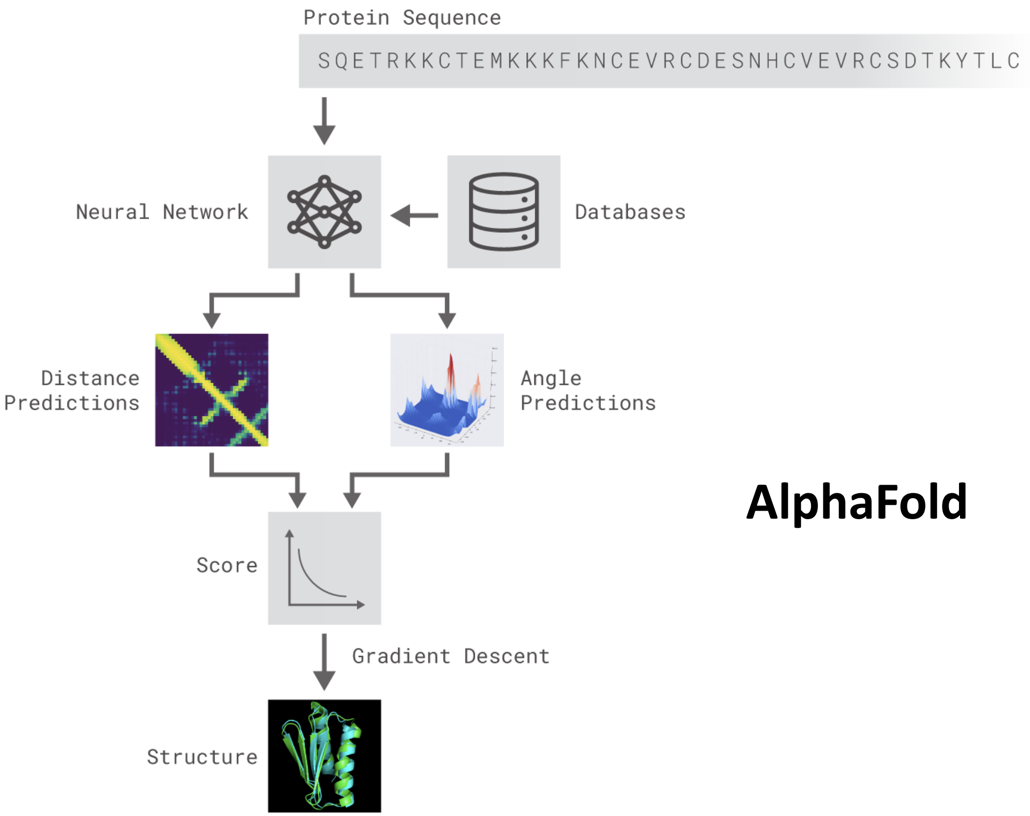


Halicin: structurally new antibiotic

https://news.mit.edu/2020/artificial-intelligence-identifies-new-antibiotic-0220

Personalized Exoskeletons

http://roams.caltech.edu/

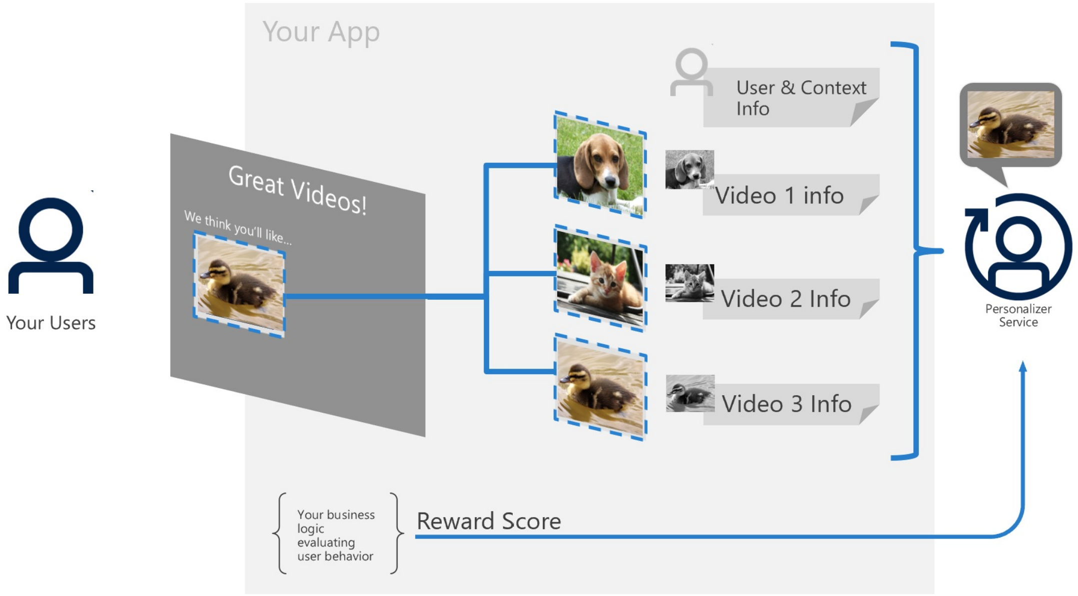


AlphaFold

...and autonomous decision-making

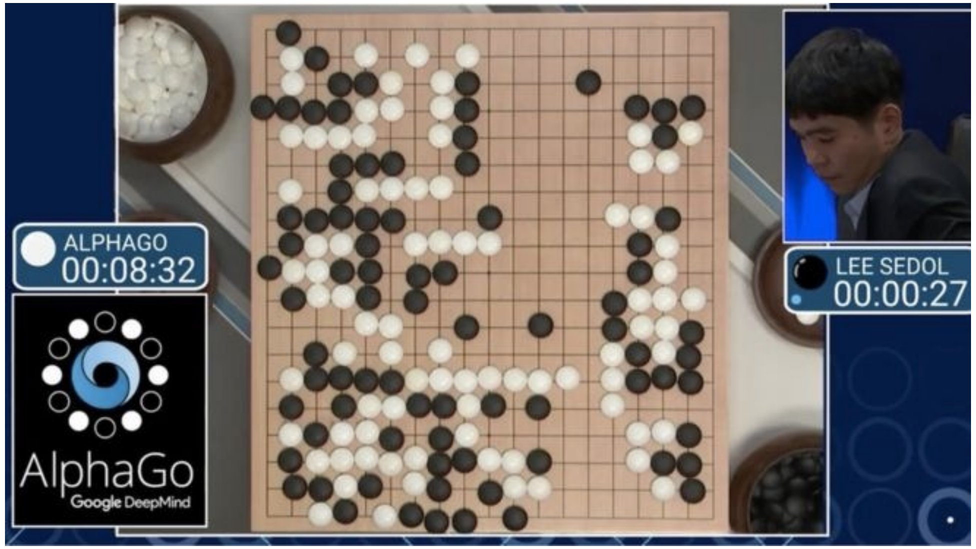
MLNav: Learning to Safely Navigate on Martian Terrains

https://arxiv.org/abs/2203.04563



Microsoft Azure Personalizer

https://learn.microsoft.com/en-us/azure/cognitive-services/personalizer/how-personalizer-works



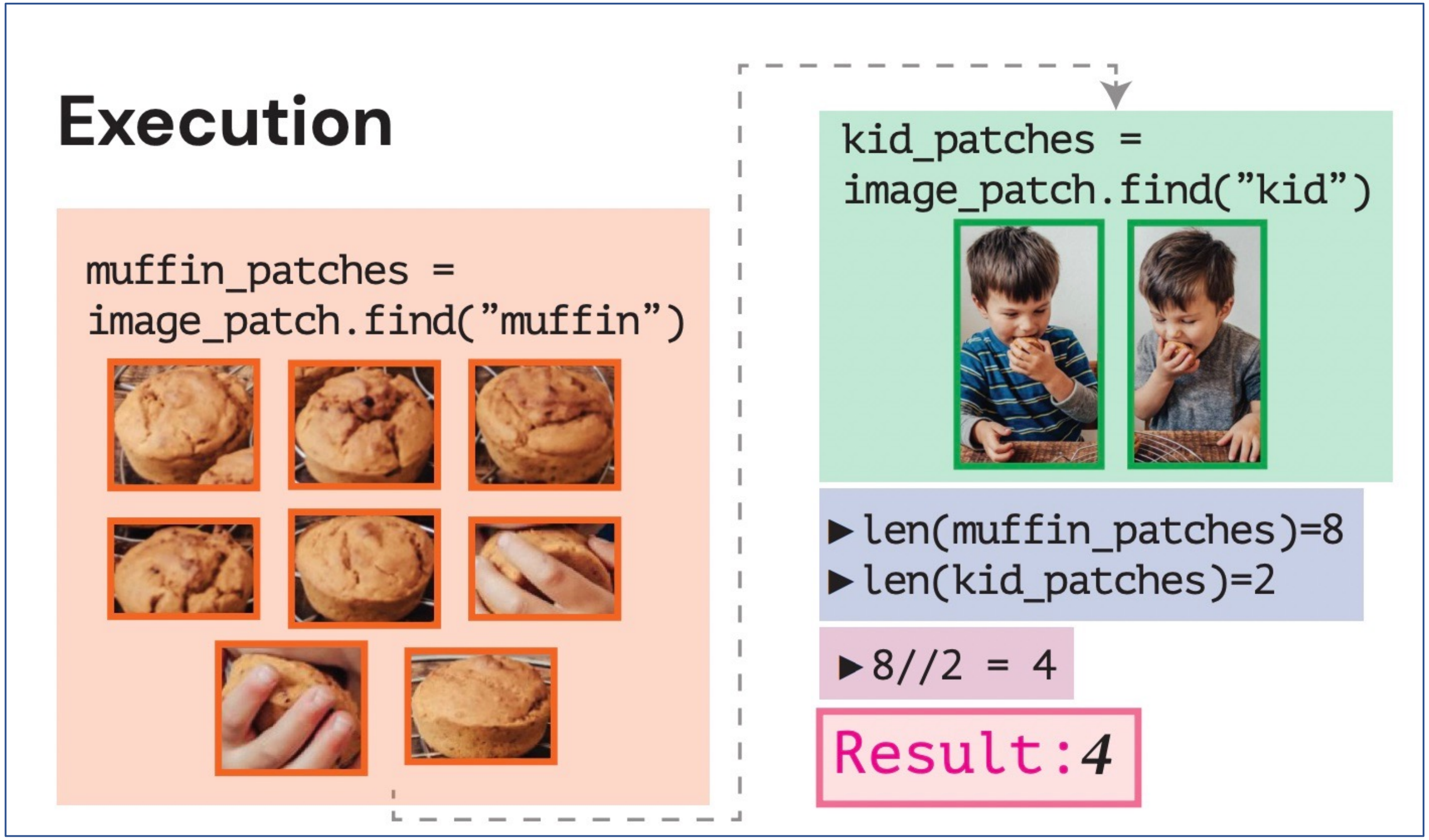
AlphaGo

…and creativity

https://www.vice.com/en/article/bvmvqm/an-ai-generated-artwork-won-first-place-at-a-state-fair-fine-arts-competition-and-artists-are-pissed

…and common-sense reasoning

Query: How many muffins can each kid have for it to be fair?



ViperGPT

https://viper.cs.columbia.edu/

Operationally: What is Machine Learning?

(Optimization Perspective)

Data: x

Learning Signal: $L_x(\cdot)$

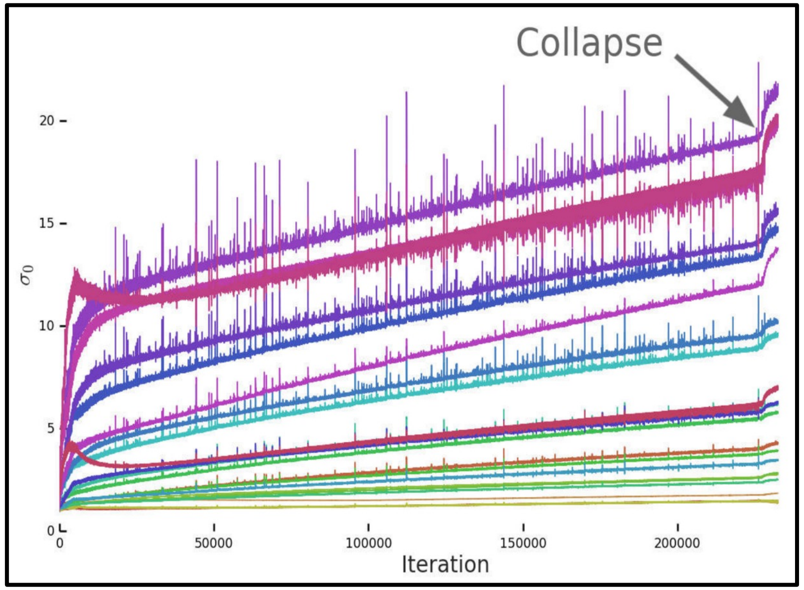
Optimization Problem:

$$\text{argmin}_\theta L(\theta) = \sum_x L_x(f(x;\theta))$$

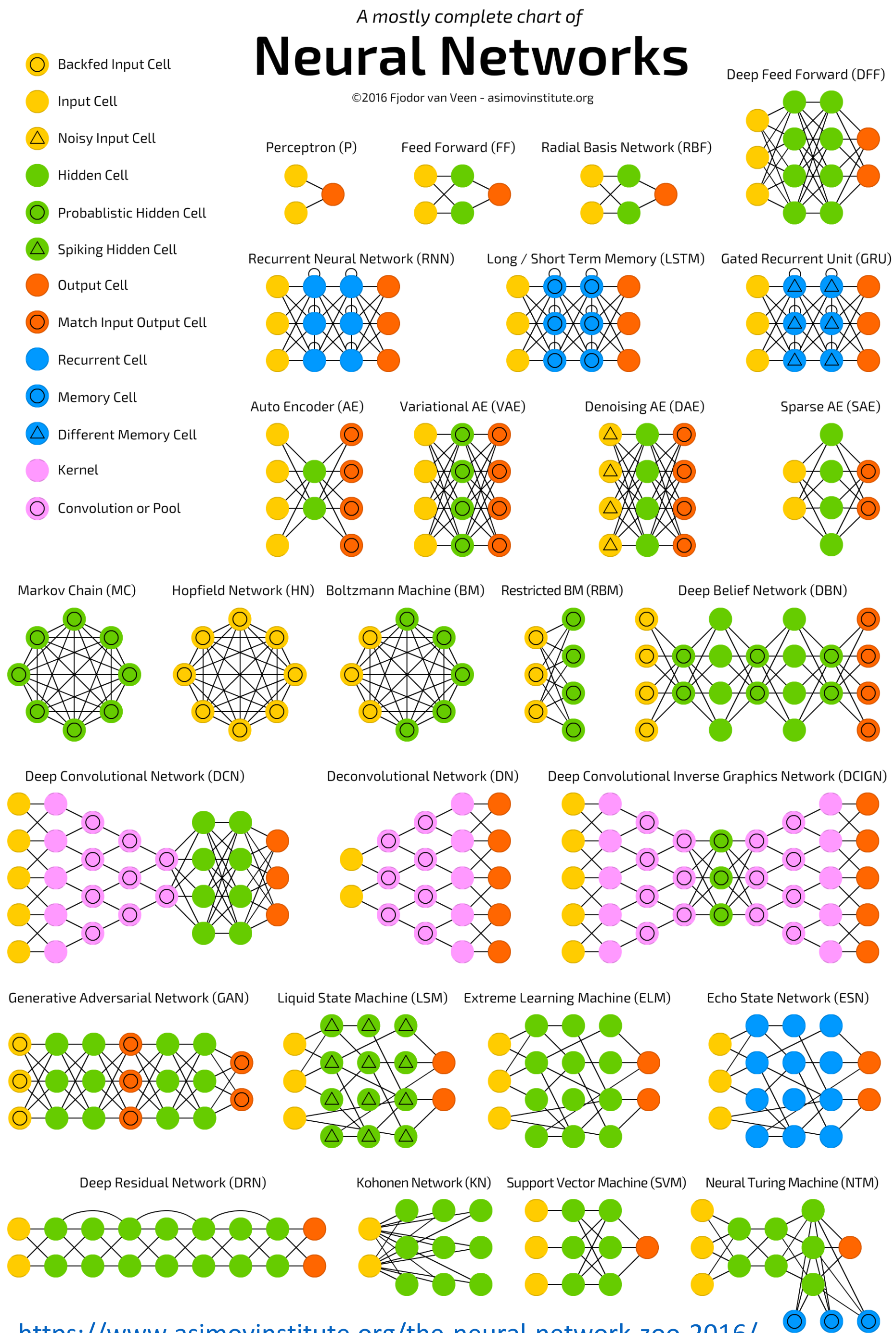
Profit!

Tuning Neural Networks is Messy and Hard

Initialization, activation, loss, architecture type, depth & width, dropout rate, optimizer, learning rate, momentum, batch size, …



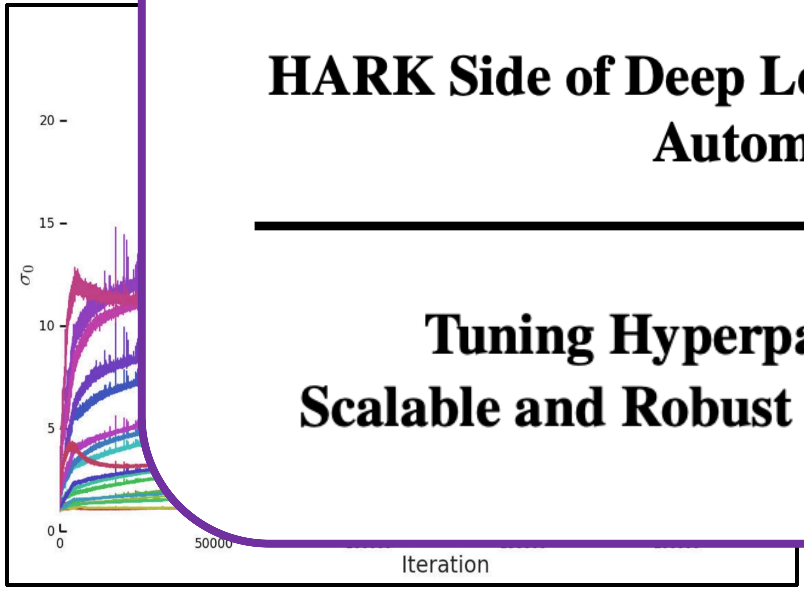
(Brock et al, 2019)



https://www.asimovinstitute.org/the-neural-network-zoo-2016/

Tuning Neural Networks is Messy and Hard

Initialization, activation, loss, architecture type, depth & width rate, momen

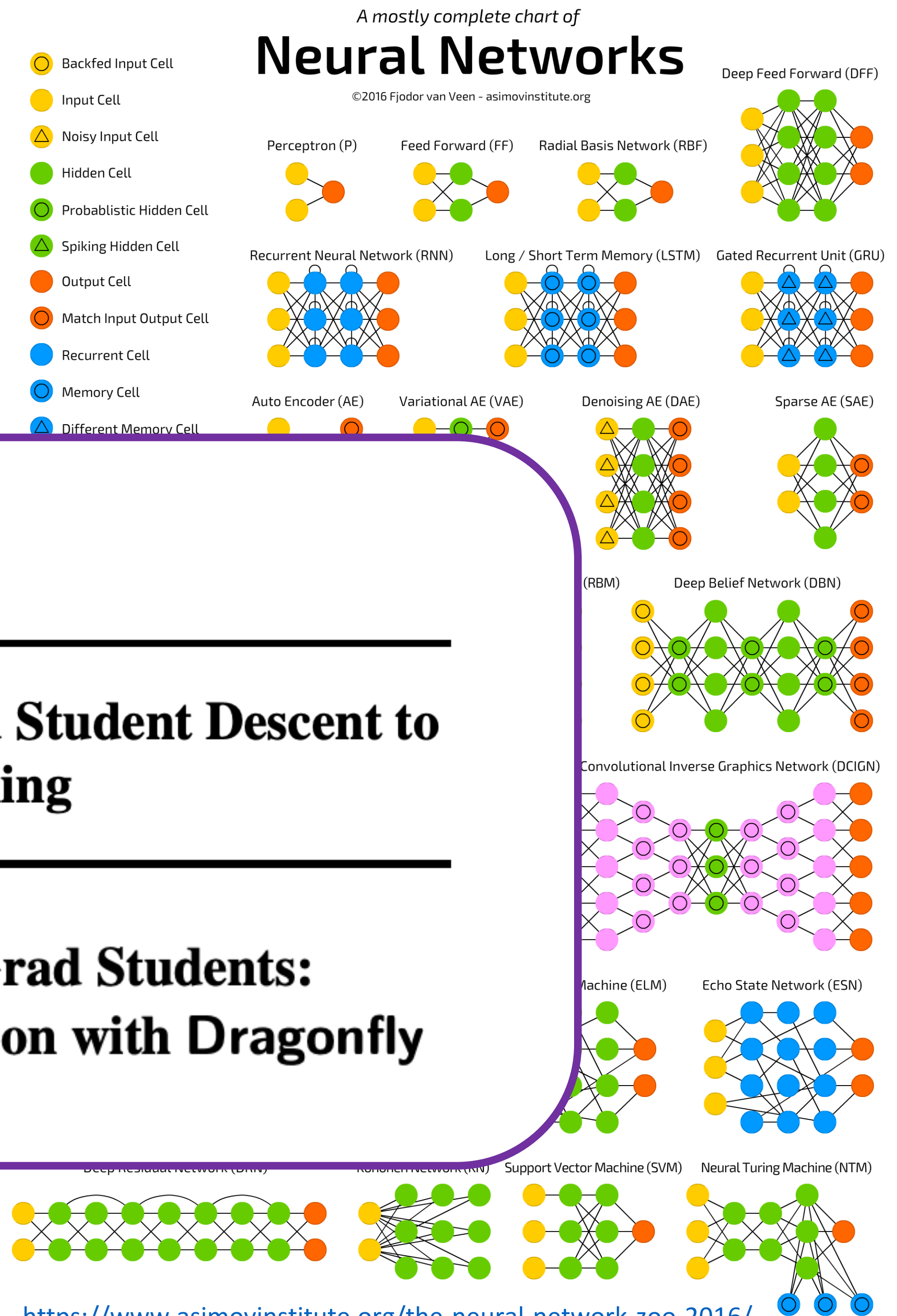


(Brock et al, 2019)

