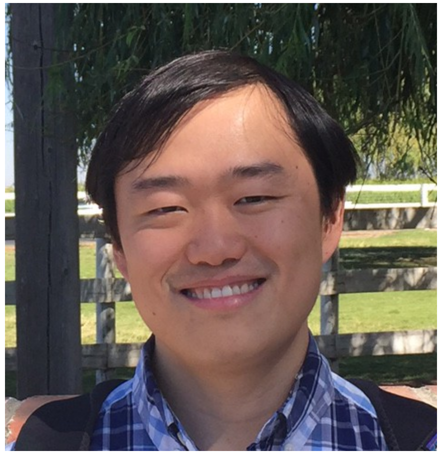
Understanding Deep Models With Teacher-student Setting.

Yuandong Tian

Research Scientist and Manager
Facebook AI Research

facebook Artificial Intelligence

Career Path



Theoretical Understanding of Models and Algorithms

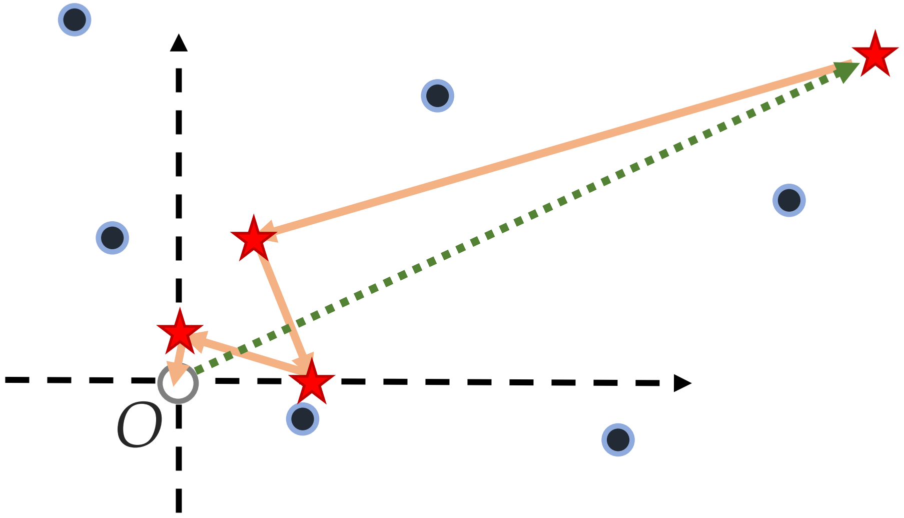
Computer Vision | Reinforcement Learning

2006 2008 2013 2015 2019

PhD in Carnegie Mellon | Waymo | Facebook AI Research

Data-Driven Descent (PhD work)

Template I_0 Deformed image I_{p^*} Landmark estimation p



To achieve $||p - p^*|| \leq \epsilon$

Method	Sample complexity
Gradient descent	Local optimality
Nearest Neighbor	$O(\epsilon^{-d})$
[Y. Tian and S. Narasimhan, CVPR 10]	$O(C^d \log \epsilon^{-1})$
[Y. Tian and S. Narasimhan, ICCV 13, ***Marr Prize Honorable Mention***]	$O(C_1^d + C_2 \log \epsilon^{-1})$

DarkForestGo

DarkForest versus Koichi Kobayashi (9p)

[Better Computer Go Player with Neural Network and Long-term Prediction, Y. Tian and Y. Zhu, ICLR 2016]

MIT Technology Review

Intelligent Machines

How Facebook's AI Researchers Built a Game-Changing Go Engine

The best human players easily beat the best computer-based Go engines. That looks set to change thanks to a new approach pioneered by Facebook's artificial intelligence researchers.

by **Emerging Technology from the arXiv** Dec 4, 2015

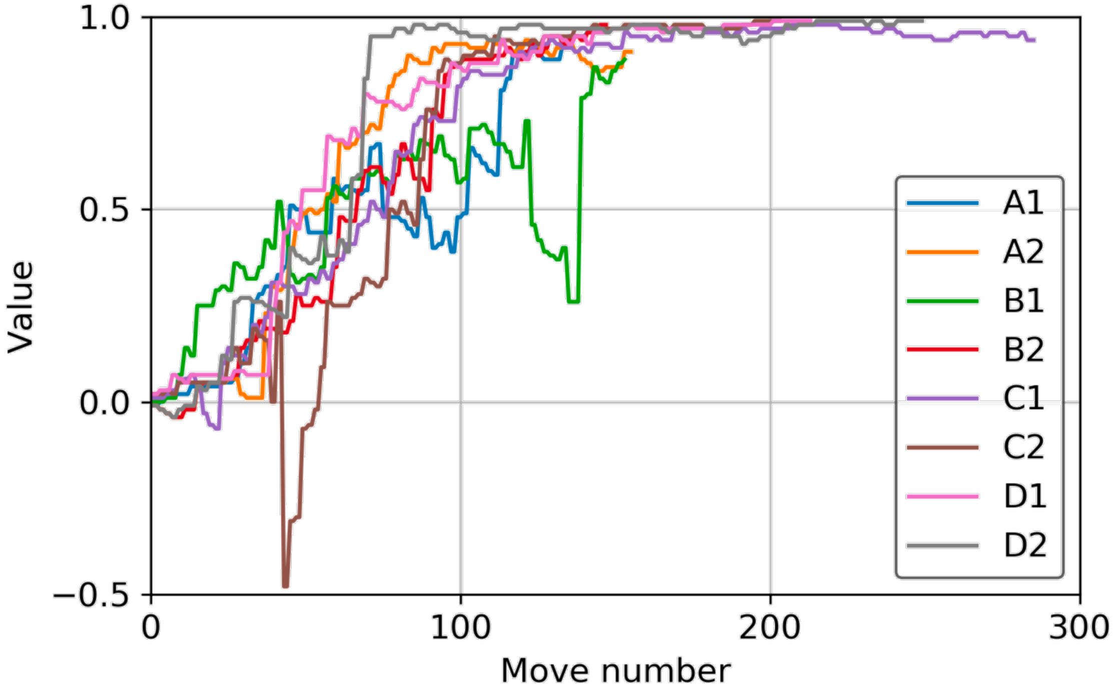
One of the last bastions of human mastery over computers is the game of Go —the best human players beat the best Go engines with ease.

ELF OpenGo

Vs top professional players

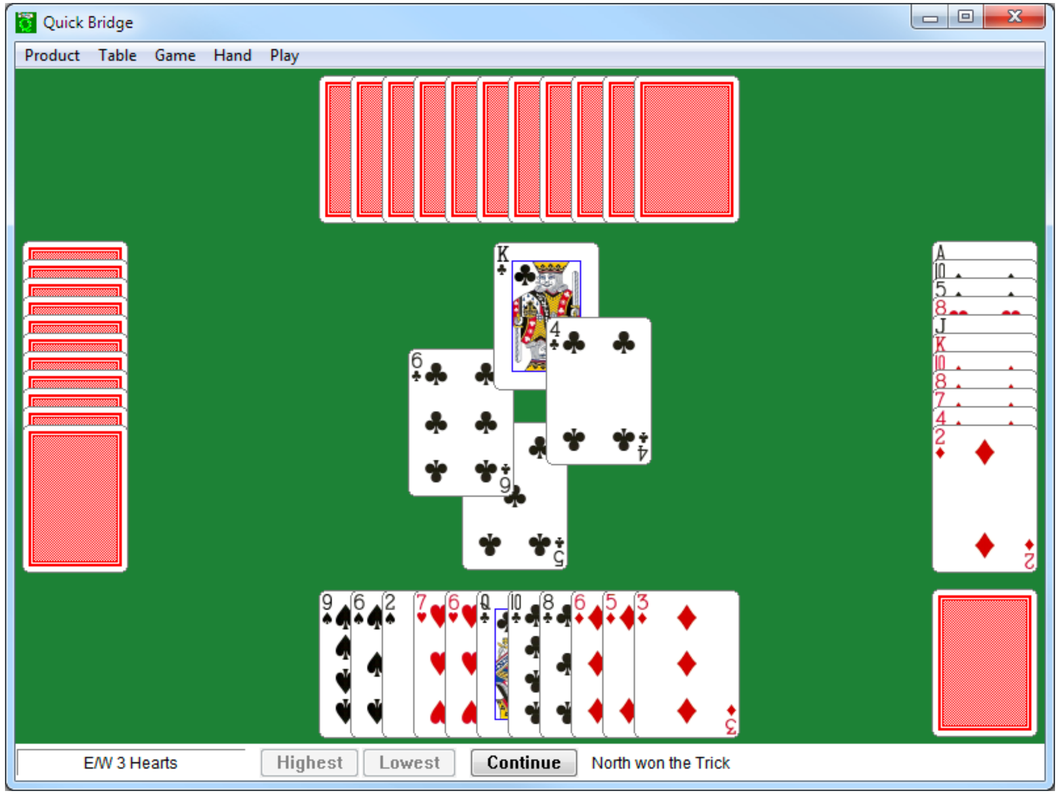
Name (rank)	ELO (world rank)	Result
Kim Ji-seok	3590 (#3)	5-0
Shin Jin-seo	3570 (#5)	5-0
Park Yeonghun	3481 (#23)	5-0
Choi Cheolhan	3466 (#30)	5-0

Single GPU, 80k rollouts, 50 seconds
Offer unlimited thinking time for the players



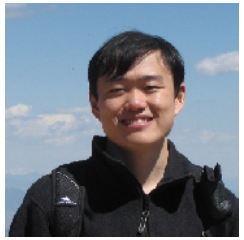
[ELF OpenGo: An Analysis and Open Reimplementation of AlphaZero, Y. Tian et al, ICML 2019]