Heuristic Tuning

HARK Side of Deep Learning - From Grad Student Descent to Automated Machine Learning

Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimisation with Dragonfly



https://www.asimovinstitute.org/the-neural-network-zoo-2016/

Canonical View of ML Optimization

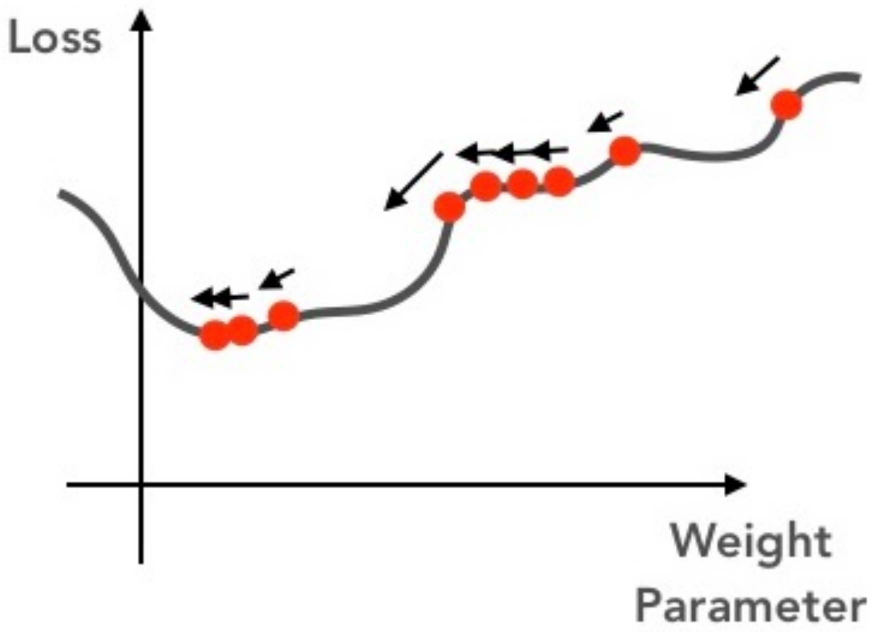
- Typical update rule:

Learning Rate (tunable hyperparameter)

$$\theta \leftarrow \theta - \eta \underbrace{\nabla L(\theta)}$$

Parameters

Gradient of Loss w.r.t. parameters



(ignoring stochastic aspect, i.e., full batch optimization)

Canonical View of ML Optimization

In Theory:

- Set η via perturbation analysis
  - How much f can change w.r.t. θ
  - (e.g., global Lipschitz constant of f)

In Practice:

- Re-tune η if we change anything about learning setup!

(ignoring stochastic aspect, i.e., full batch optimization)

Looking Inside a Neural Network

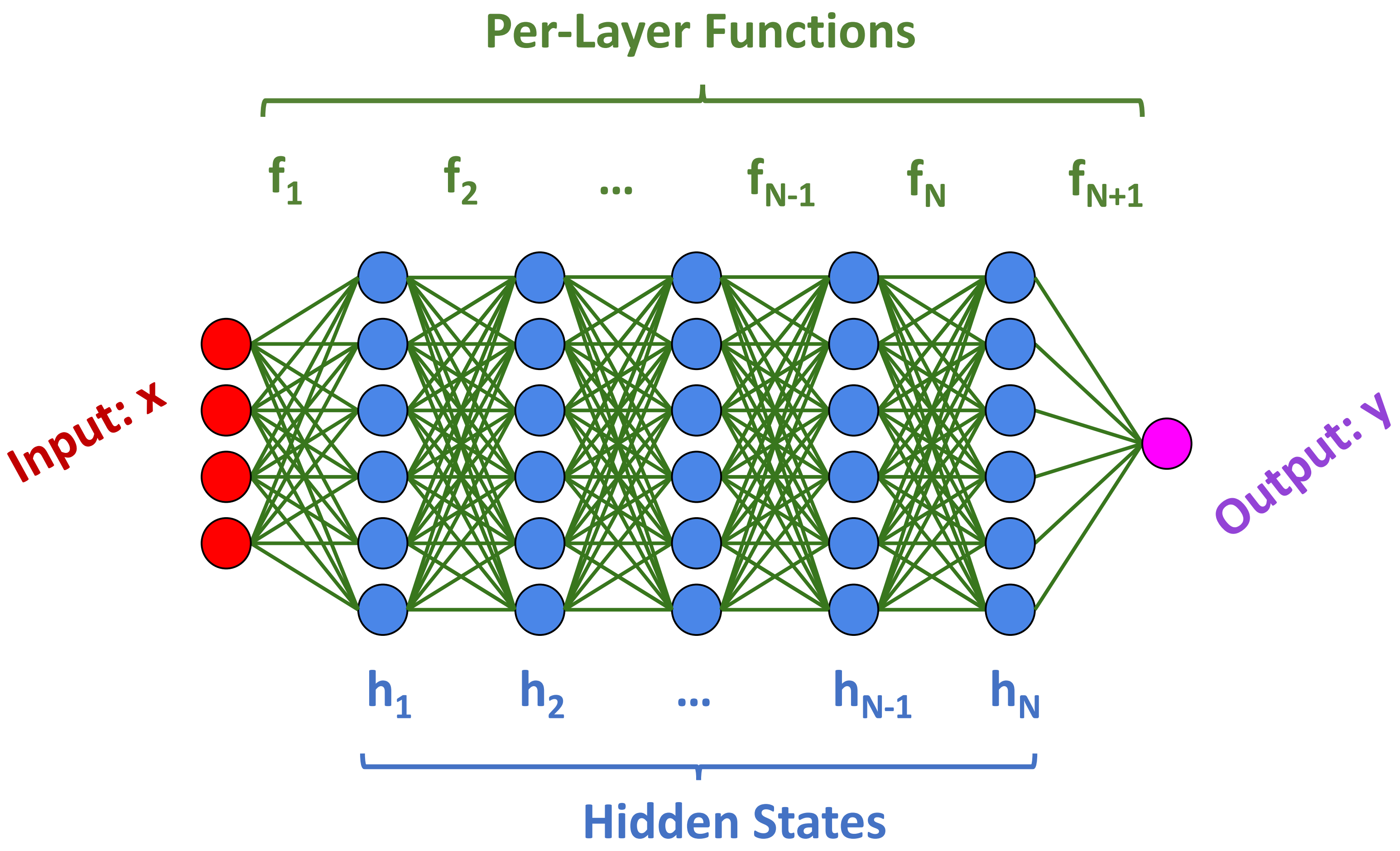
$$h_n = f_n(h_{n-1}; \theta_n)$$

Parameters

Example:
Linear w/ ReLU activation

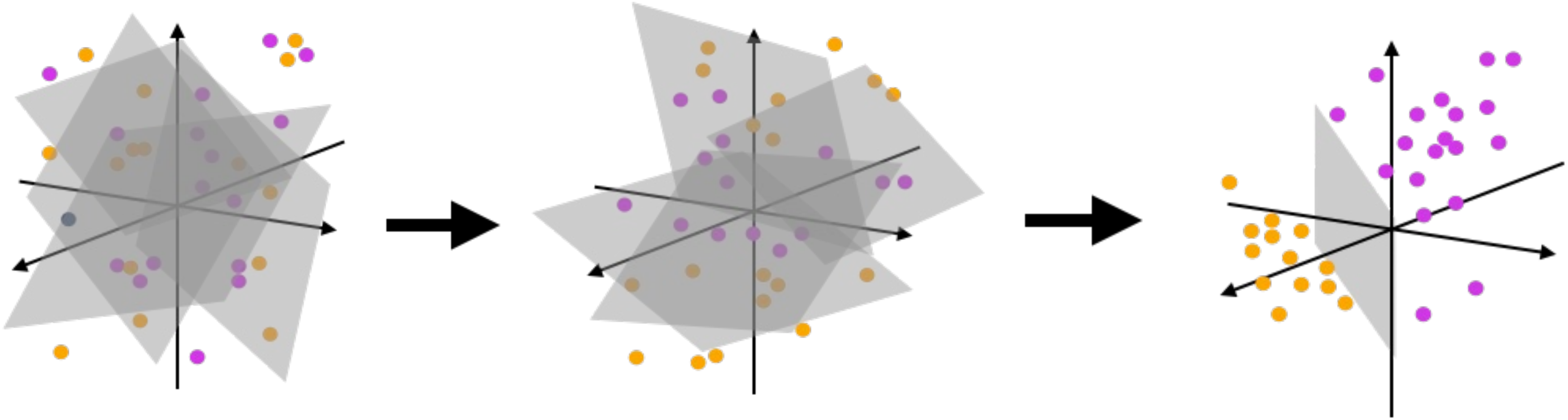
$$f(h;\theta) = [\theta^T h]_+$$

(ignoring bias/offset)



Images by Joe Marino

Intuition (for binary classification)

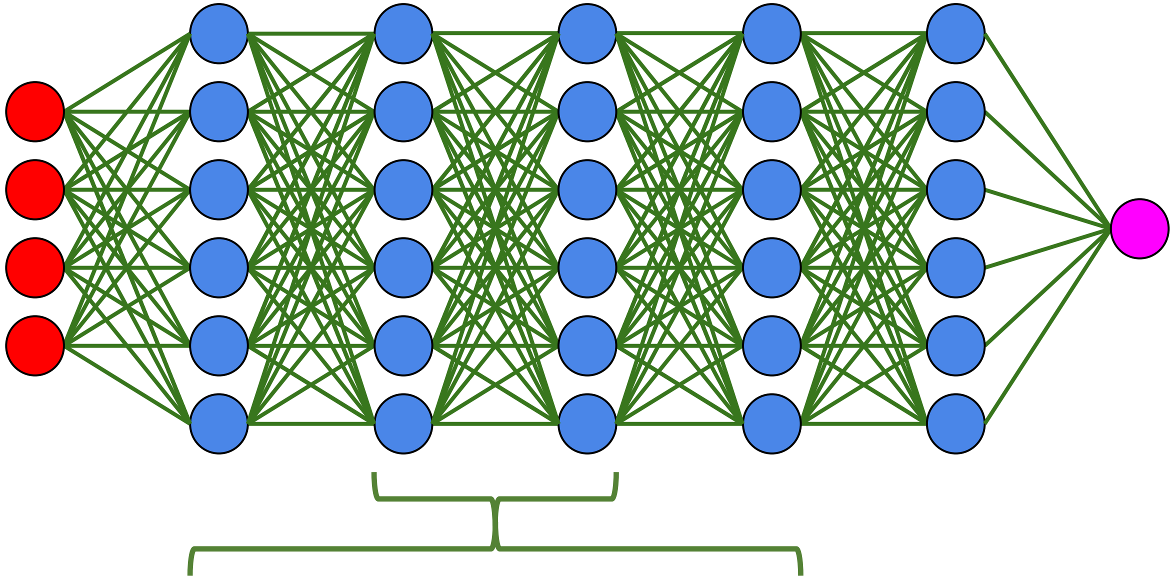


Sequence of transformations

- Each dimension is a half-space mapping
- Goal: last layer is a separable space with perfect classification

Idea #1: Per-Layer Perturbation Analysis

- How does the layer’s function change under parameter perturbation?

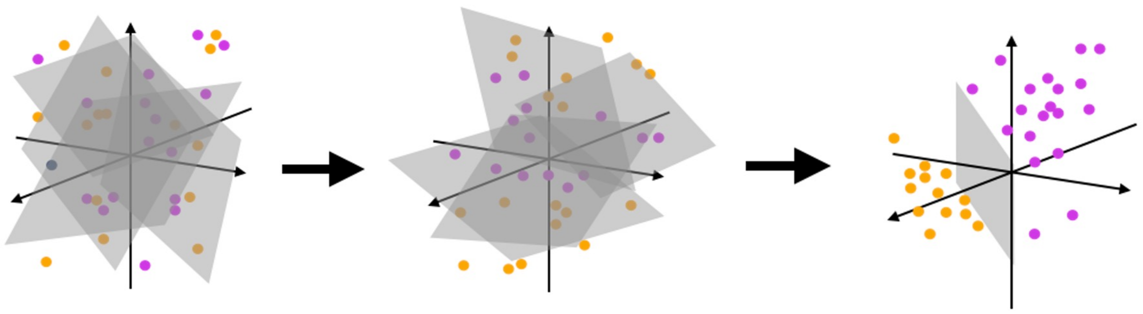


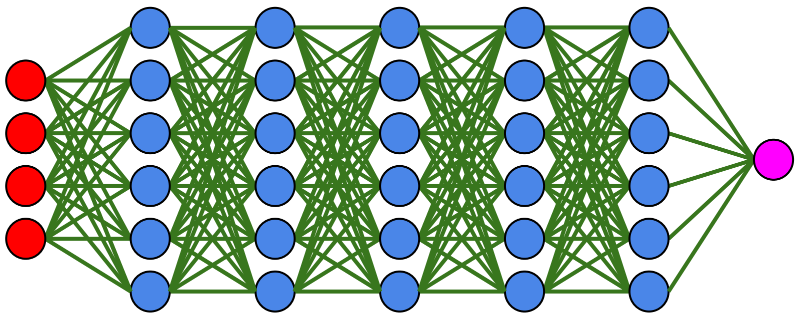
$$\theta_n \quad \text{vs} \quad \theta_n + \Delta\theta_n \longleftarrow \text{perturbation}$$

Potential Application: depth- & width-invariant learning rate η

Idea #2: Control Dynamics of Hidden Layers

Recall: we want each layer to push representation towards good answer



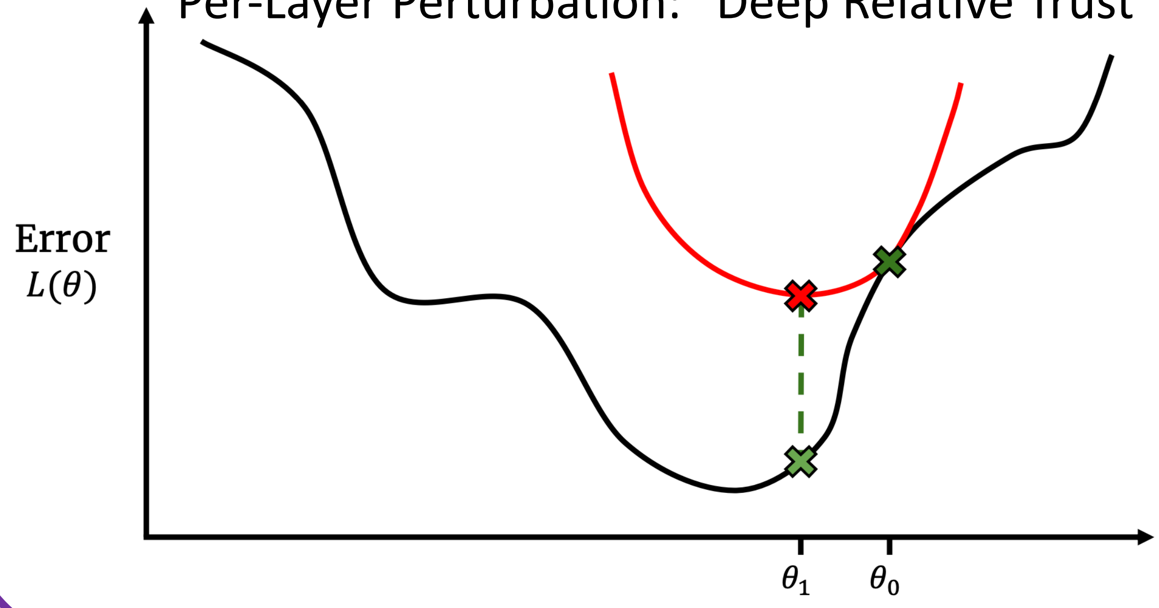


Goal: control sequence of hidden layers $h_1, \ldots, h_N$

- quickly and robustly converge to low loss

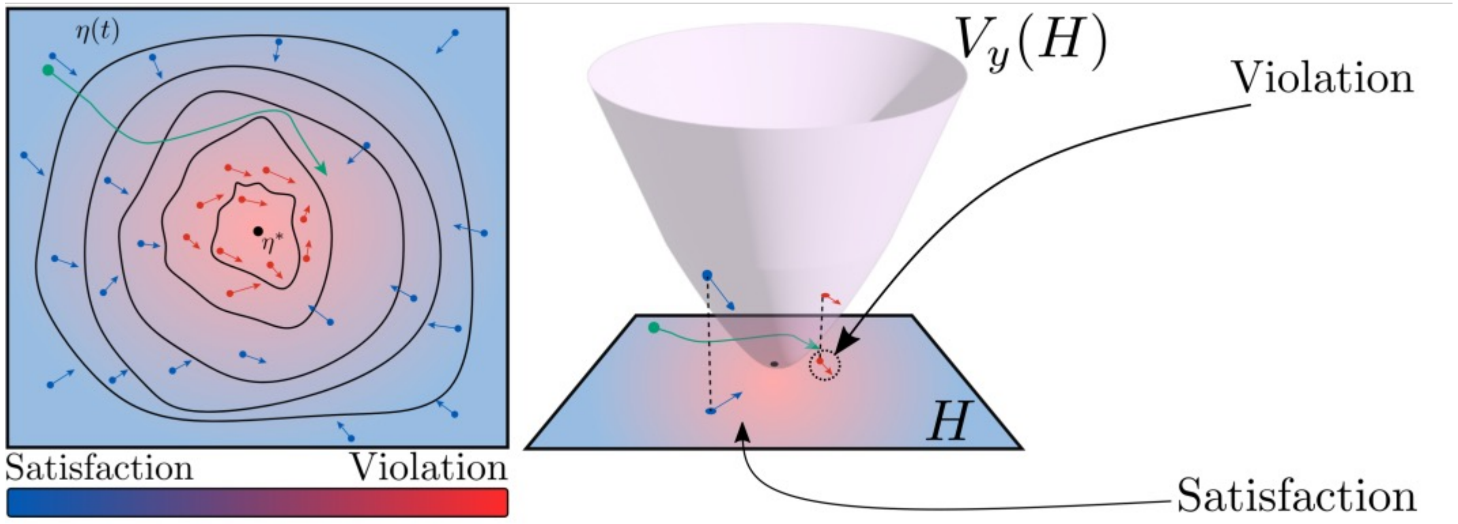
Majorize-Minimize Framework

Per-Layer Perturbation: “Deep Relative Trust”



How to
exploit the structure
of NNs to develop
a more nuanced theory?

Control-Theoretic Shaping
of Neural ODEs
“Lyapunov Loss”

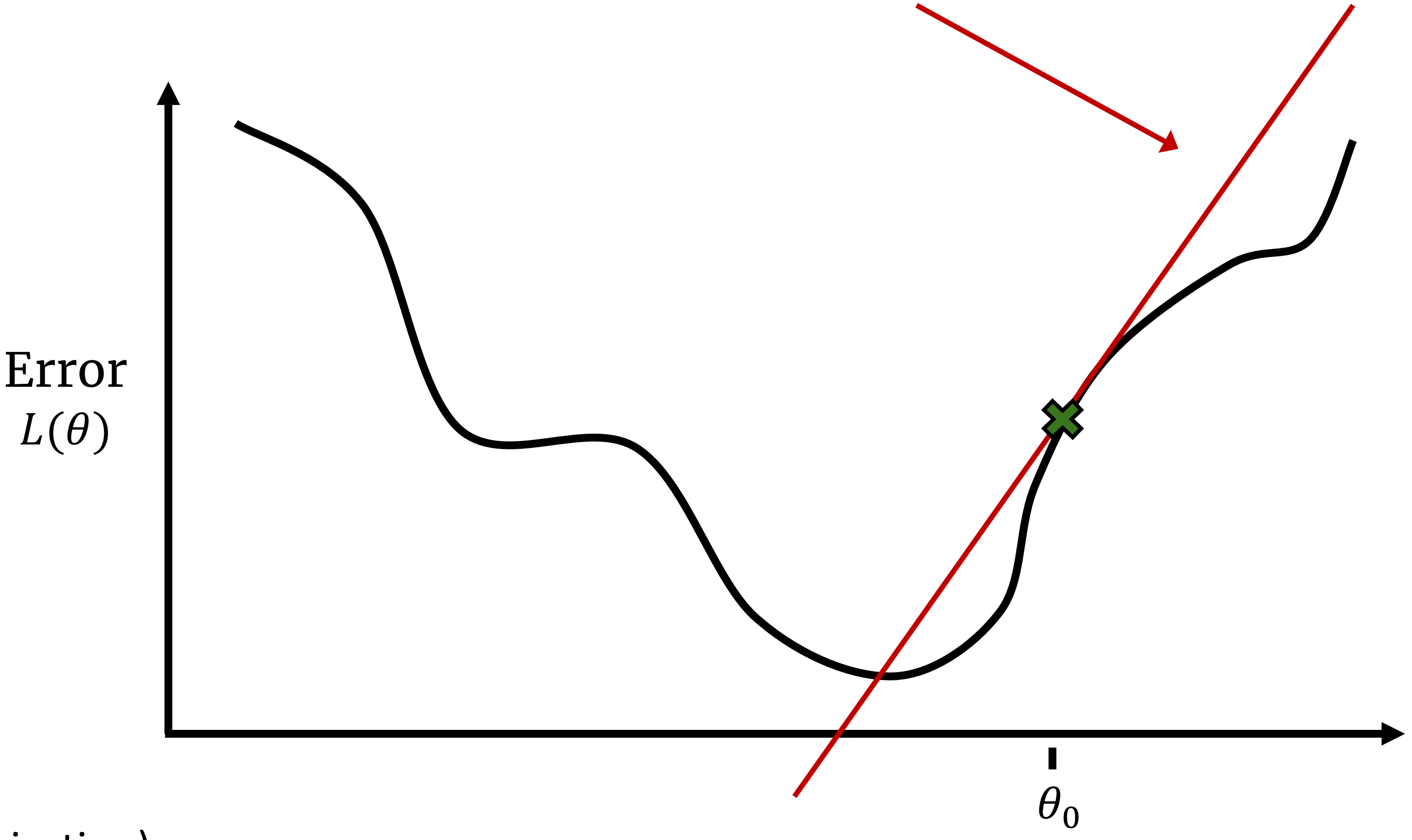


Warm-Up: Local Perturbation Analysis

- **Linear approximation breaks down as $\Delta\theta$ increases!**
- **Understand rate of break-down via perturbation analysis.**

Taylor Expansion:

$$L(\theta + \Delta\theta) = \underbrace{L(\theta) + \nabla_\theta L(\theta)^T \Delta\theta}_{\text{Linear Approximation}} + \frac{1}{2}\Delta\theta^T \nabla_\theta^2 \Delta\theta + \ \ldots$$



(ignoring stochastic aspect, i.e., full batch optimization)

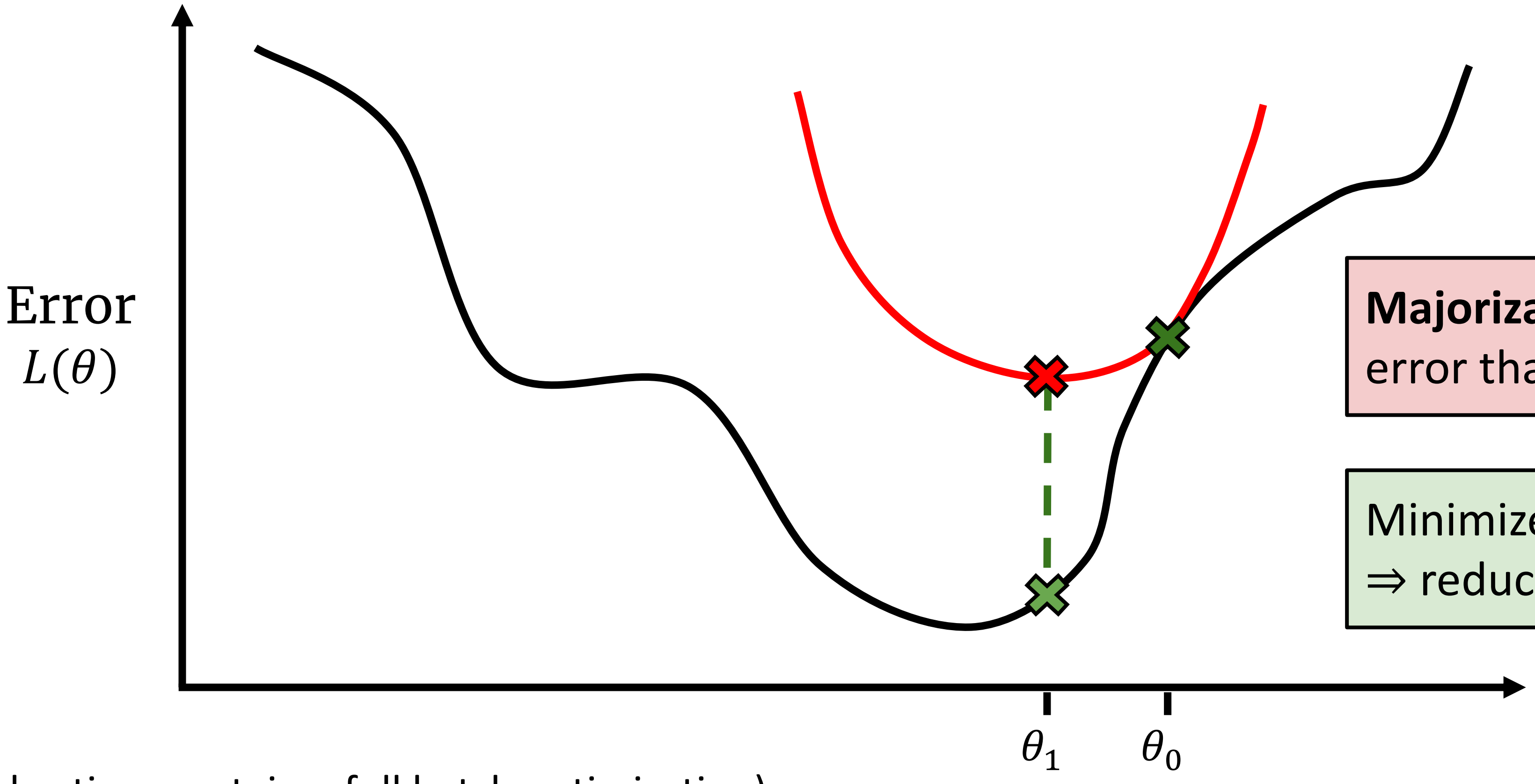
Majorize-Minimize Framework

https://en.wikipedia.org/wiki/MM_algorithm

Order-k Taylor approximation (k=1 for linear)

Typical Form: $L(\theta + \Delta\theta) \leq L^{(k)}(\theta + \Delta\theta) + \psi_\theta(\Delta\theta)$

Upper bound of the rest



Majorization: upper bound on error that lies tangent.

Minimize majorization $\Rightarrow$ reduces error.

(ignoring stochastic aspect, i.e., full batch optimization)

Majorize-Minimize in Action

Order-k Taylor Approx　　Upper bound of rest

Example 1: $\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta} + \boxed{\lambda|\Delta\theta|^2}$ $\Rightarrow$ *gradient descent.*

D = Bregman Divergence

Example 2: $\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta} + \boxed{\lambda \mathrm{D}(\theta + \Delta\theta, \theta)}$ $\Rightarrow$ *mirror descent.*

H = Hessian

Example 3: $\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta + \frac{1}{2}\Delta\theta^T H \Delta\theta} + \boxed{\lambda|\Delta\theta|^3}$

$\Rightarrow$ *cubic regularized Newton.*

(ignoring stochastic aspect, i.e., full batch optimization)

Aside: Duality of Majorization & Trust Regions

$$\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta} + \boxed{\lambda|\Delta\theta|^2}$$

- Closed-form solution: $\Delta\theta = -\left(\frac{2}{\lambda}\right)\nabla_\theta L(\theta)$

Learning Rate

Theoretical guidance via perturbation sensitivity, e.g., Lipschitz constant

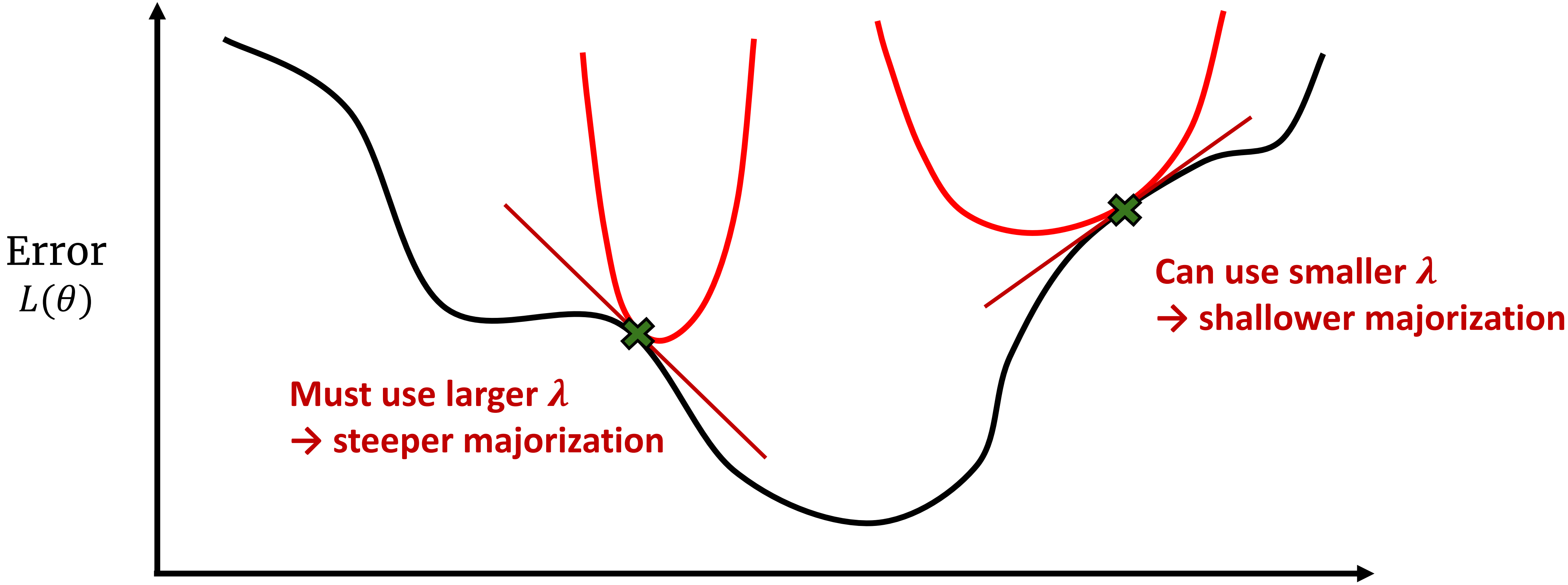
- Implies GD update rule: $\theta \leftarrow \theta - \left(\frac{2}{\lambda}\right)\nabla_\theta L(\theta)$
- Analogous to: $\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta}$ s.t. $\boxed{|\Delta\theta|^2 \leq C}$

"Trust Region"

Architecture-Naive Majorizations

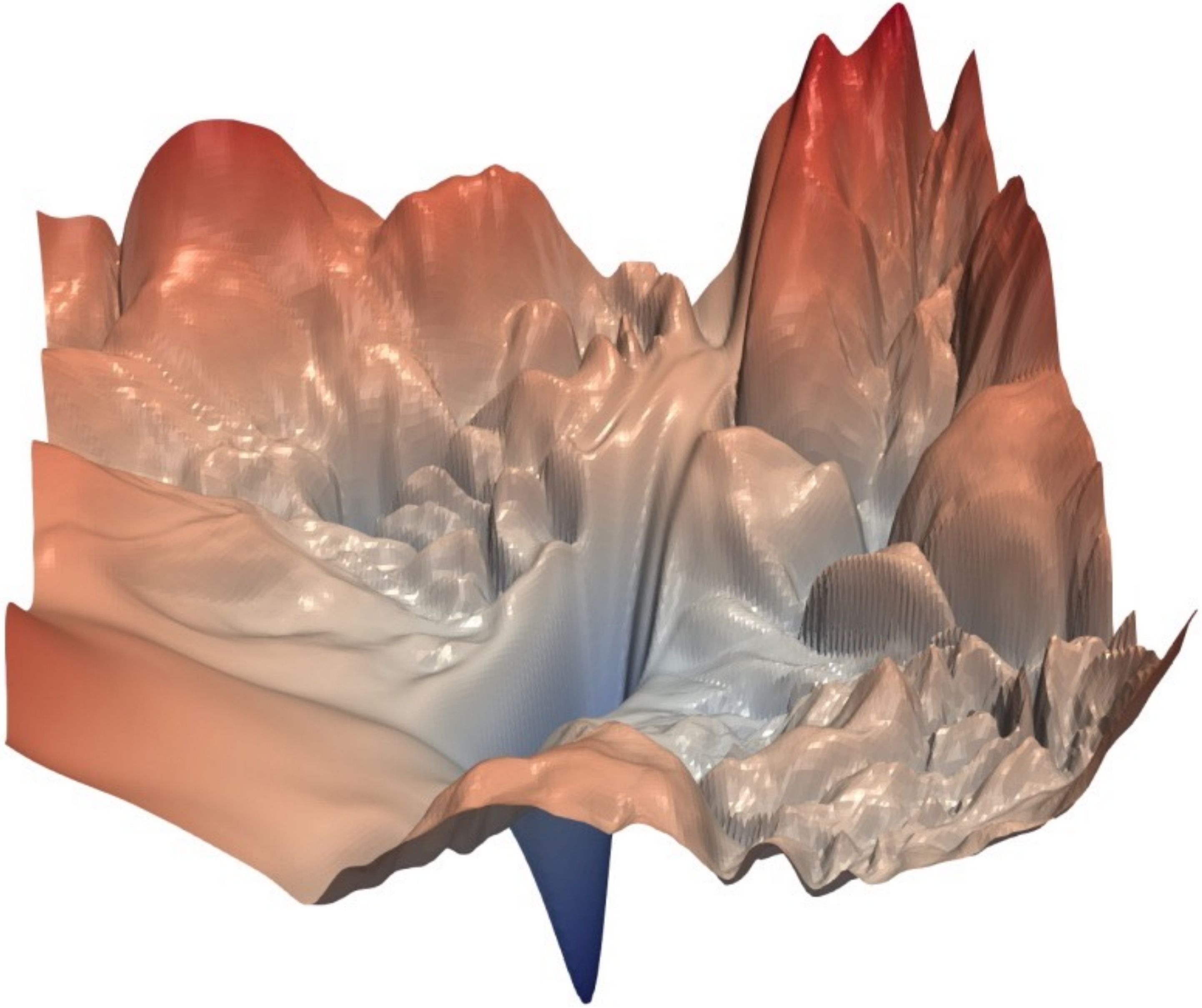
$$\min_{\Delta\theta} \boxed{\nabla_\theta L(\theta)^T \Delta\theta} + \boxed{\lambda|\Delta\theta|^2}$$

$$\theta \leftarrow \theta - \left(\frac{2}{\lambda}\right) \nabla_\theta L(\theta)$$



Architecture-Naive Majorizations

- **Deep networks have complicated optimization landscapes**
- **Using a single isotropic majorization can be very inefficient!**

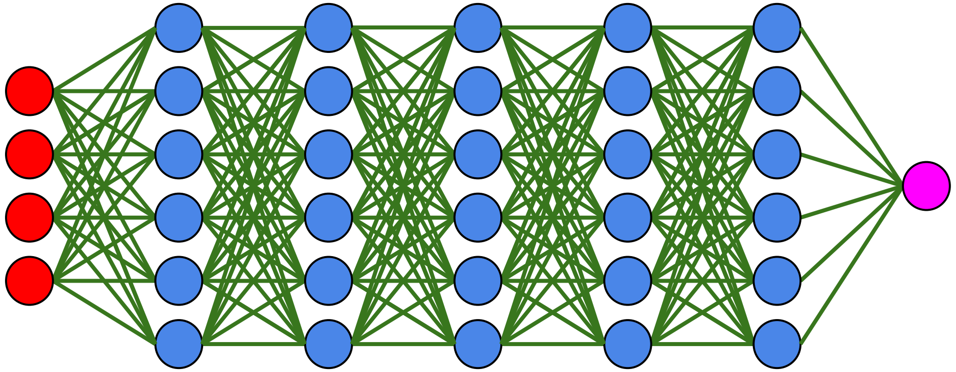


https://arxiv.org/abs/1712.09913

How to define a majorization that exploits NN structure?

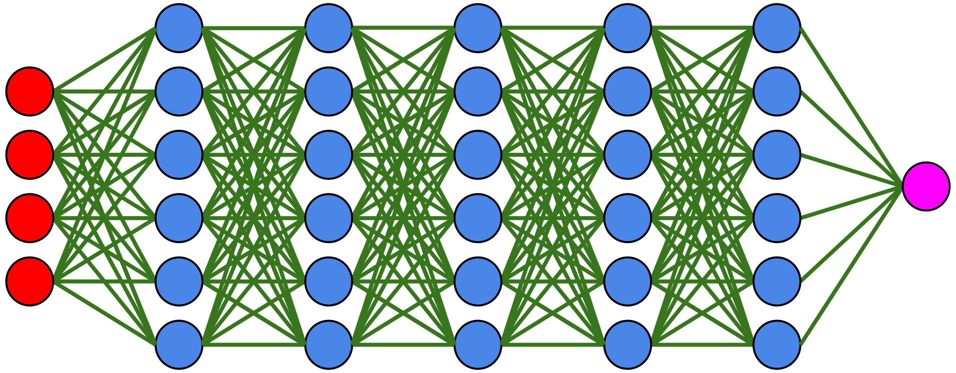
Jeremy Bernstein

Kevin Huang



- **(First) Key Idea:** per-layer perturbation analysis
- Perturb entire layer's parameters => perturbation of final output

Thought Experiment



- Vary only n-th layer: $\theta_n + \Delta\theta_n$
- What is the Lipchitz constant of entire function f?
- **Depends on parameters $\boldsymbol{\theta}$ of other layers!**
  - **If other parameters are larger => perturbation is larger!**

Architecture-Aware Perturbation Bounds

Lemmas 5.1 & 6.2 in https://arxiv.org/abs/2210.10101

Jeremy Bernstein

Kevin Huang

For L_2 loss:

$$L(\theta + \Delta\theta) \leq \underbrace{L^{(1)}}_{\text{1st order Taylor}} + \underbrace{\psi_\theta(\Delta\theta)}_{\text{Upper Bound of Rest}}$$

Architecture-Aware Perturbation Bounds

Lemmas 5.1 & 6.2 in https://arxiv.org/abs/2210.10101


Jeremy Bernstein


Kevin Huang

Overall perturbation sensitivity

$$\Delta f(\theta) := f(\theta + \Delta\theta) - f(\theta)$$

Breakdown of linear approx

1st order Taylor

For L_2 loss:

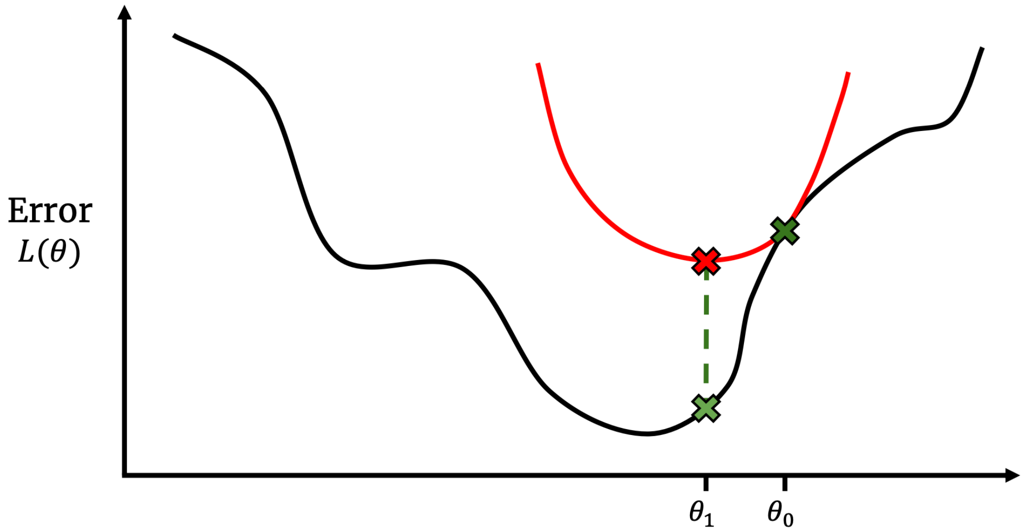
$$L(\theta + \Delta\theta) \leq L^{(1)} + O(|\Delta f(\theta)|_2) + O(|\Delta f(\theta) - \Delta\theta^T \nabla_\theta f(\theta)|_2)$$

$$\boxed{\leq C_1 \left[\prod_{n=1}^{N}\left(1 - \frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right) - 1\right]}$$

$$\boxed{\leq C_2 \left[\prod_{n=1}^{N}\left(1 - \frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right) - 1 - \sum_{n=1}^{N}\frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right]}$$

“Deep Relative Trust”: $\dfrac{|\boldsymbol{\Delta\theta_n}|_*}{|\boldsymbol{\theta_n}|_*}$

Majorize-Minimize for Neural Networks



- Derive ***majorization*** of error
- Plug in ***architecture perturbation bound***
- ***Minimize*** to obtain optimization algorithm

Related: “Automatically Bounding the Taylor Remainder Series: Tighter Bounds and New Applications” (https://arxiv.org/abs/2212.11429)

Desiderata & Caveats

$$L(\theta + \Delta\theta) \leq L^{(1)} + O(|\Delta f(\theta)|_2) + O(|\Delta f(\theta) - \Delta\theta^T \nabla_\theta f(\theta)|_2)$$

$$\boxed{\leq C_1 \left[\prod_{n=1}^{N}\left(1 - \frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right) - 1\right]}$$

$$\boxed{\leq C_2 \left[\prod_{n=1}^{N}\left(1 - \frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right) - 1 - \sum_{n=1}^{N}\frac{|\Delta\theta_n|_*}{|\theta_n|_*}\right]}$$