Contract Bridge

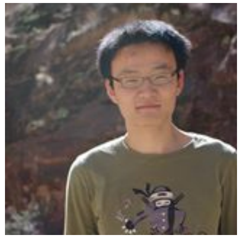


SoTA performance on Bridge Bidding

A new theoretical framework for multi-agent collaborative games



Yuandong Tian

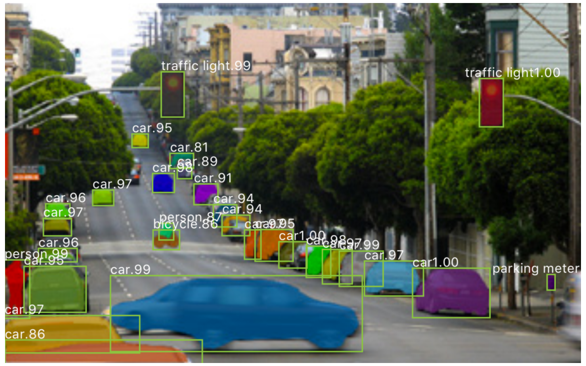


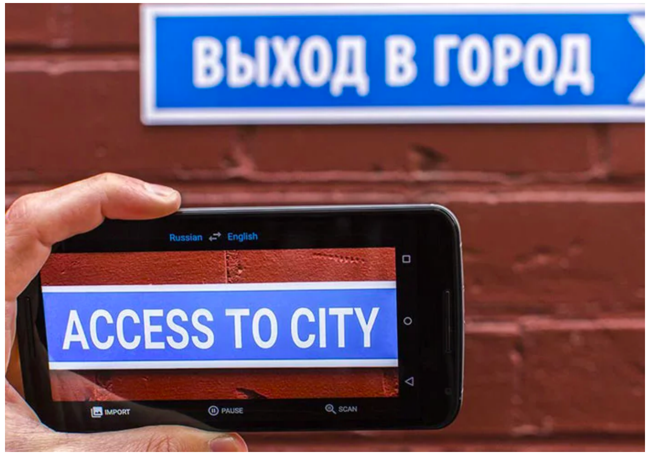
Qucheng Gong

Tina Jiang

[Y. Tian et al., Joint Policy Search for Collaborative Multi-agent Imperfect Information Game, NeurIPS 2020]

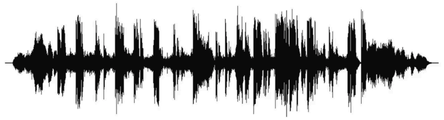
Great Empirical Success



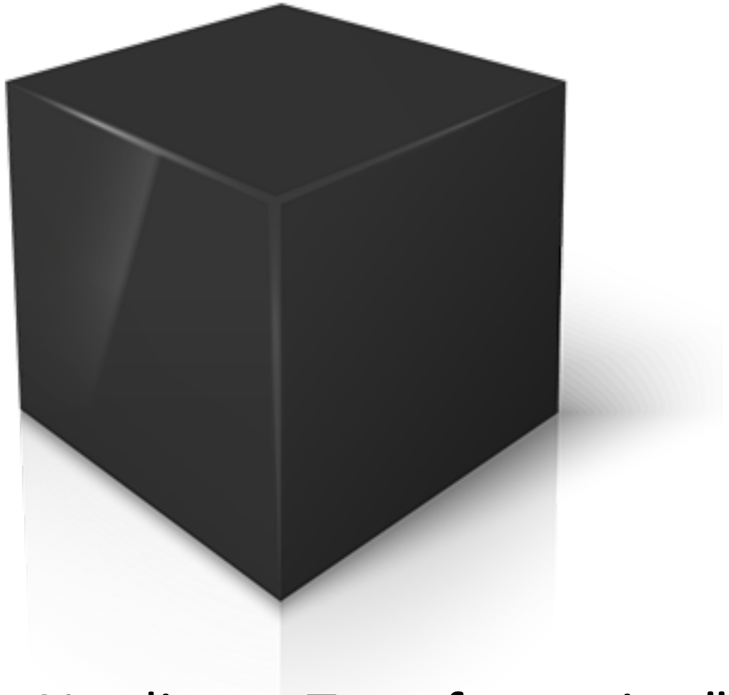


How do deep models work?

Input



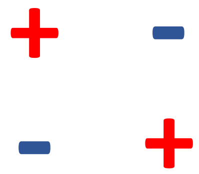
This is an apple



“Some Nonlinear Transformation”

Output

Three Major Problems

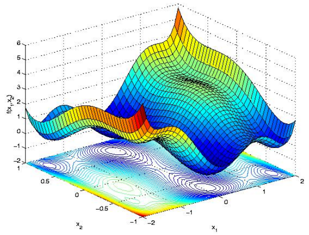


Understanding how Deep Models work

Expressibility

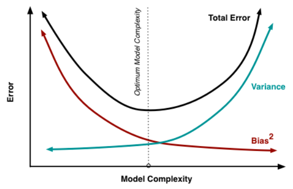
"Neural Network is a universal approximator"
"Deep Models can express functions more efficiently than shallow ones"

Optimization



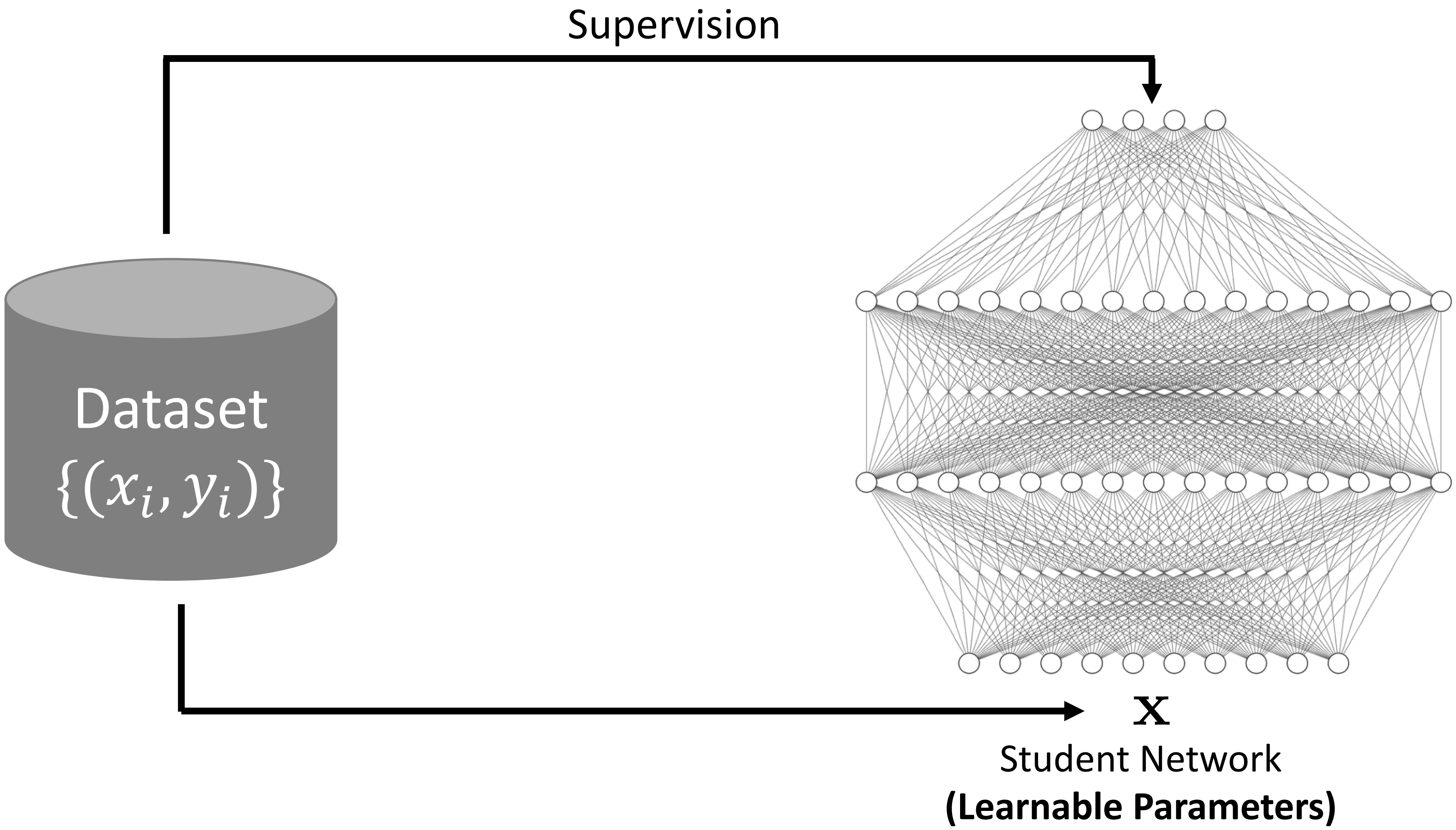
"Gradient vanishing/exploding"
"Gradient Descent might get stuck at saddle point / local minima"
"Can GD/SGD go to global optima? How fast?"