- “Clean” bound only for deep linear networks
  - Formula more complicated with non-linearities
  - First ever analysis even for deep linear networks
- Majorization has no (known) closed-form solution
  - Solving for the optimal $\Delta\theta$ is itself an optimization problem

First Result

Theorem 6.2 in https://arxiv.org/abs/2210.10101

Jeremy Bernstein

Kevin Huang

- **Key restriction:** enforce relative update of each layer to be the same
  - Suboptimal but closed-form update rule

Parameters of Layer n

Per-Layer Scaling

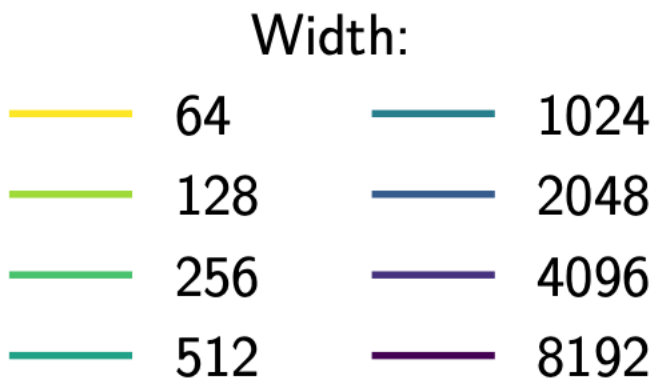
$$\theta_n \leftarrow \theta_n - \eta \frac{1}{N} \frac{|\theta_n|_F}{\sqrt{\min(\mathrm{dim}_n, \mathrm{dim}_{n-1})}} \cdot \frac{\nabla_{\theta_n} L(\theta)}{|\nabla_{\theta_n} L(\theta)|_F}$$

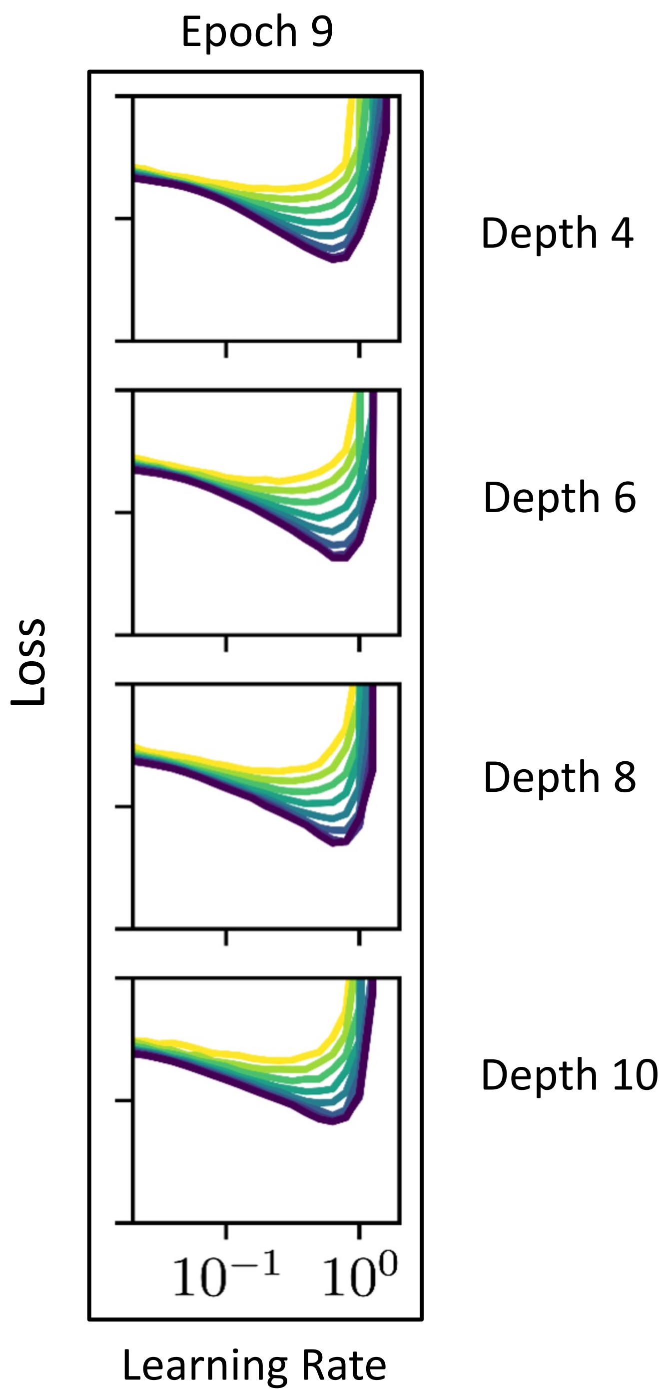
Normalized Gradient

Total Depth

dimensionality of hidden layer

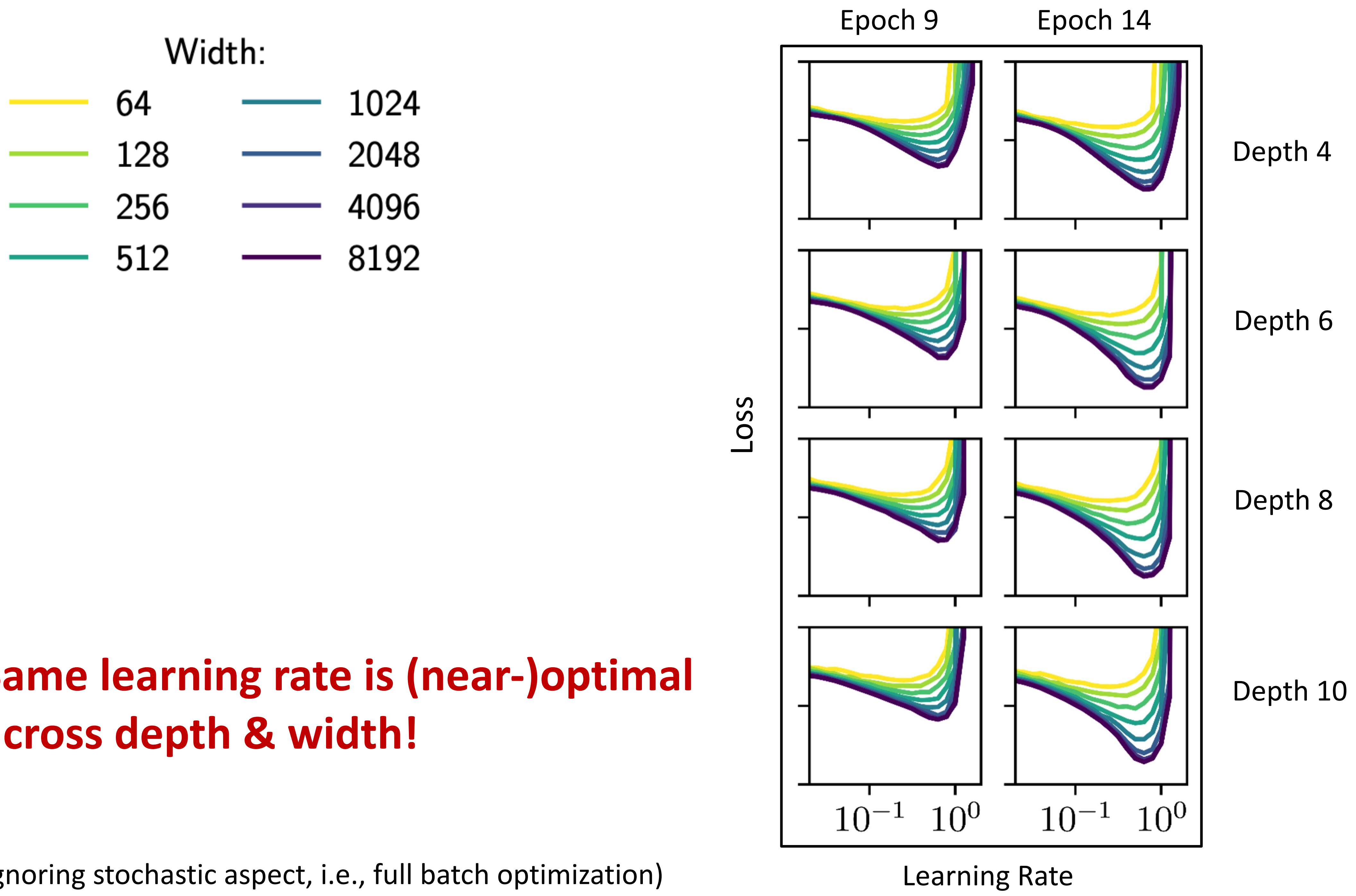
- **Learning rate η transfers to wider & deeper networks!**
  - (Related to mu-Parameterization by Greg Yang et al.)





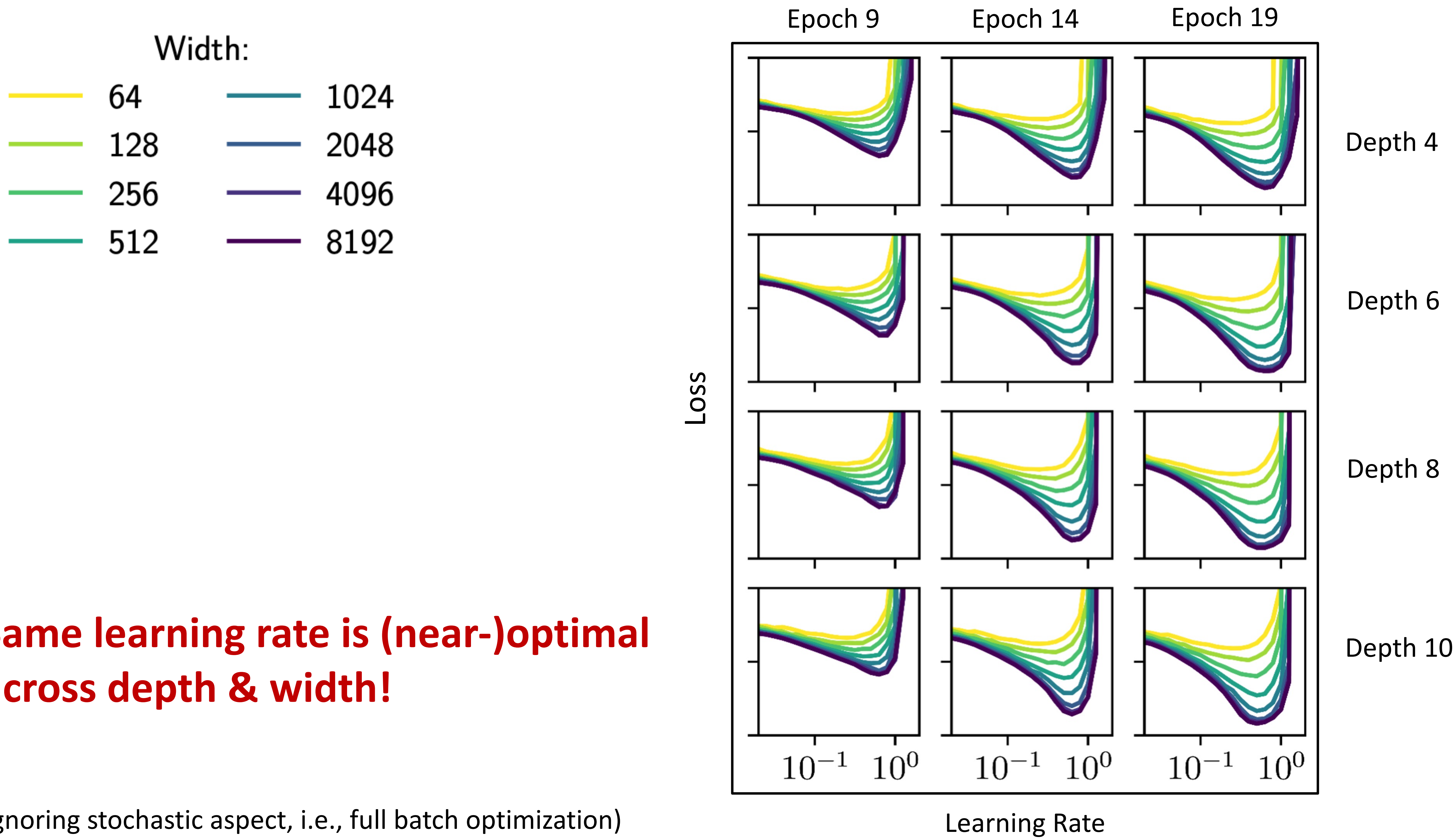
Same learning rate is (near-)optimal across depth & width!

(ignoring stochastic aspect, i.e., full batch optimization)



Same learning rate is (near-)optimal across depth & width!

(ignoring stochastic aspect, i.e., full batch optimization)

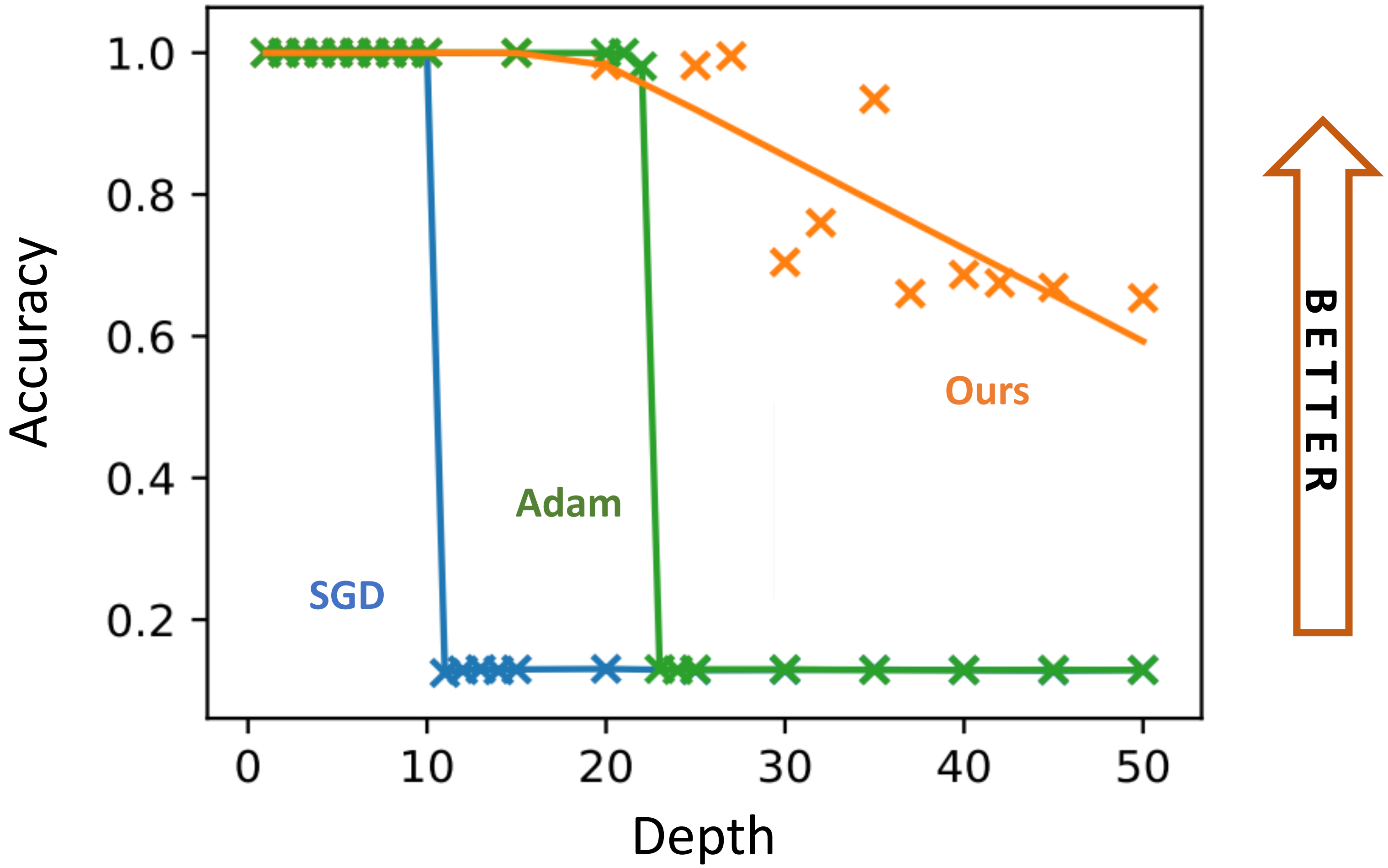


Same learning rate is (near-)optimal across depth & width!

(ignoring stochastic aspect, i.e., full batch optimization)

Case Study: Training Very Deep Networks

(Fully connected, no Skip Connections or Normalization Layers)



Training very deep networks is hard!
Practitioners use techniques like
skip connections & normalization layers.

https://arxiv.org/abs/2002.03432

Latest Result

Automatic Gradient Descent

Initialize Weights:

- **for** layer n in $\{1, \dots, N\}$:
  - $\theta_n \sim \mathrm{unif}\big(\mathrm{orthogonal}(\mathrm{dim_n}, \mathrm{dim_{n-1}})\big)$
  - $\theta_n \sim \theta_n \cdot \sqrt{\frac{\mathrm{dim_n}}{\mathrm{dim_{n-1}}}}$

$$\Rightarrow \quad |\theta_n| = \sqrt{\frac{\mathrm{dim_n}}{\mathrm{dim_{n-1}}}}$$

Update Weights:

- $G \leftarrow \frac{1}{N}\sum_{n=1}^{N} |\nabla_{\theta_n} L|_F \cdot \sqrt{\frac{\mathrm{dim_n}}{\mathrm{dim_{n-1}}}}$
- $\eta \leftarrow \log \frac{1+\sqrt{1+4G}}{2}$
- **for** layer n in $\{1, \dots, N\}$:
  - $\theta_n \leftarrow \theta_n - \frac{\eta}{N} \cdot \frac{\nabla_{\theta_n} L}{|\nabla_{\theta_n} L|_F} \cdot \sqrt{\frac{\mathrm{dim_n}}{\mathrm{dim_{n-1}}}}$

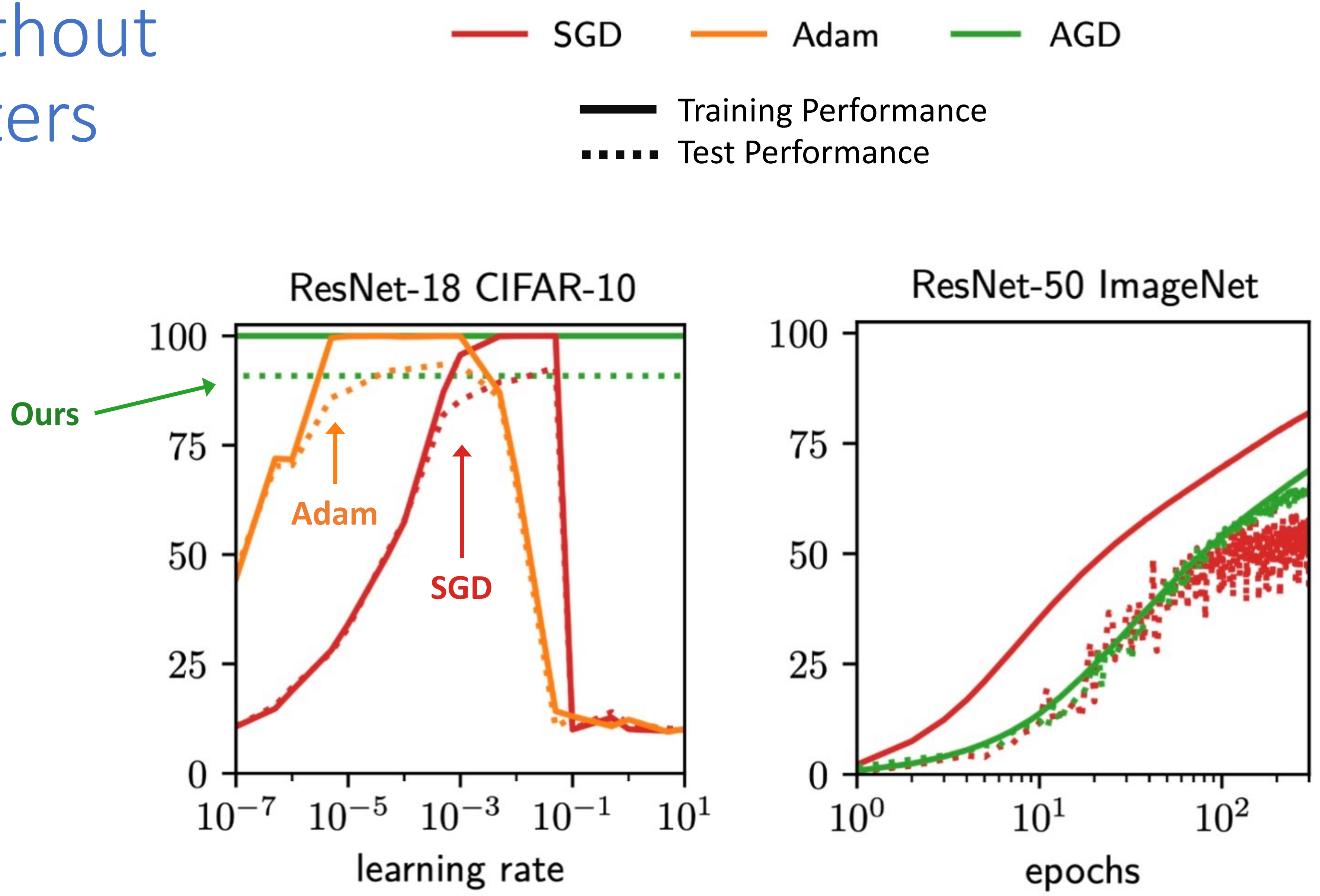
$$\Rightarrow \quad |\Delta\theta_n| = \frac{\eta}{N} \cdot \sqrt{\frac{\mathrm{dim_n}}{\mathrm{dim_{n-1}}}}$$