Generalization



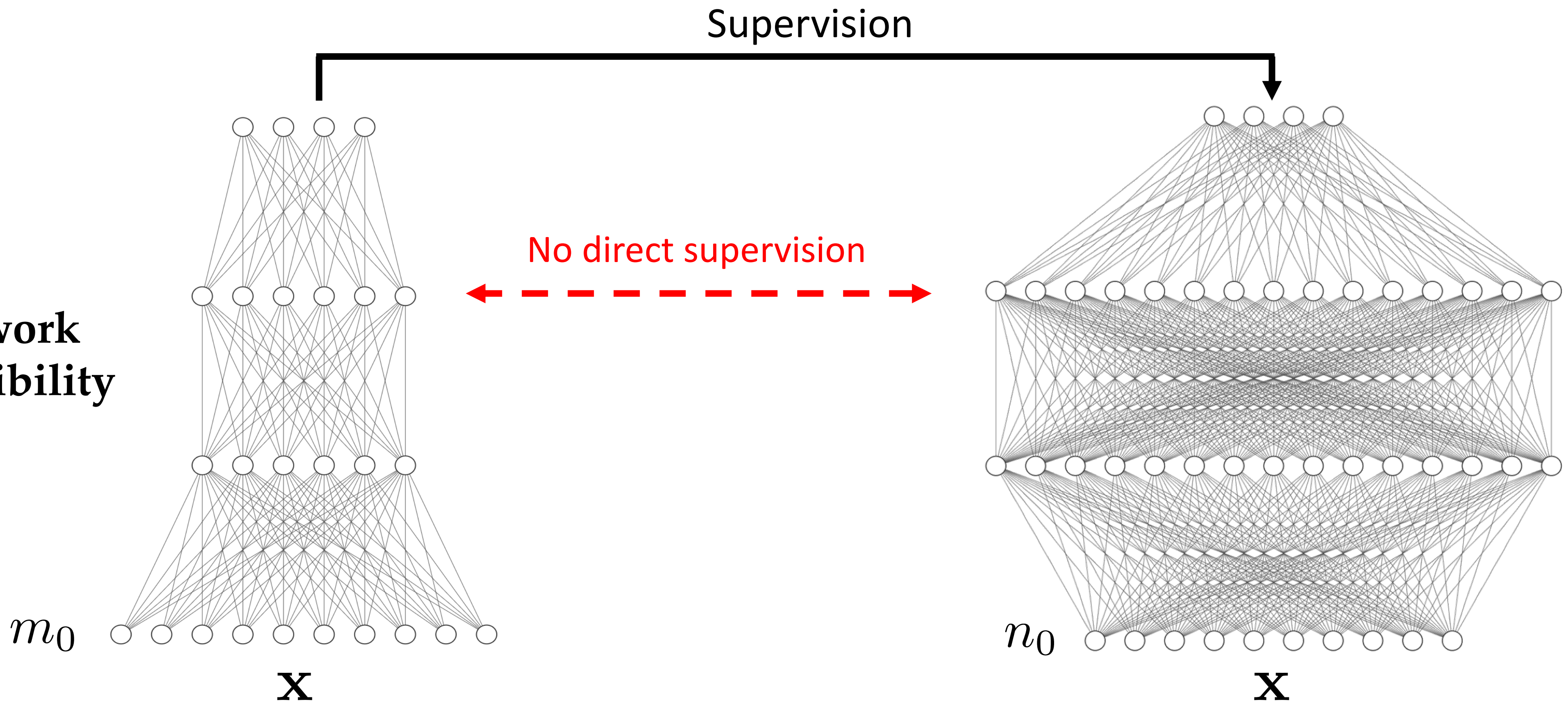
"Does zero training error often lead to overfitting?"
"More parameters might lead to overfitting."

Supervised Learning



Student-Teacher Setting

By Network Expressibility



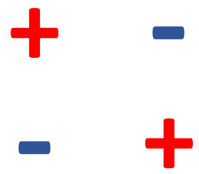
Teacher/Oracle Network
(Fixed parameters)

Student Network
(Learnable Parameters)

Why Student-Teacher Setting?

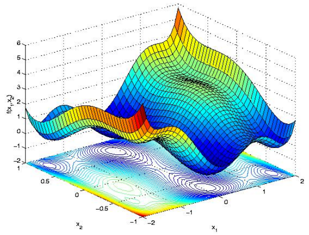
Understanding how Deep Models work

Expressibility



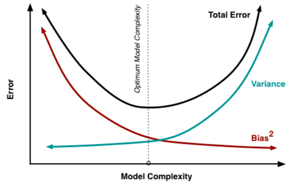
Provide a target function with bounded complexity

Optimization



Study fine dynamics behaviors by comparing with teacher

Generalization



Our focus → ***Student Specialization*** yields generalization

Old History of Teacher-Student Setting

$$\epsilon(\boldsymbol{J}) = \frac{1}{2}\langle|f(\boldsymbol{J},\boldsymbol{\xi}) - f(\boldsymbol{B},\boldsymbol{\xi})|^2\rangle_{\xi} \qquad f(\boldsymbol{J},\boldsymbol{\xi}) = \sum_{i=1}^{K}\sigma(\boldsymbol{J}_i \cdot \boldsymbol{\xi})$$

Study when the input dimension $n_0 = m_0 \to +\infty$ (i.e., **thermodynamics limits**)