Jeremy Bernstein

Chris Mingard

https://arxiv.org/abs/2304.05187

AGD trains without hyperparameters

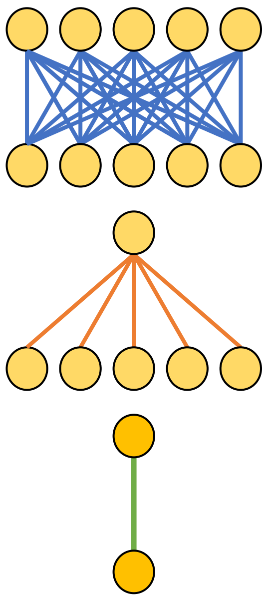


https://arxiv.org/abs/2304.05187

Different Ways to Design Optimizers

θ_n

1.2	- 0.5	2.3	0.4	1.3
- 0.4	2.1	- 0.8	0.7	1.5
1.1	0.5	- 2.4	0.3	1.0
0.4	- 2.1	- 0.2	0.1	0.5
1.4	- 0.2	- 1.1	0.1	1.4



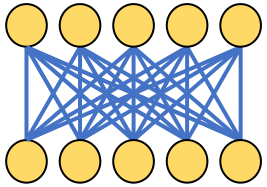
Per-layer

Per-neuron

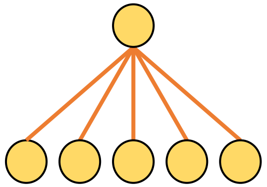
Per-synapse

Different Ways to Design Optimizers

- **Fromage:** Per-layer was our first foray
  - https://arxiv.org/abs/2002.03432




- **Nero:** Per-neuron
  - Also constrain per-neuron weight norm
  - Connections to batch-norm
  - Connections to generalization
  - https://arxiv.org/abs/2102.07227

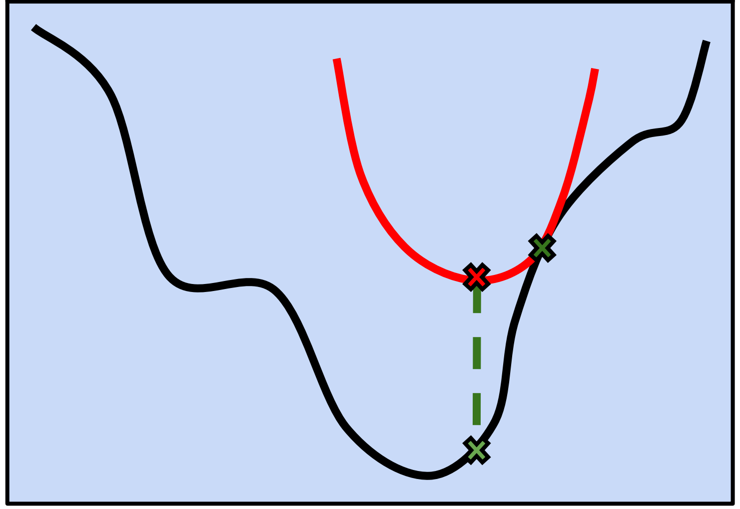


- **Madam:** Per-synapse
  - Also sign-constrain weights
  - Leads to multiplicative update rule
  - Connections to biological synapses
  - https://arxiv.org/abs/2006.14560



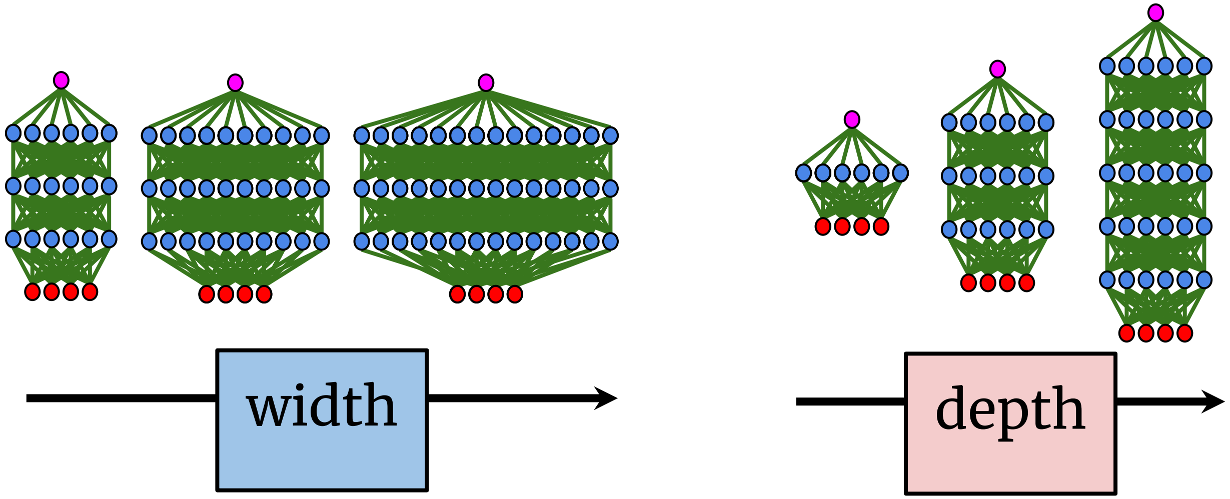
Summary: Architecture-Aware Perturbation Bounds

Majorise-minimise

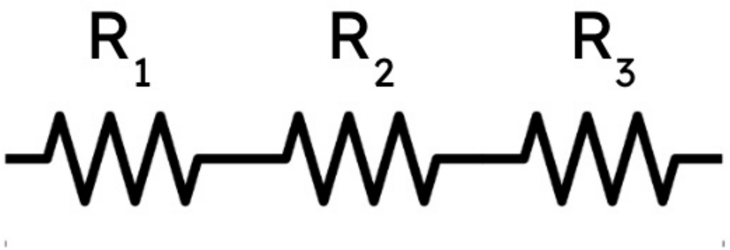


Architectural perturbation bound

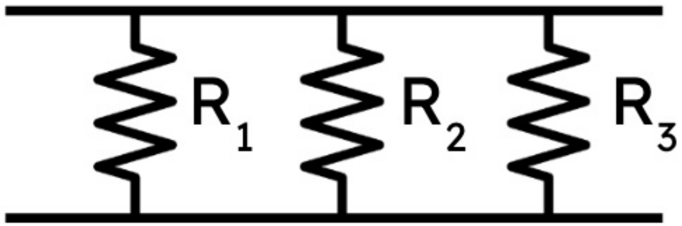
$$\|\Delta f(x)\| \leq C \cdot \left[\prod_l \left(1 + \frac{\|\Delta W_l\|}{\|W_l\|} \right) - 1 \right]$$



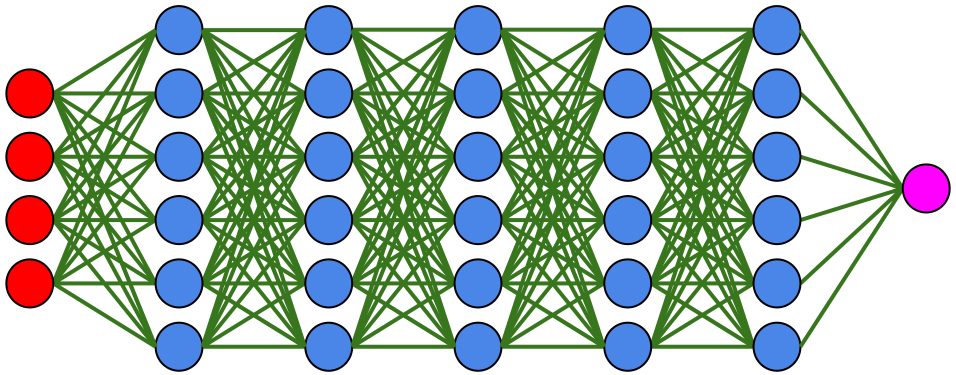
Towards A Practical Theory of Deep Learning Optimization



$$\boxed{R = R_1 + R_2 + R_3}$$



$$\boxed{1/R = 1/R_1 + 1/R_2 + 1/R_3}$$



Theory of Composite Functions?

- Learning Rate
- Learning Rate Decay
- Momentum
- Gradient Averaging
- Warm-up Iterations
- …

Optimisation & Generalisation in Networks of Neurons

Thesis by

Jeremy Bernstein

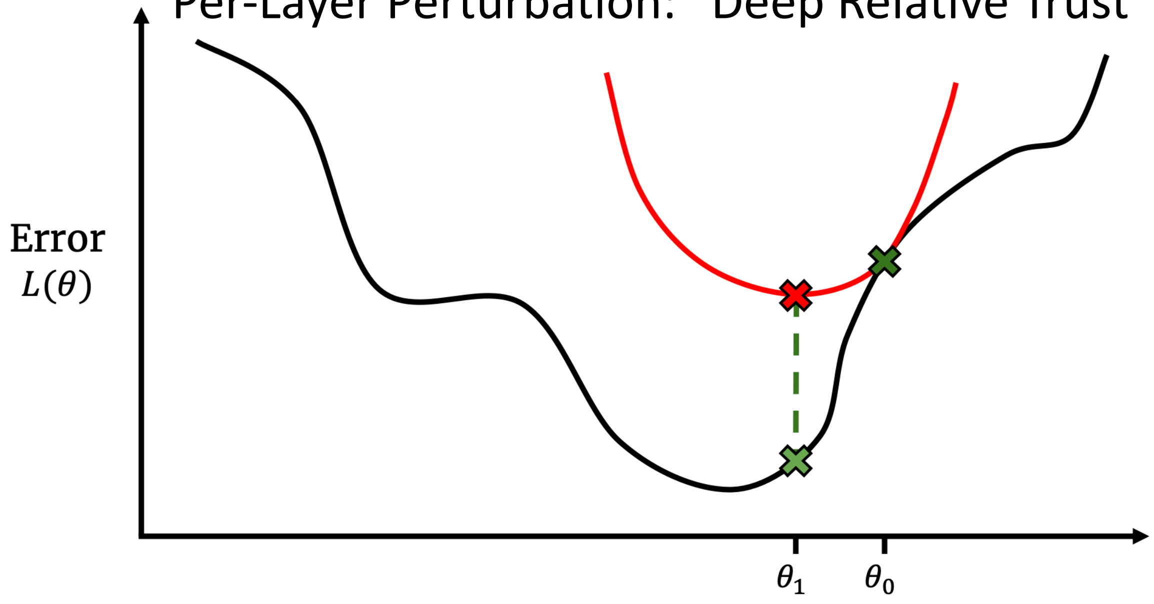
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Jeremy Bernstein

https://arxiv.org/abs/2210.10101

Majorize-Minimize Framework

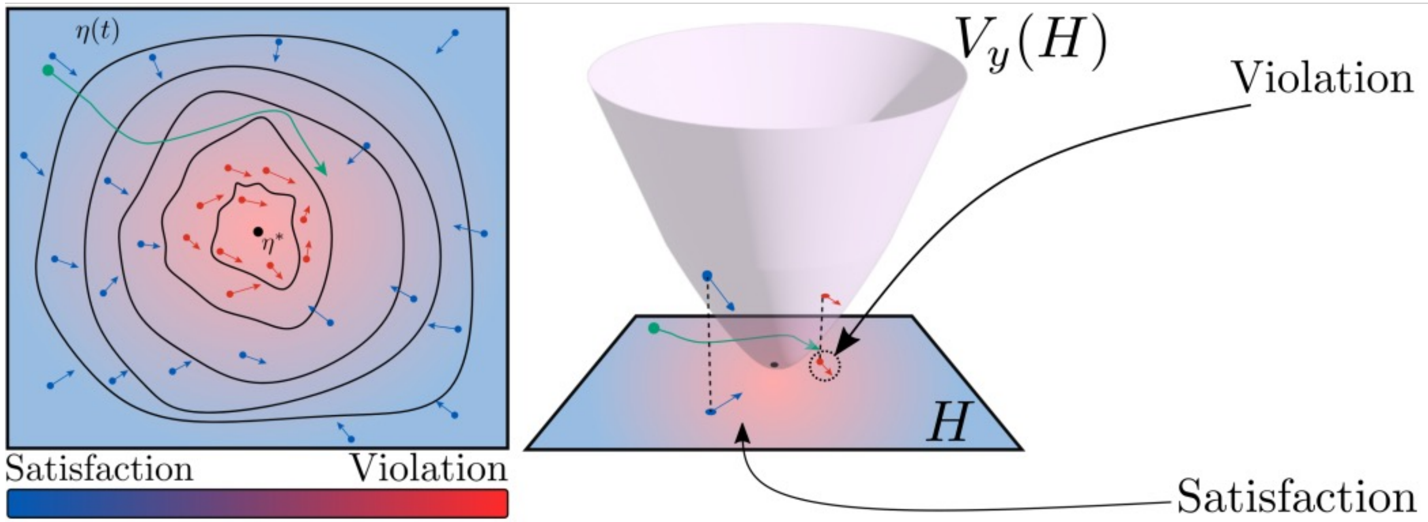
Per-Layer Perturbation: “Deep Relative Trust”



How to exploit the structure of NNs to develop a more nuanced theory?

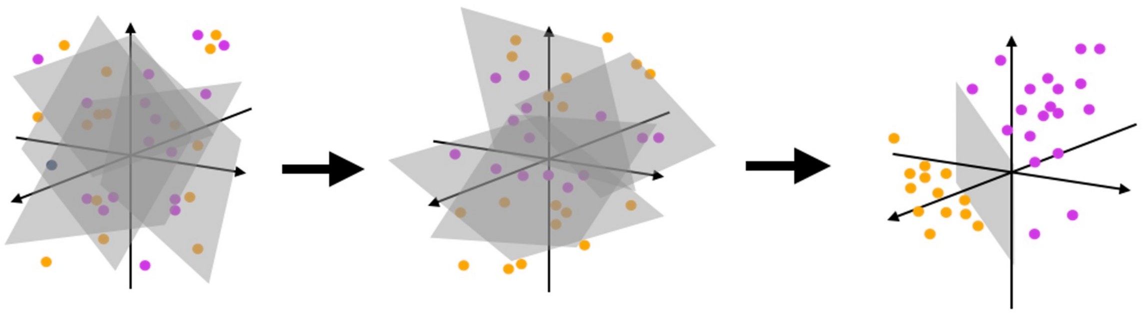
Control-Theoretic Shaping of Neural ODEs

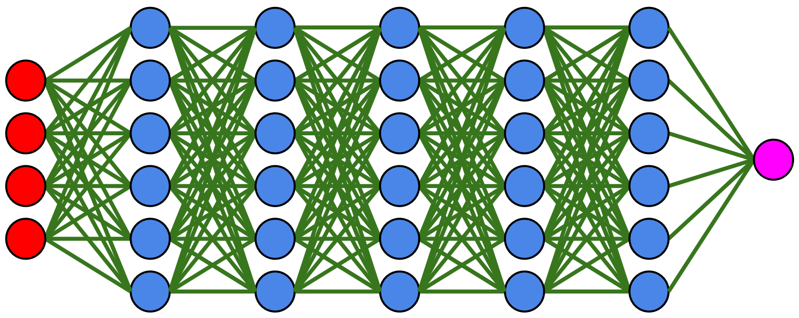
“Lyapunov Loss”



Recall Idea #2: Control Dynamics of Hidden Layers

We want each layer to push representation towards good answer

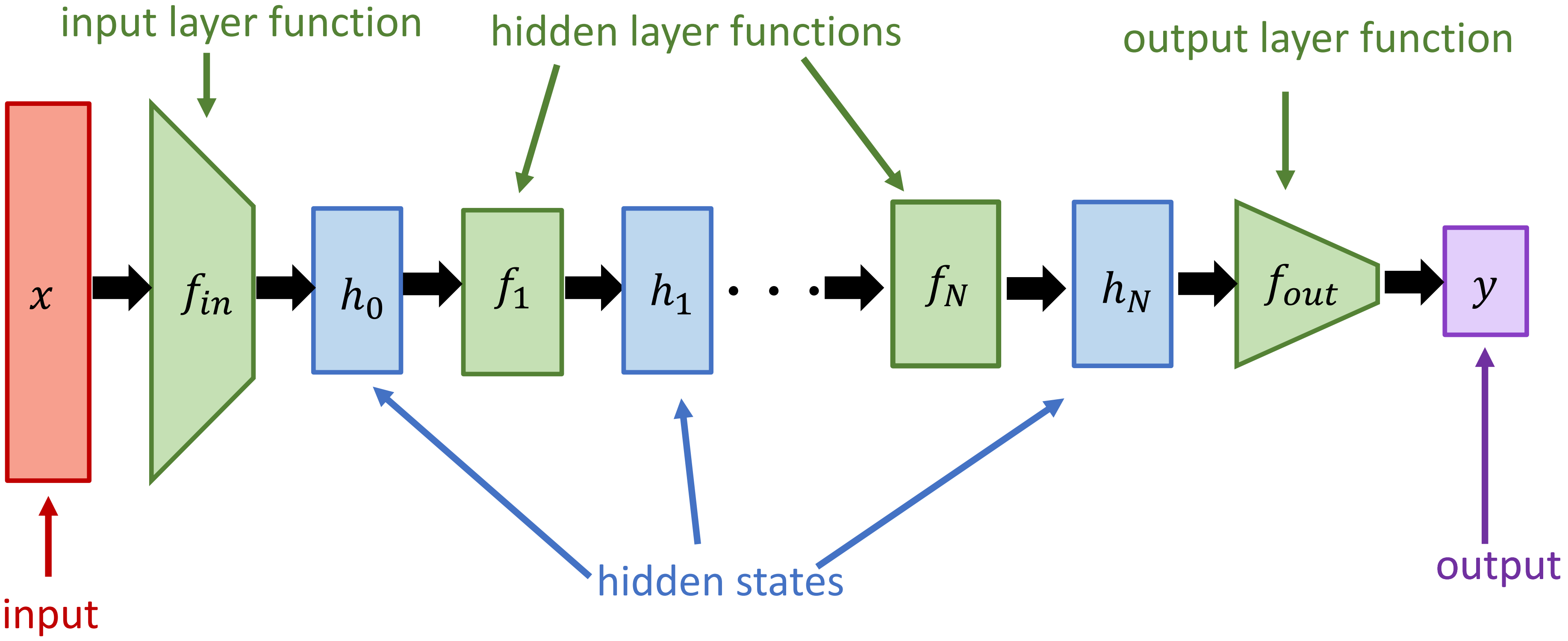




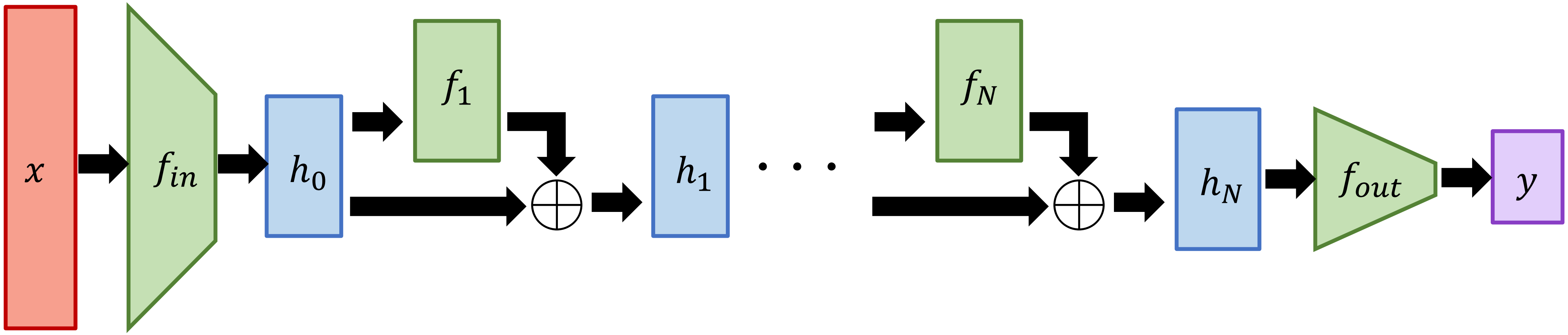
Goal: control sequence of hidden layers $h_1, \dots, h_N$