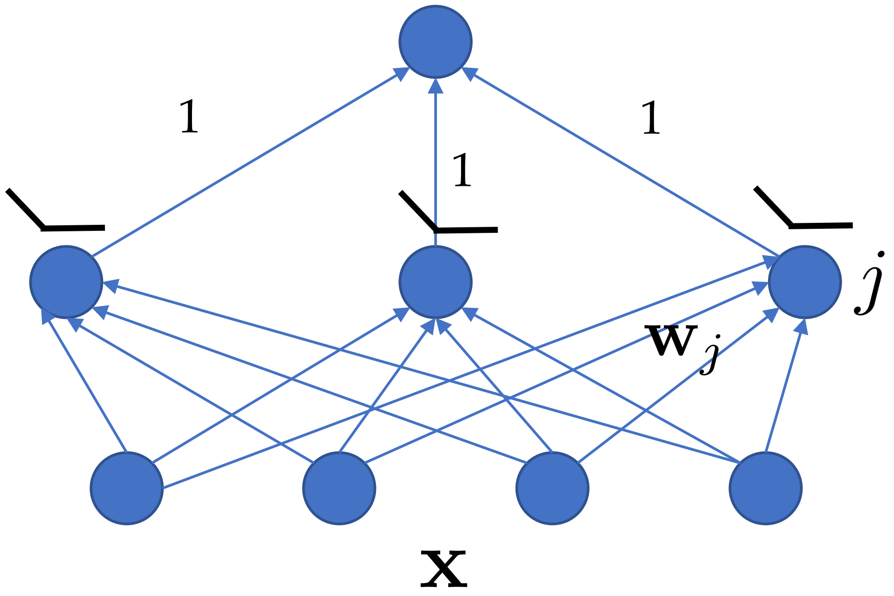
In some situations, student nodes are "***specialized***" to teacher node

One layer of trainable parameters
Nonlinear function $\sigma(x) = \mathrm{erf}(x\ /\ 2)$
Locally linearized analysis around symmetry breaking plane and final solution

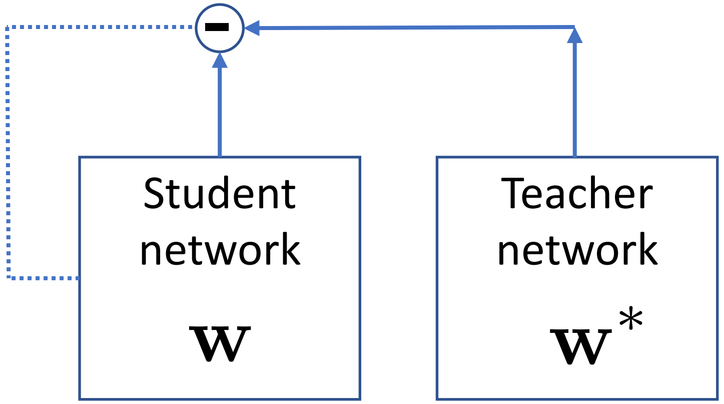
[On-line learning in soft committee machines, Saad & Solla, Phys. Rev 1995]
[S. Goldt et al, Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup, NeurIPS 2019]