- quickly and robustly converge to low loss

Warm Up: Deep Networks



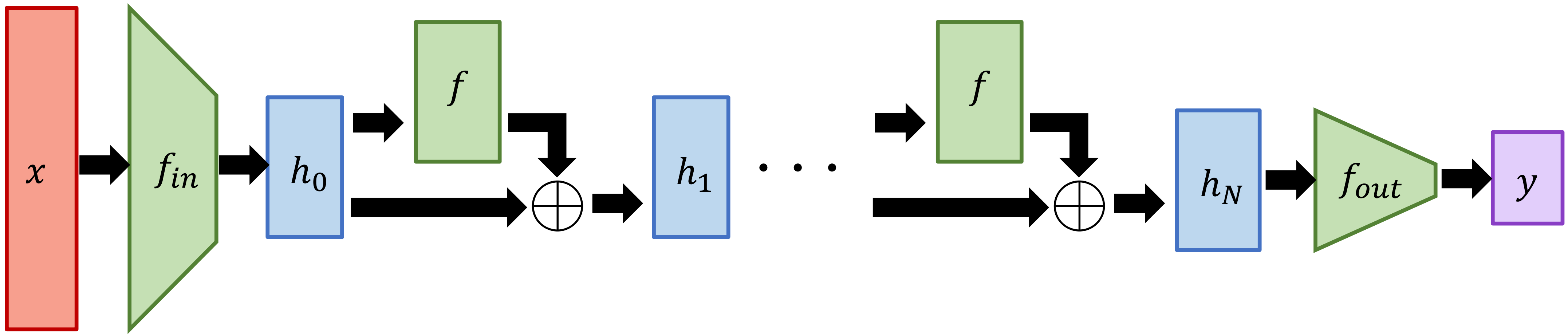
Warm Up: ResNets



Functional Form:

$$h_0 = f_{in}(x)$$
$$h_n = h_{n-1} + f_n(h_{n-1})$$
$$y = f_{out}(h_N)$$

ResNets => Continuous-in-Depth => Neural ODEs



"Euler Integration Step Size"

$$h_n = h_{n-1} + \delta f(h_{n-1}) \xrightarrow{\lim_{\delta \to 0}} \frac{\partial h}{\partial t} = f(h)$$

Neural Ordinary Differential Equations (NODEs)

https://arxiv.org/abs/1806.07366

$$h_0 = f_{in}(x)$$

Input Layer

$$\frac{\partial h}{\partial t} = f(h, x)$$

Continuum of hidden layers
“Dynamics”

$$y(t) = f_{out}(h(t))$$

Output Layer

WLOG: $t \in [0,1]$

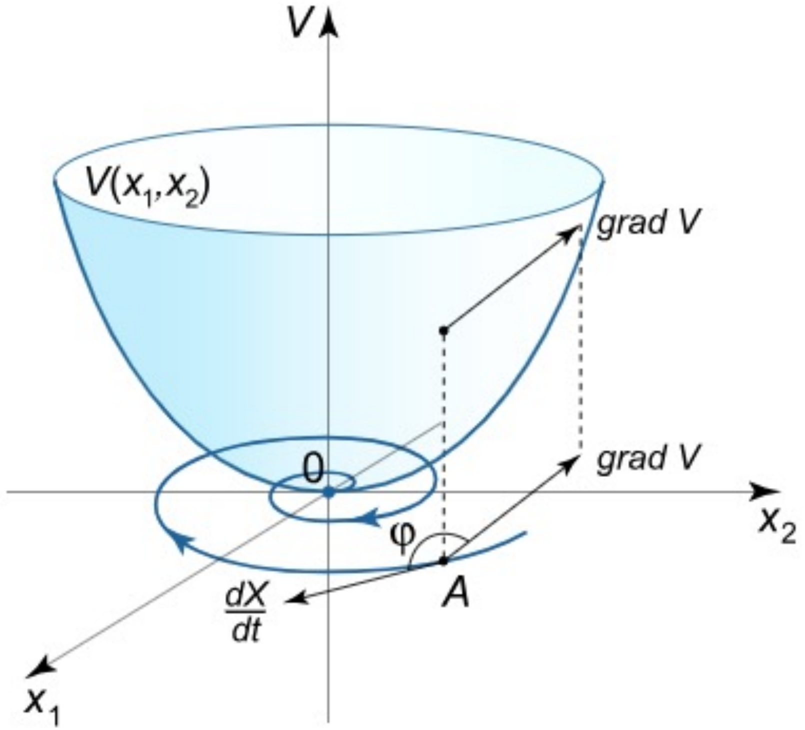
Comments:

- Forward pass requires ODE solve
- Can evaluate output at any time
  - E.g., y(0.5)
- Includes continuous normalizing flows & other generative models

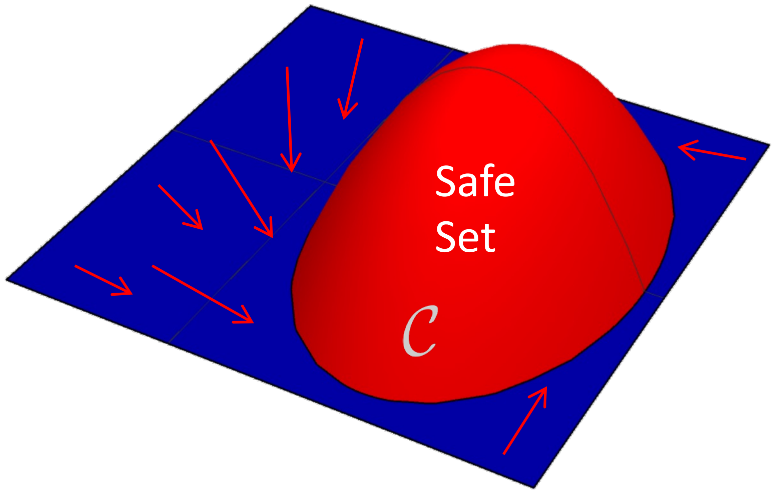
(“augmented” NODE since f depends on x -- https://arxiv.org/abs/1904.01681)

Using Control Theory to Shape Learning

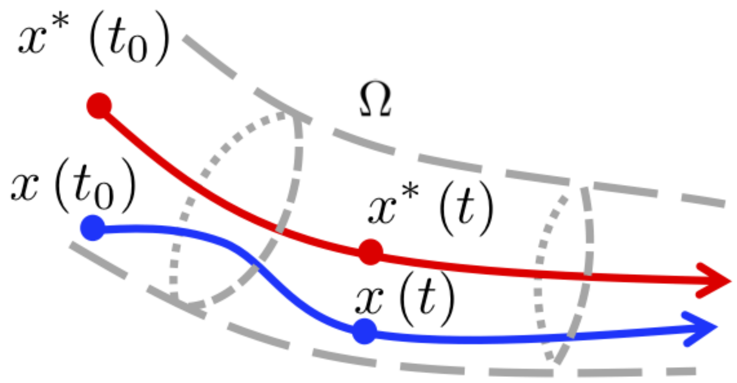
- Control can shape dynamical systems (most commonly ODEs)



Stability



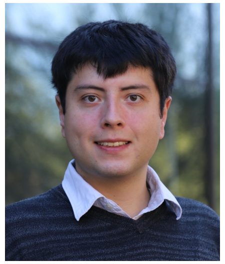
Safety



Adversarial Robustness

Images by Aaron Ames & Brett Lopez

Why Shape Dynamics of NODEs?

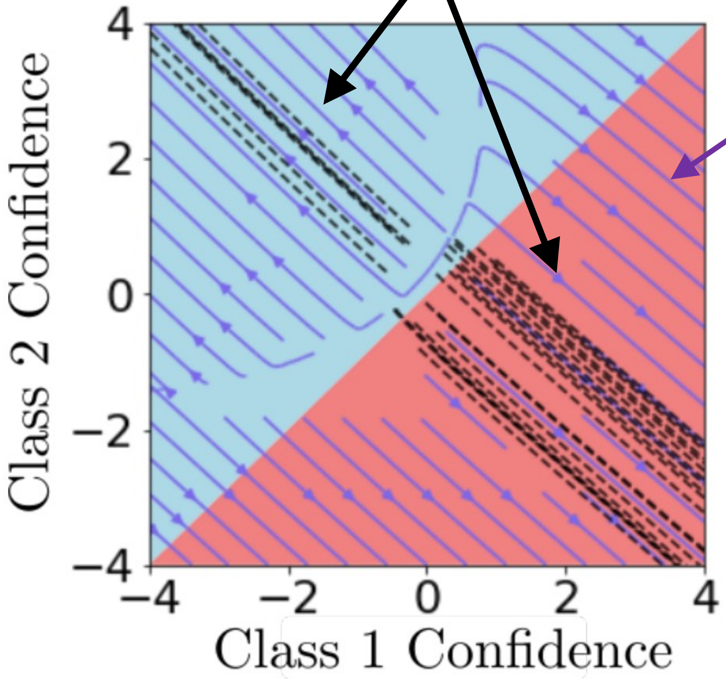


Ivan
Jimenez Rodriguez

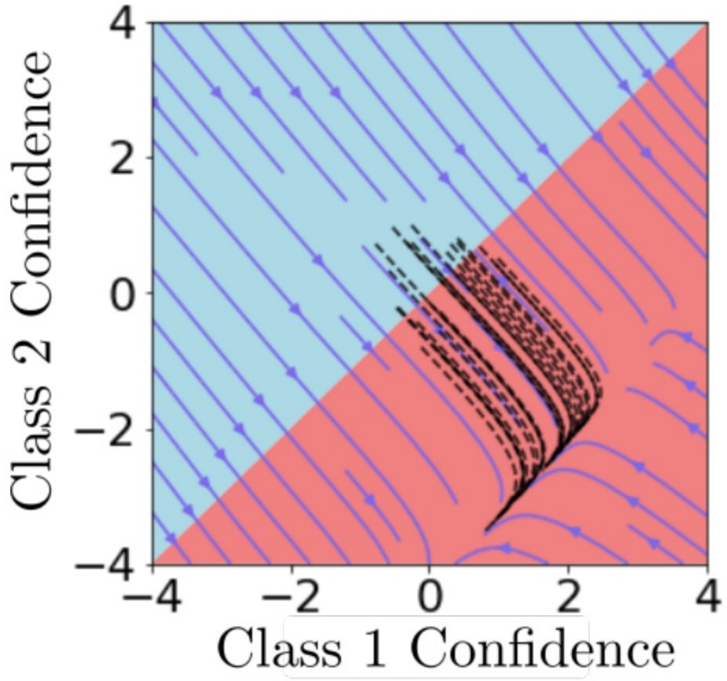
2D state space, Red Class is correct
Showing inference trajectories under perturbations

Trajectories

Dynamics:



Unstable NODE
Standard BackProp



Our Approach
Lyapunov Loss

What dynamics can we shape?

$$h_0 = f_{in}(x)$$

"Dynamics"

$$\frac{\partial h}{\partial t} = f(h, x)$$

$$y(t) = f_{out}(h(t))$$

WLOG: $t \in [0,1]$

- Full state space: $h(t)$?
  - (too unwieldy)

"Potential function"

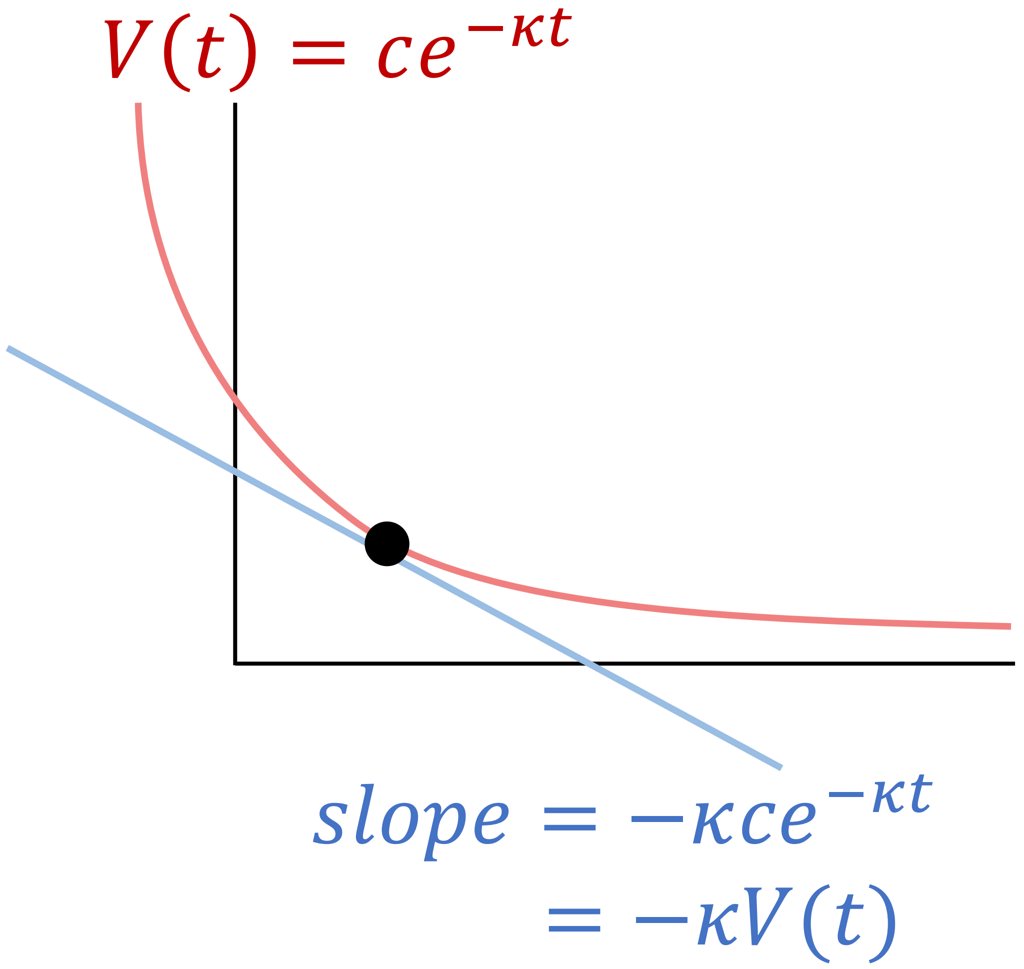
- **Projection**: $V(h(t)) \equiv L\big(y(t)\big)$
  - **Dynamics of the training loss**
  - 1-D projection of state space

Exponential Stability

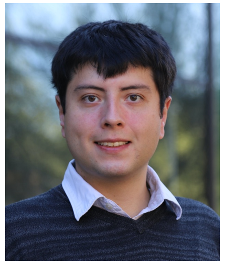
Goal: $V(h(t)) \leq V(h(0))e^{-\kappa t}$

Key invariant: $\frac{\partial V}{\partial h}^T f(h) \leq -\kappa V(h)$

Benefits: Fast convergence & Robustness

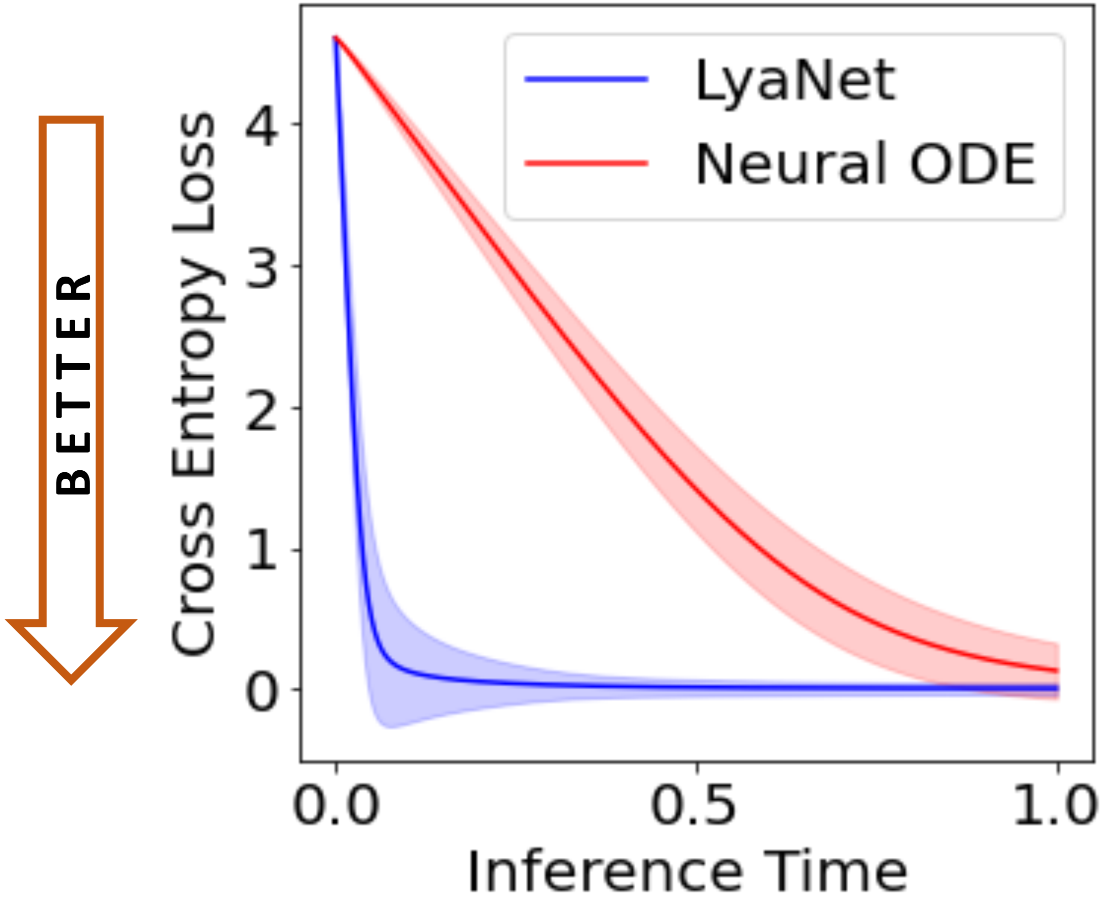


Exponential Convergence in Action



Ivan
Jimenez Rodriguez

CIFAR-10

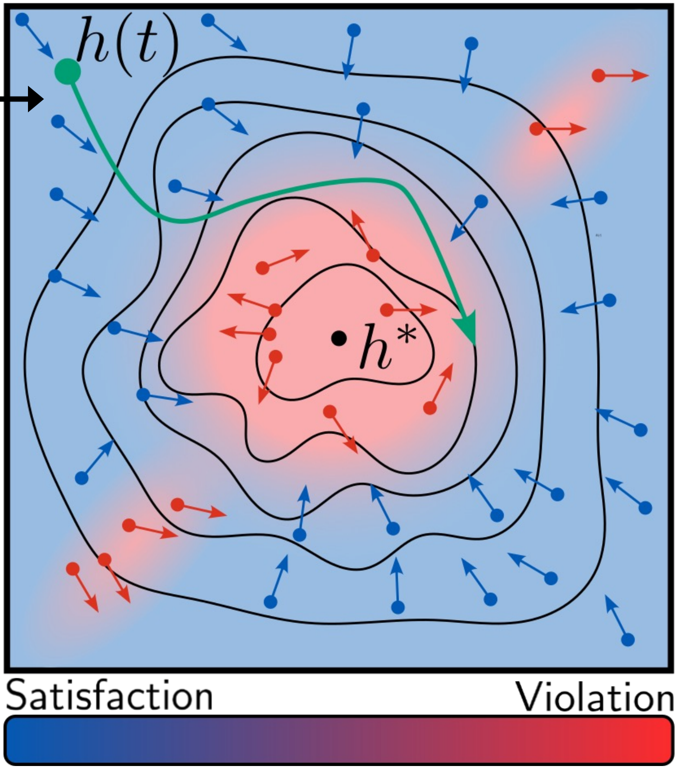


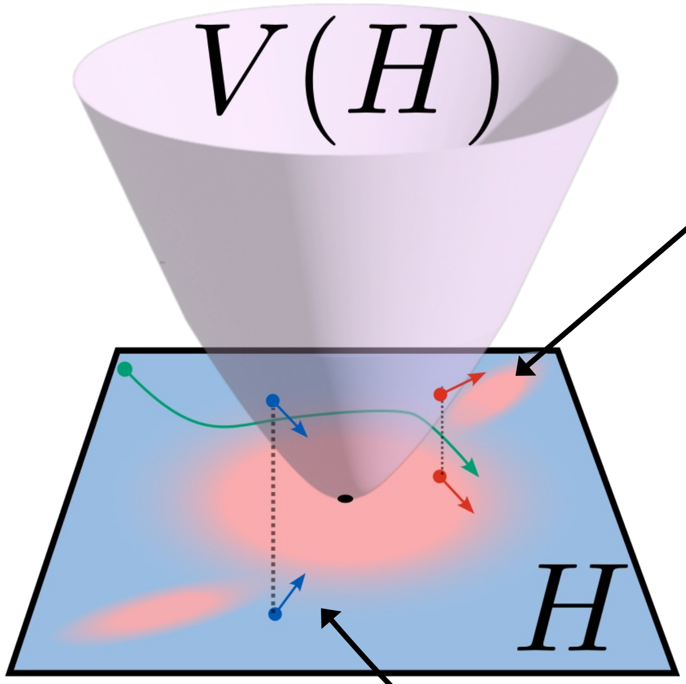
LyaNet is trained to optimize for exponential convergence

https://arxiv.org/abs/2202.02526

Measuring Progress via Contraction Condition

One trajectory





Violation:

$$\frac{\partial V^T}{\partial h} f(h) > -\kappa V(h)$$

Satisfaction:

$$\frac{\partial V^T}{\partial h} f(h) \leq -\kappa V(h)$$

Contraction Satisfied Everywhere => Exponential Stability!