Simplest Teacher Student Setting: ReLU networks with Gaussian Inputs

$$g(\mathbf{x};\mathbf{w}) = \sum_{j=1}^{K} \sigma(\mathbf{w}_j^T \mathbf{x})$$



$$\sigma(x) = \max(x, 0)$$

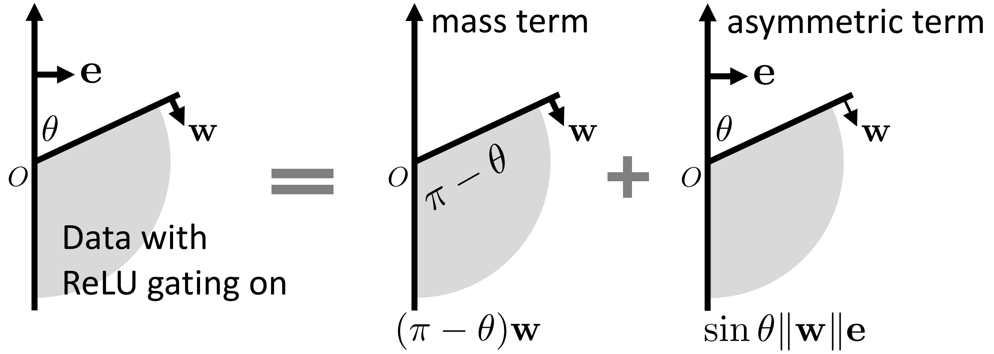


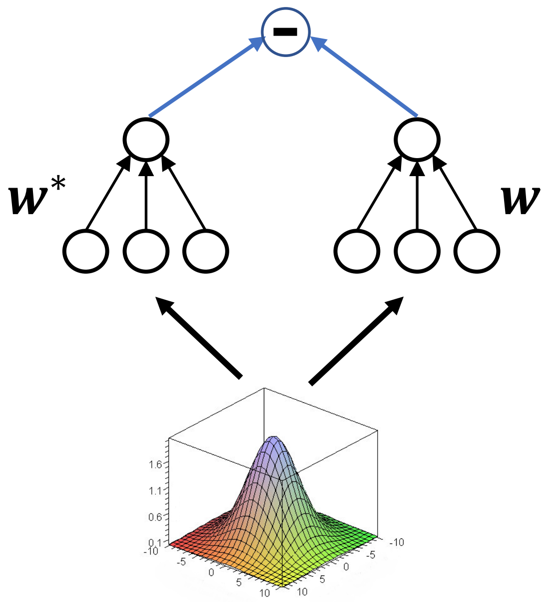
$$J(\mathbf{w}) = \frac{1}{2}\|g(X;\mathbf{w}^*) - g(X;\mathbf{w})\|^2$$

We focus on *population gradient*

[Y. Tian, *An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis*, ICML 2017]

An Analytic Formula





Close-form Population Gradient:

$$\mathbb{E}\left[\nabla_{\mathbf{w}} J\right] = \boxed{\frac{N}{2}(\mathbf{w} - \mathbf{w}^*)} + \boxed{\frac{N}{2\pi}\left(\theta \mathbf{w}^* - \frac{\|\mathbf{w}^*\|}{\|\mathbf{w}\|}\sin\theta \mathbf{w}\right)}$$

Linear component.
Global convergence if this is the only term

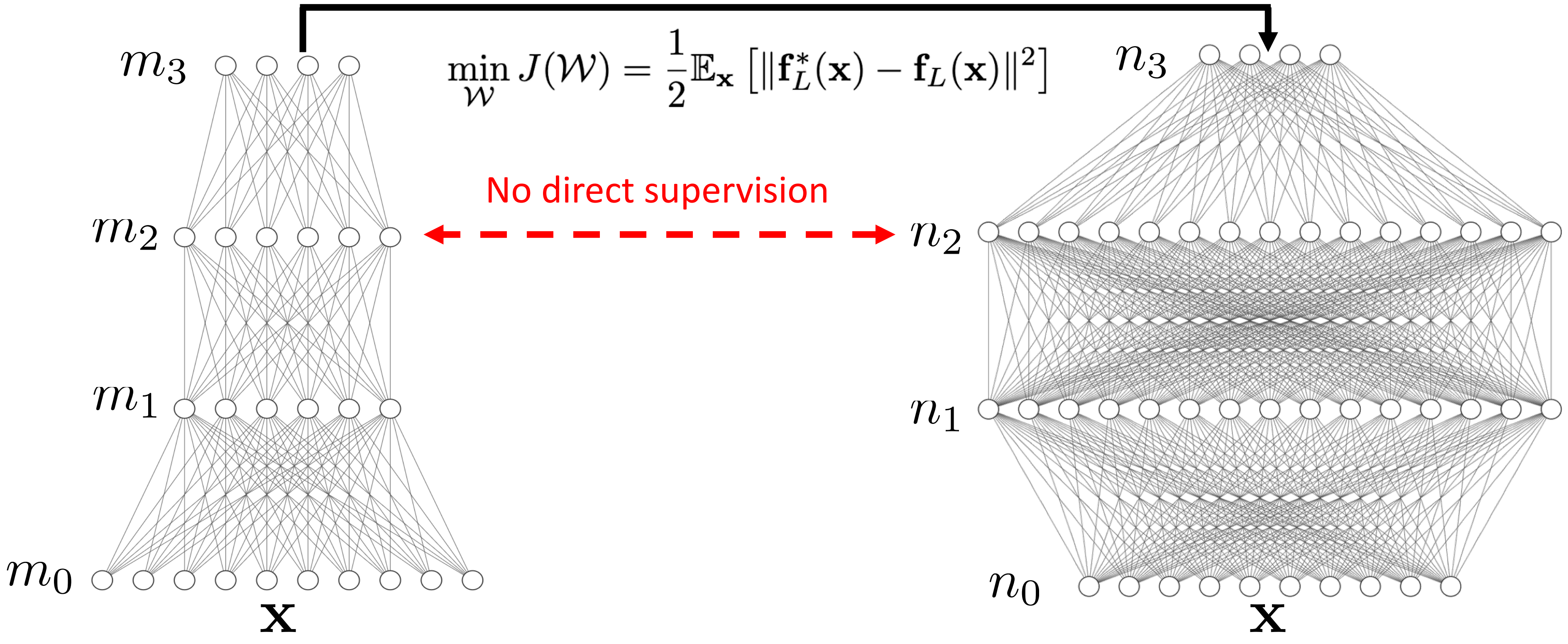
Nonlinear component
due to ReLU gating

[Y. Tian, *An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis*, ICML 2017]