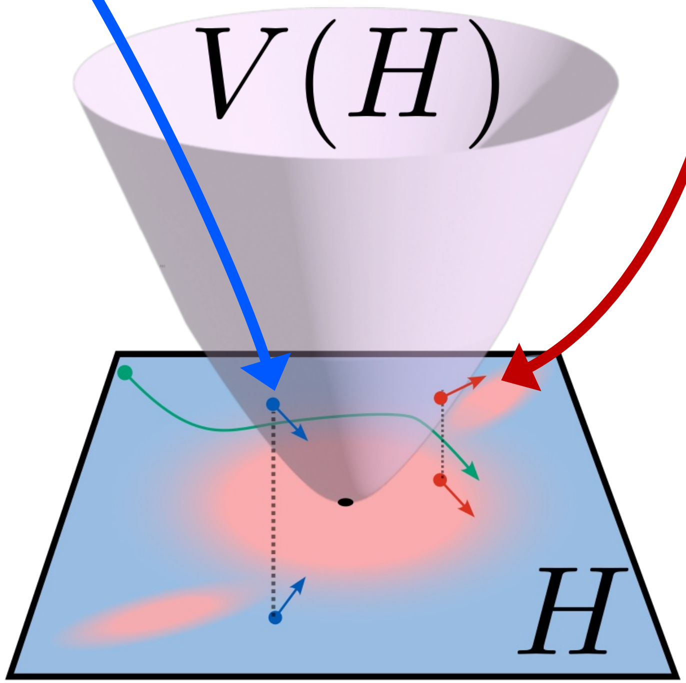
Lyapunov Loss

Satisfaction:

$$\frac{\partial V}{\partial h}^T f(h) \leq -\kappa V(h)$$

Violation:

$$\frac{\partial V}{\partial h}^T f(h) > -\kappa V(h)$$



- Point-wise Lyapunov Loss

$$L_V(x,y,h) \equiv \max\left\{0, \underbrace{\frac{\partial V_y}{\partial h}^T f(h) + \kappa V_y(h)}_{\text{Contraction condition violation}}\right\}$$

- Lyapunov Loss:

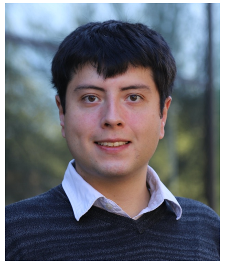
$$L_V(\theta) \equiv \boldsymbol{E}_{x,y}\left[\int_0^1 L_V\big(x,y,h(t)\big)dt\right]$$

Achieving zero Lyapunov Loss (almost) everywhere implies exponential stability!

https://arxiv.org/abs/2202.02526

LyaNet

A Lyapunov Framework for Training Neural ODEs



Ivan
Jimenez Rodriguez

1. Interpret training loss as potential function: $V(h(t)) \equiv L\big(y(t)\big)$

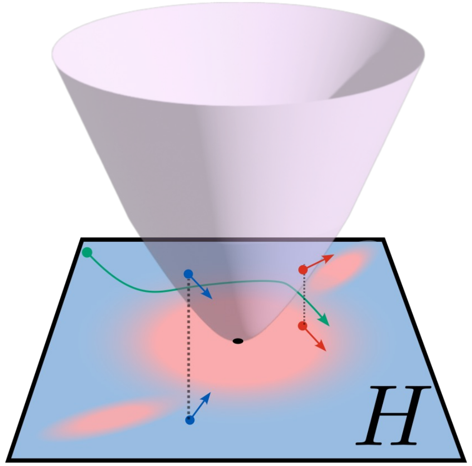
2. Instantiate (point-wise) Lyapunov Loss:

$$L_V(x, y, h) \equiv \max\left\{0, \frac{\partial V_y}{\partial h}^T f(h) - \kappa V_y(h)\right\}$$

3. Optimize Lyapunov Loss everywhere

https://arxiv.org/abs/2202.02526

Optimization Considerations



$$L_V(\theta) \equiv \boldsymbol{E}_{x,y}\left[\int_0^1 L_V\big(x, y, h(t)\big)dt\right]$$

Lyapunov Loss

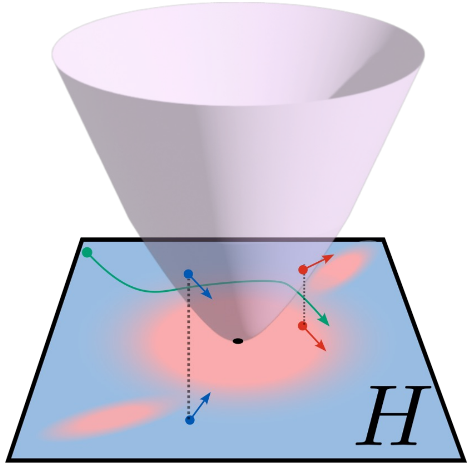
- Evaluating integral exactly is hard
- Approximate by sampling (simplest is Monte Carlo)
  - Sample (x,y,h) uniformly at random
  - Backprop on point-wise Lyapunov Loss

$$L_V(x, y, h) \equiv \max\left\{0, \frac{\partial V_y}{\partial h}^T f(h) - \kappa V_y(h)\right\}$$

Point-wise Lyapunov Loss

https://arxiv.org/abs/2202.02526

Benefits of Sampling



- Avoids expensive ODE solve

- Goal is to minimize Lyapunov Loss everywhere

$$L_V(\theta) \equiv \boldsymbol{E}_{x,y}\left[\int_0^1 L_V\big(x, y, h(t)\big)dt\right]$$

Achieving $L_V(\theta) = 0$ under uniform measure implies $L_V(\theta) = 0$ in original measure

- Similar idea used in Score-Based Generative Models & Moser Flows

https://arxiv.org/abs/2202.02526

Connection to Control Theory

https://ieeexplore.ieee.org/document/6709752

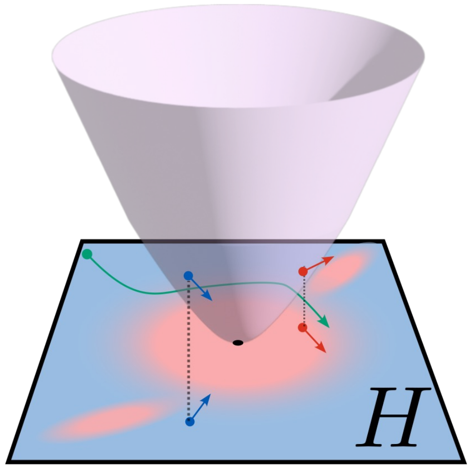
- V is an ***Exponentially-Stabilizing Control Lyapunov Function (ES-CLF)*** for a controllable ODE if for all states $h \in H$:

$$\min_{\theta} \left[\frac{\partial V}{\partial h}^T f(h;\theta) + \kappa V(h) \right] \leq 0$$

"controller"

Equivalent to Lyapunov Loss

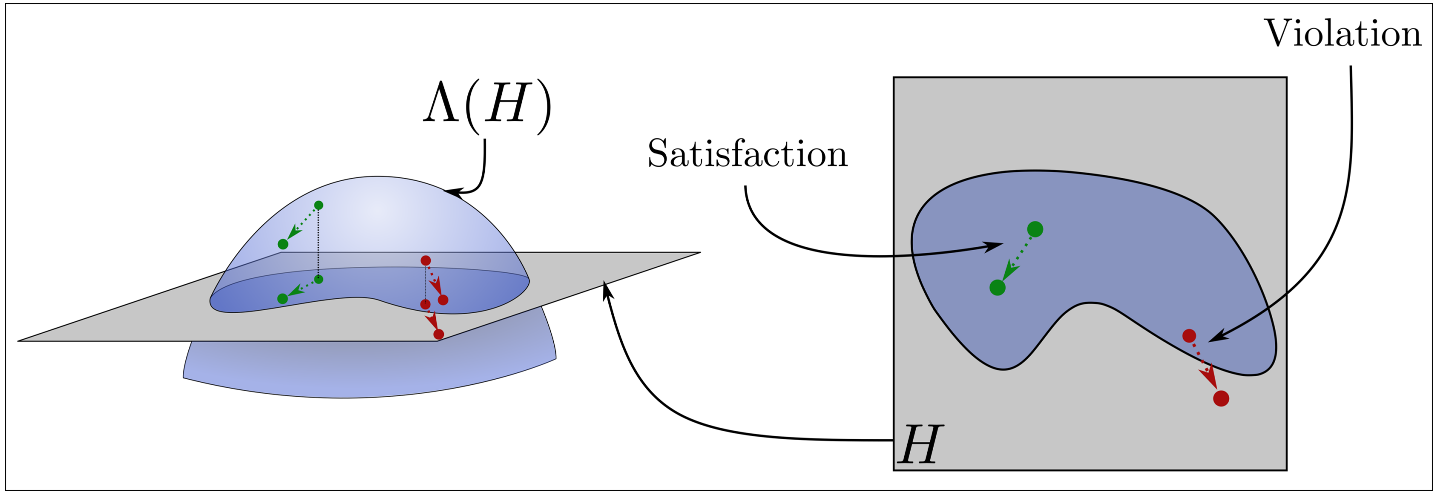
- Can we find a controller θ that makes V an ES-CLF?
  - In control, f has uncontrolled dynamics and θ is low-dim
- Connection between controllability & learnability
  - Can we find a θ that achieves zero Lyapunov Loss?
  - For NODEs, f is fully controlled and over-parameterized



Comments & Extensions

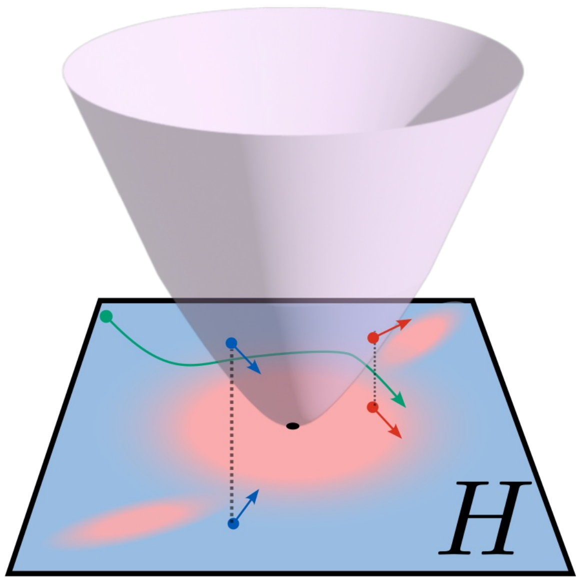
- Stabilize to sets rather than points
  - E.g., under adversarial perturbations, can still stabilize to a region of low loss
  - Sets need not be convex
- Combinations of conditions (multiple invariances)
- Other invariances:

Forward-Invariance
(never leaves “safe” set)



Motivating Application: Continuous Control

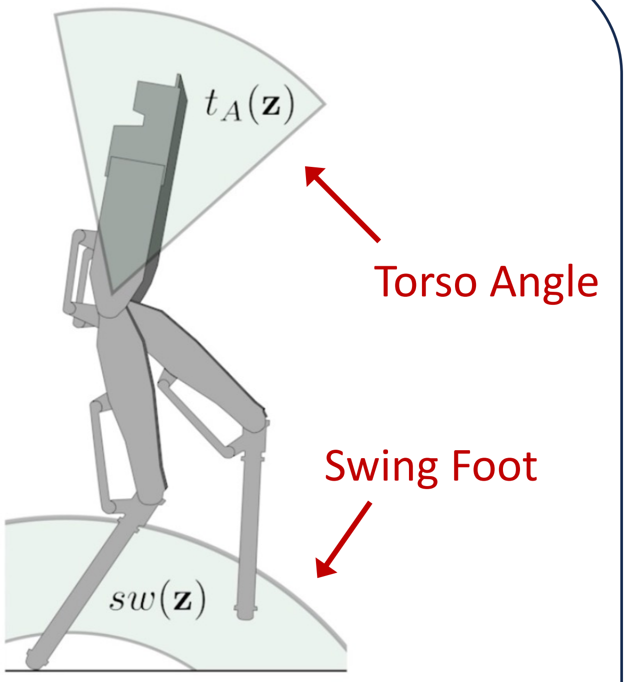
Note: Dynamics of Control System included in Neural ODE



$$\min_{\theta}\left[\frac{\partial V^T}{\partial h} f(h;\theta) + \kappa V(h)\right] \leq 0$$

Sample States
Compile Loss

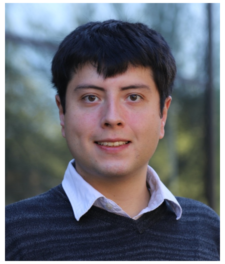
Deploy Controller



Stability/Safety Specification

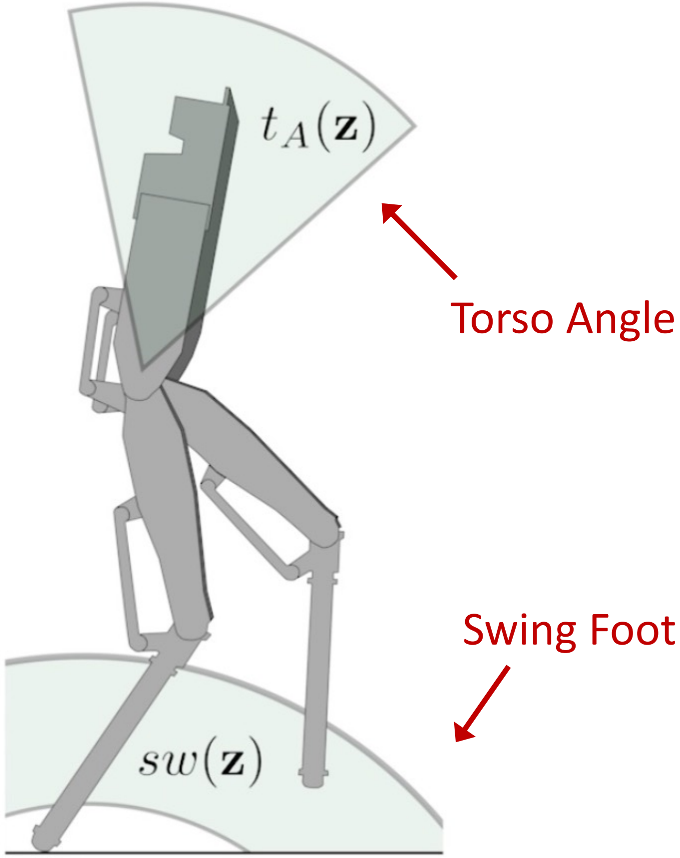
Neural Gaits

Learn policy to satisfy composition of continuous-time conditions
Implies indefinite walking (forward-invariance)

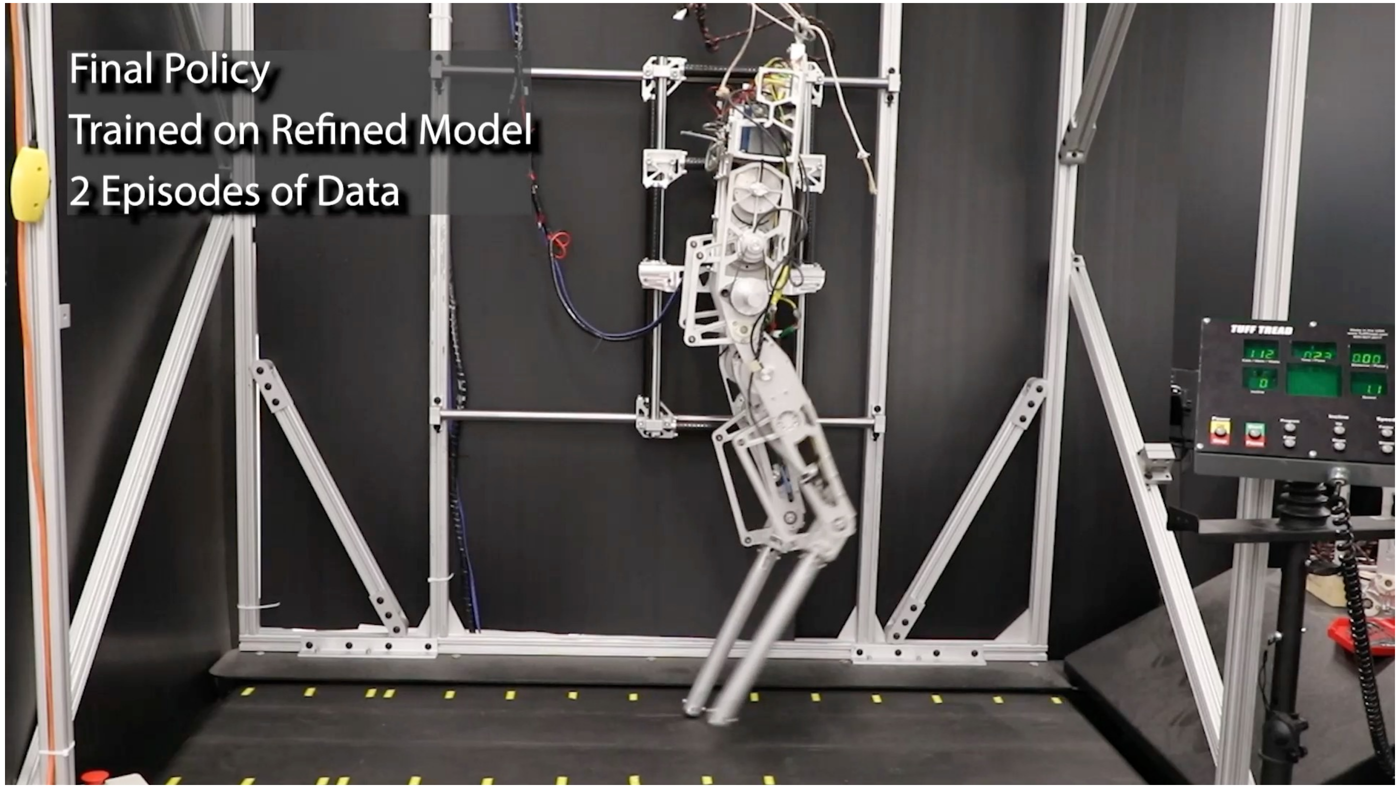


Ivan
Jimenez Rodriguez

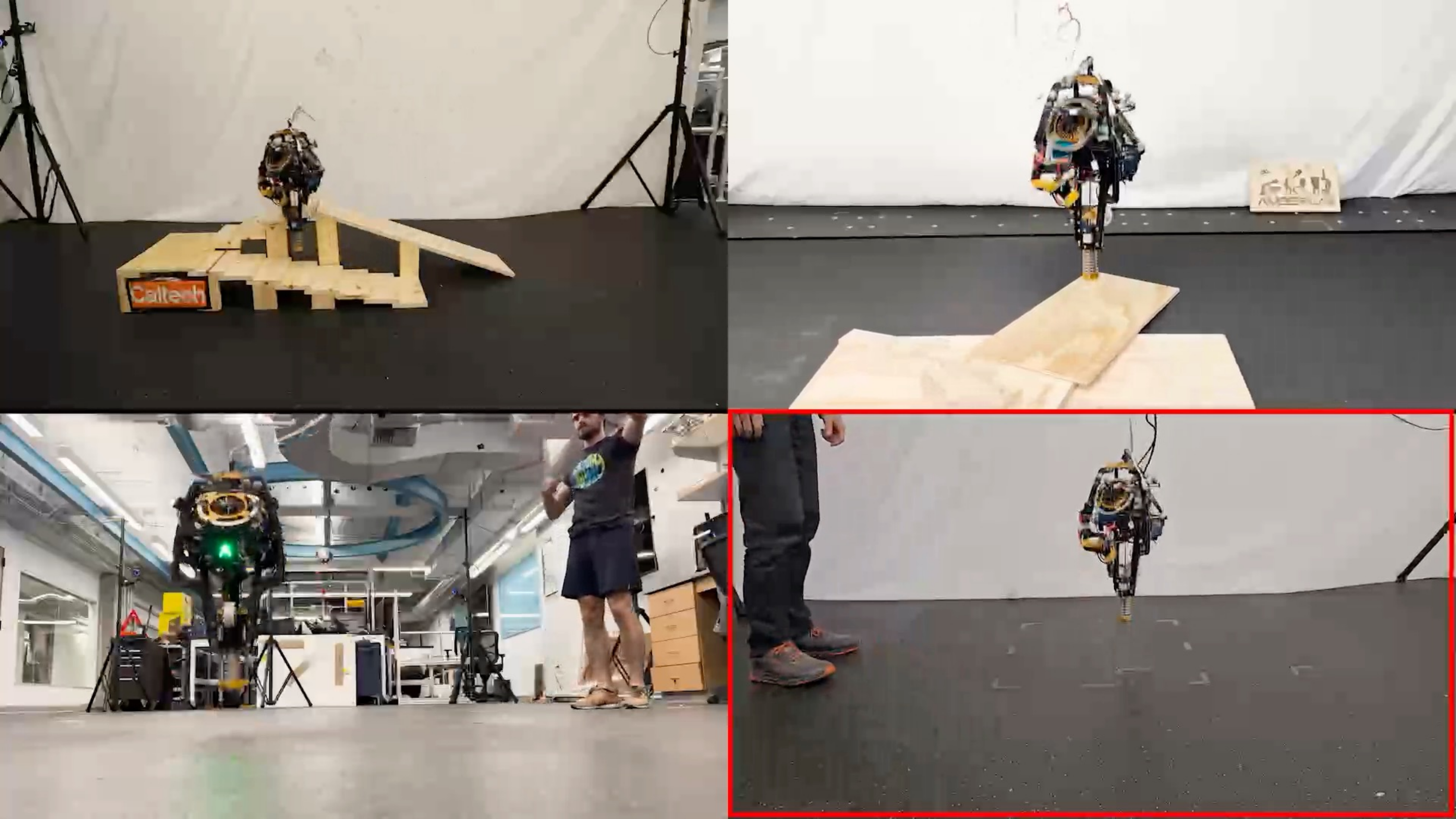
Noel
Csomay-Shanklin



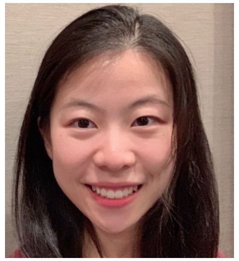
Example Barriers

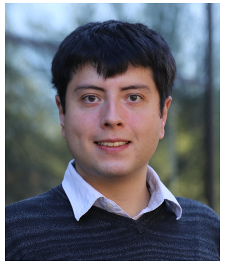


https://arxiv.org/abs/2204.08120


Caltech

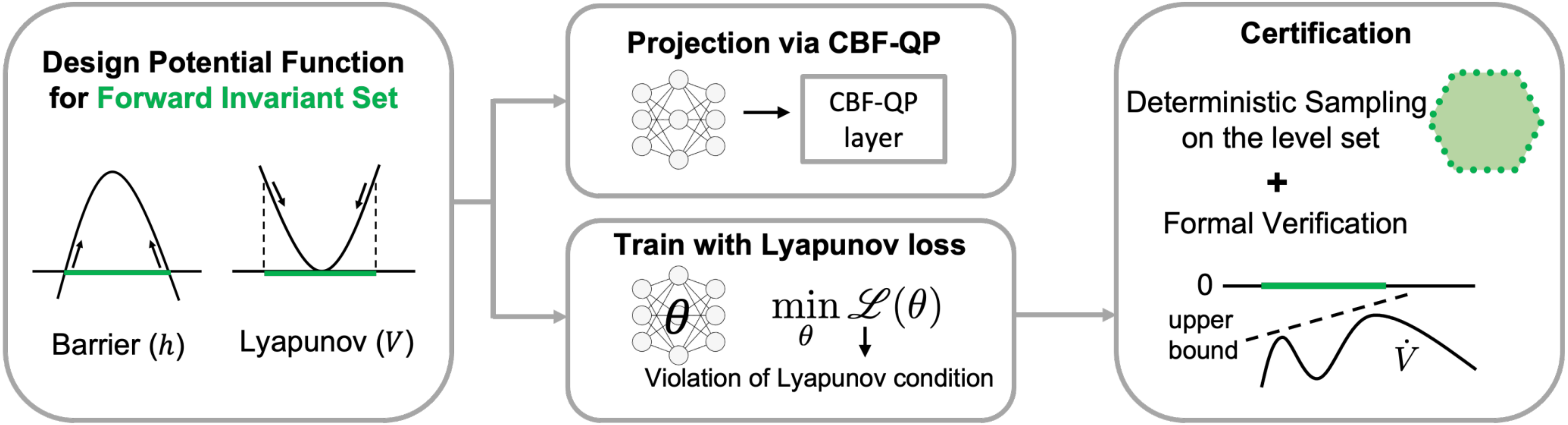
Certified Forward-Invariance in NODEs





Yujia Huang

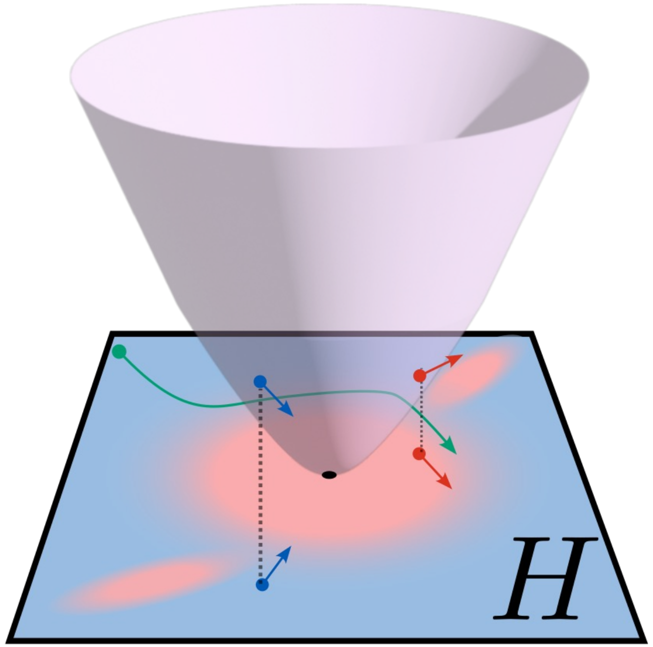
Ivan Jimenez Rodriguez

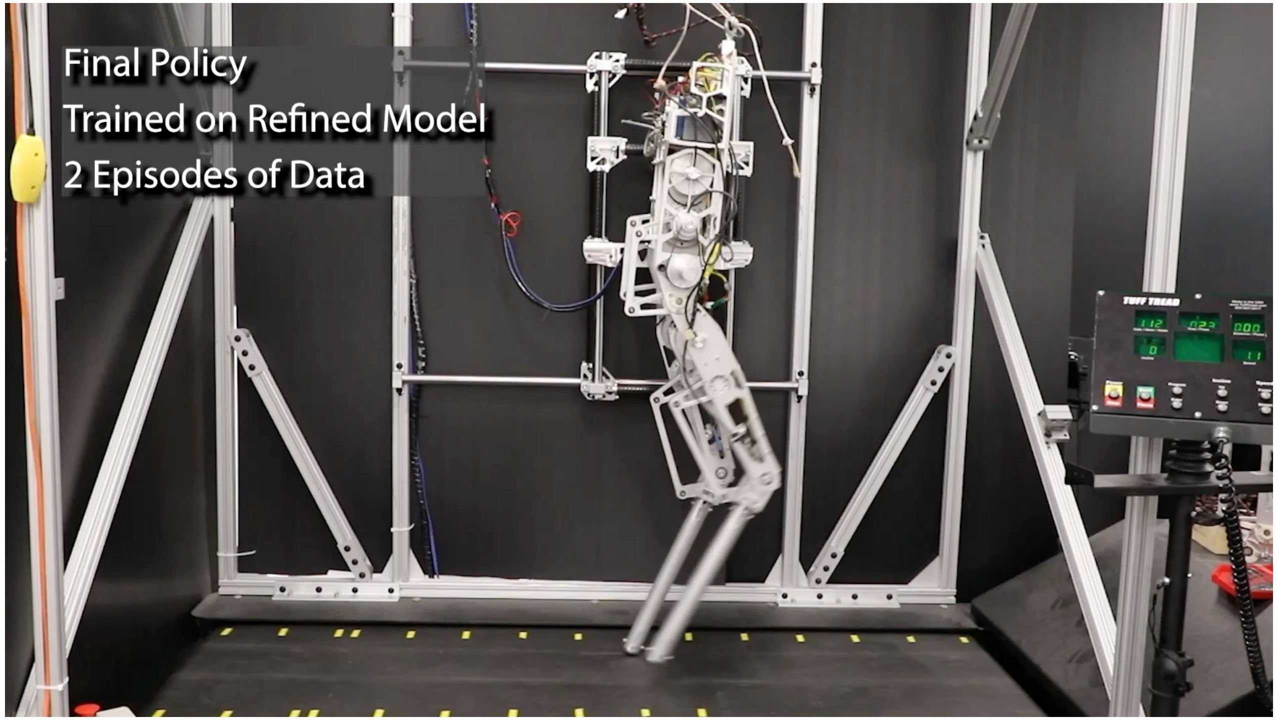


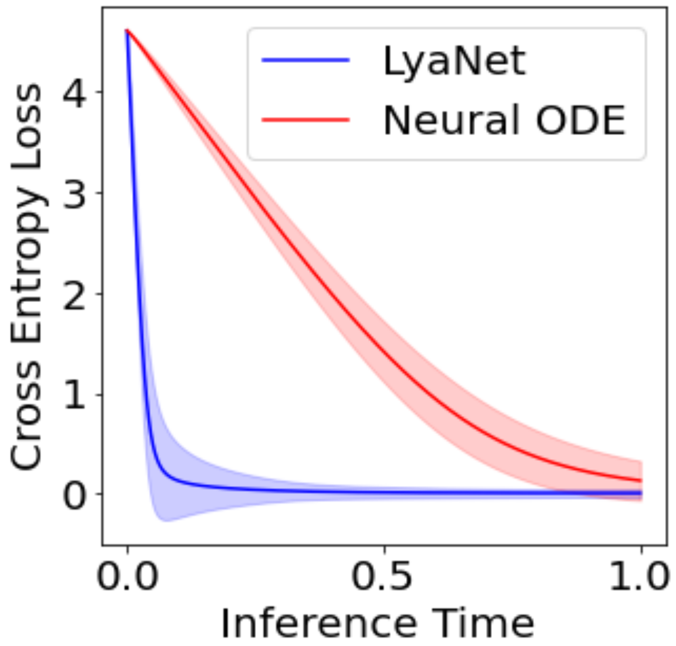
Certified Robust Forward Invariance (First Ever Result)

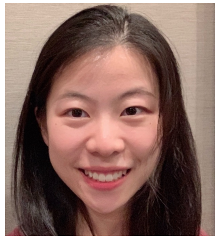
https://arxiv.org/abs/2210.16940

Summary:
Control-Theoretic Shaping of Neural ODEs



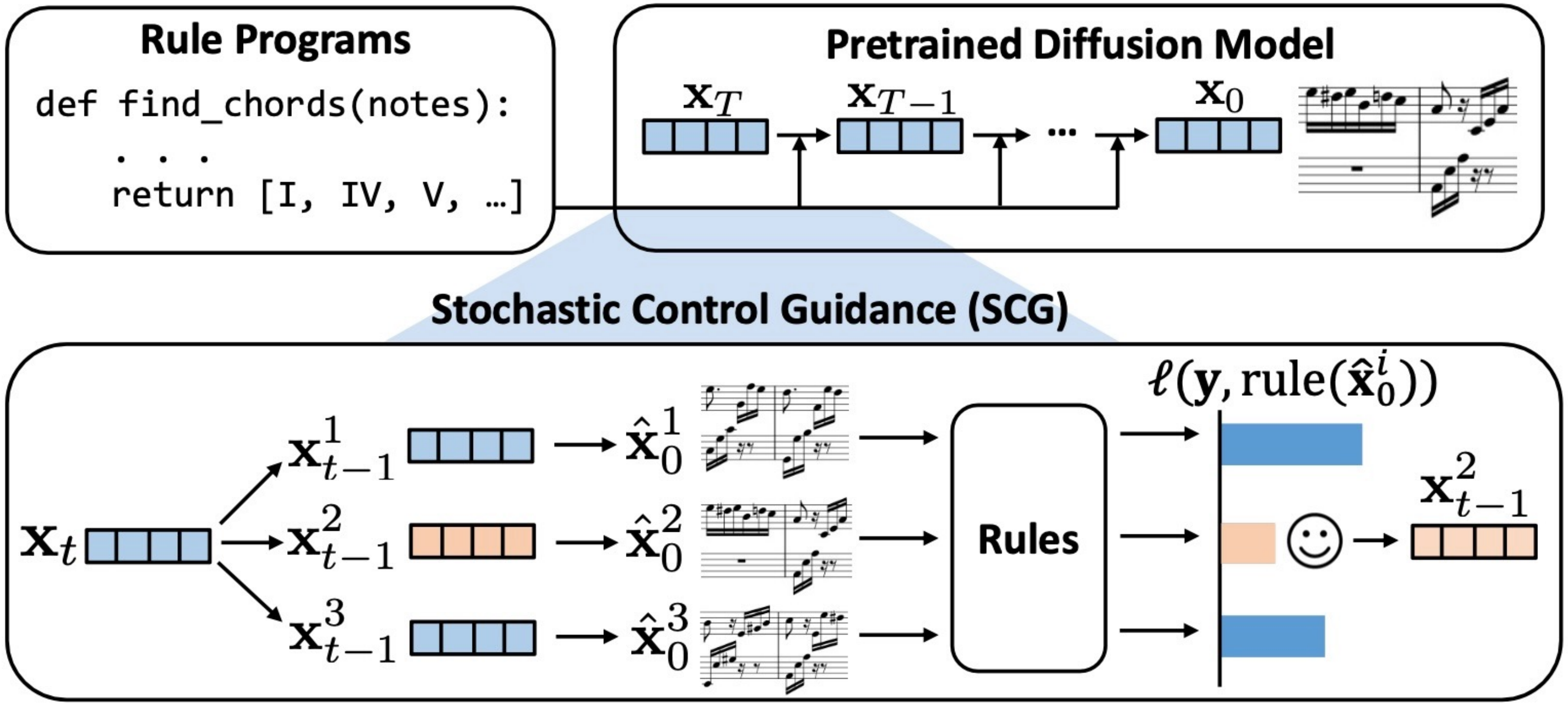


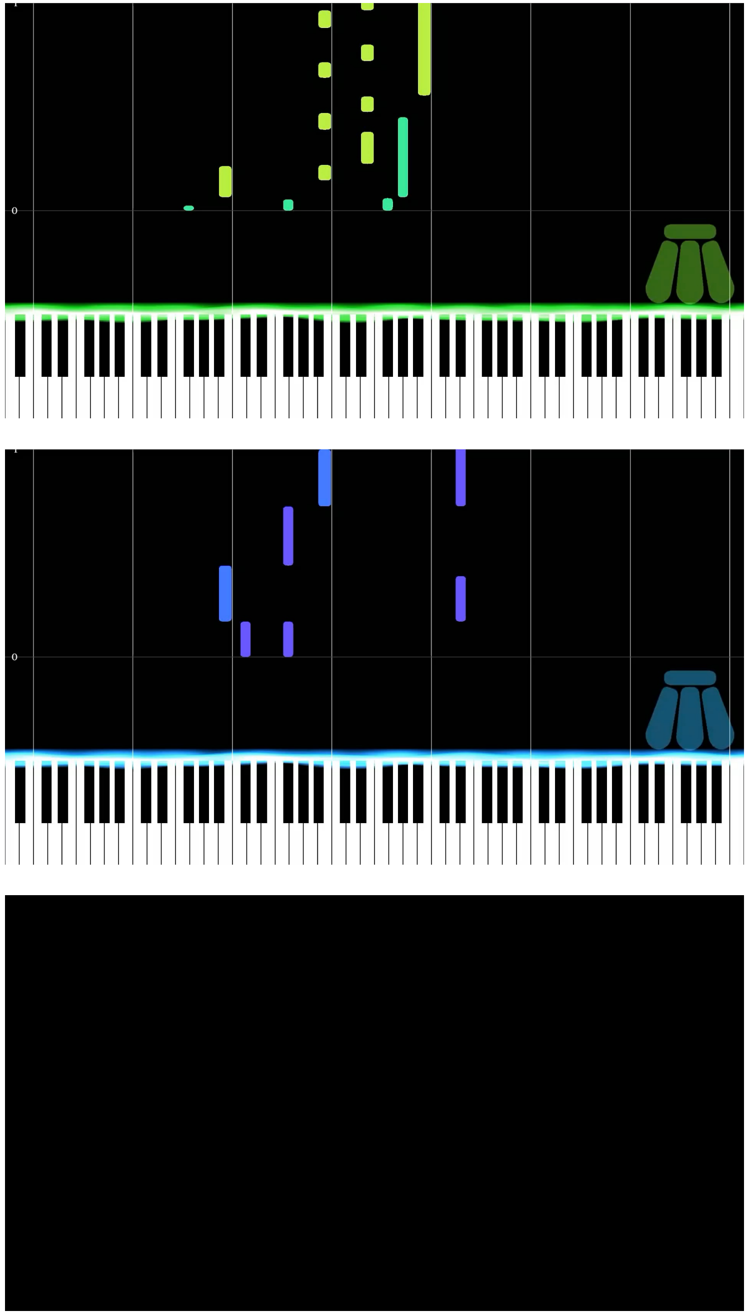




Yujia Huang

Aside: Symbolic Music Generation via Stochastic Control





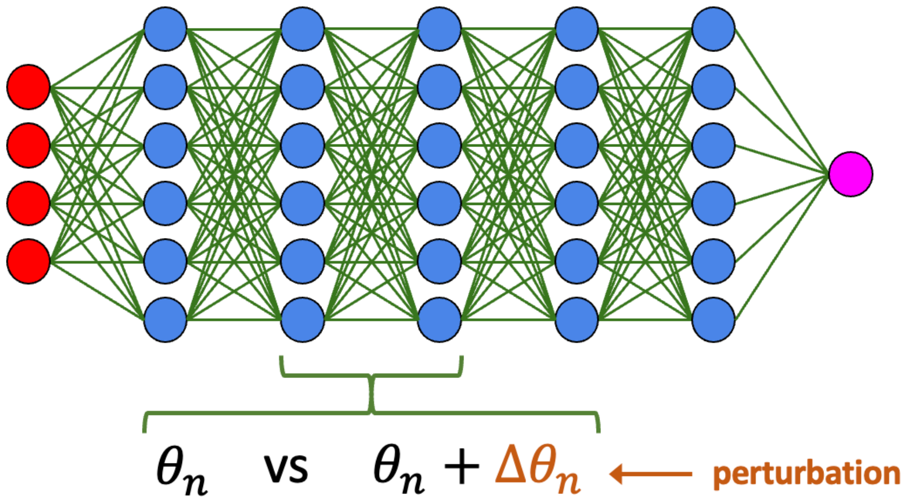
Symbolic Music Generation with Non-Differentiable Rule-Guided Diffusion

Yujia Huang, et al., arXiv https://scg-rule-guided-music.github.io/

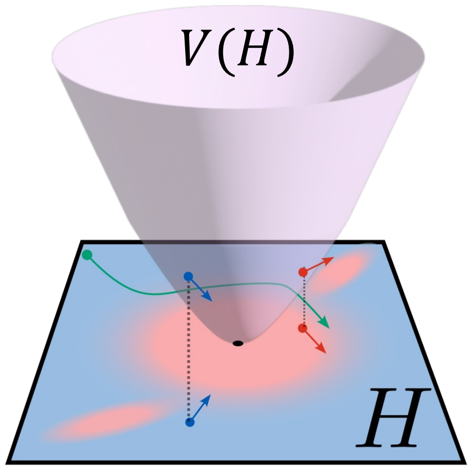
Towards Structure-Aware Theory of Deep Learning

- Neural Nets are not arbitrary black-box functions
  - Analyzing structure can lead to more nuanced theory
  - Can unlock new connections

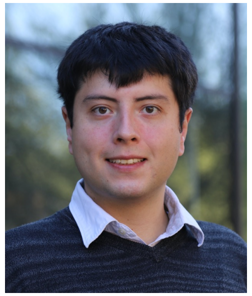
Per-Layer Perturbation Analysis

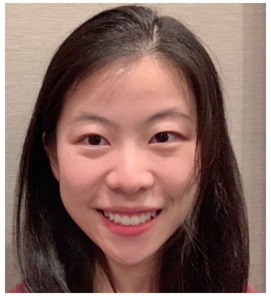


Contraction Analysis & Control Theory




Jeremy
Bernstein


Ivan
Jimenez Rodriguez


Yujia
Huang


Will
Compton

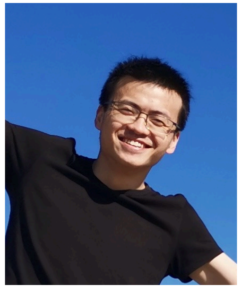

Kevin
Huang


Yang
Liu


Chris
Mingard


Noel
Csomay-Shanklin


Adishree
Ghatare


Qinsheng
Zhang


Chandramouli
Sastry

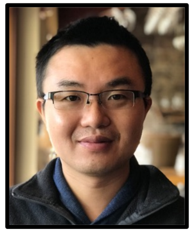

Yuanzhe
Liu


Siddharth
Gururani


Eric
Ambrose


Jiawei
Zhao


Ziniu
Hu


Ming-Yu
Liu


Arash
Vahdat


Sageev
Oore


Navid
Azizan


Markus
Meister


Anima
Anandkumar


Aaron
Ames


Yuanyuan
Shi


Huan
Zhang

References

- **Optimisation & Generalisation in Networks of Neurons**, Jeremy Bernstein, PhD Thesis, Caltech, 2022
- **On the distance between two neural networks and the stability of learning**, Jeremy Bernstein et al., NeurIPS 2020
- **Learning compositional functions via multiplicative weight updates**, Jeremy Bernstein et al., NeurIPS 2020
- **Learning by Turning: Neural Architecture Aware Optimisation**, Yang Liu*, Jeremy Bernstein*, et al., ICML 2021
- **LyaNet: A Lyapunov Framework for Training Neural ODEs**, Ivan Jimenez Rodriguez et al., ICML 2022
- **FI-ODE: Certified and Robust Forward Invariance in Neural ODEs**, Yujia Huang*, Ivan Jimenez Rodriguez*, et al., *arxiv*
- **Neural Gaits: Learning Bipedal Locomotion via Control Barrier Functions and Zero Dynamics Policies**, Ivan Jimenez Rodriguez*, Noel Csomay-Shanklin*, et al., L4DC 2022
- **Robust Agility via Learned Zero Dynamics Policies**, Noel Csomay-Shanklin*, Will Compton*, Ivan Jimenez Rodriguez*, et al., *(arxiv soon)*
- **Automatic Gradient Descent: Deep Learning without Hyperparameters,** Jeremy Bernstein*, Chris Mingard*, et al. *arxiv*
- **Symbolic Music Generation with Non-Differentiable Rule Guided Diffusion,** Yujia Huang, et al., *arxiv*

Thanks!

@yisongyue http://www.yisongyue.com