Multi-layer ReLU network

1. Finite m_0 and n_0
2. Works for $n_i \geq m_i$
(no crazy overparameterization)

Different From Neural Tangent Kernel



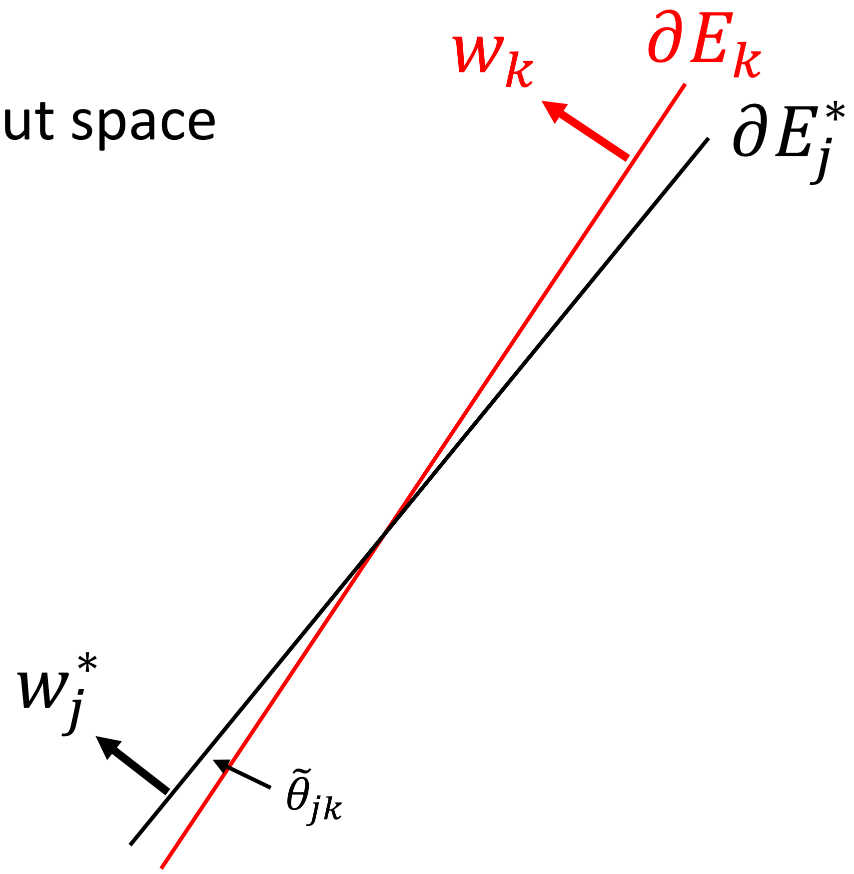
Teacher/Oracle Network
(Fixed parameters)

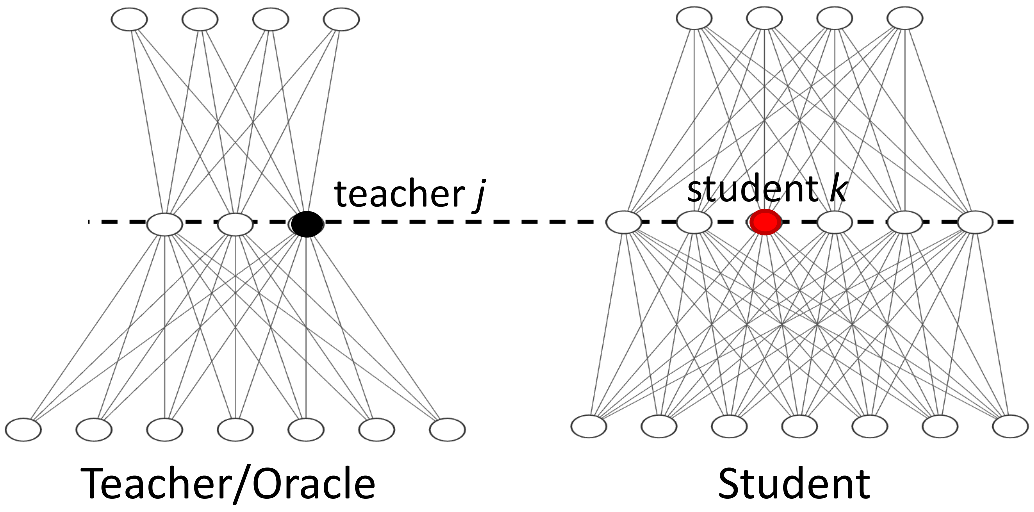
(Over-realized) Student Network
(Learnable Parameters)

[Y. Tian, *Student Specialization in Deep ReLU Networks With Finite Width and Input Dimension*, ICML 2020]

Student Specialization

Input space





∂E_k: Boundary of node k

∂E_j^*: Boundary of teacher node j

ϵ-alignment: $\sin \tilde{\theta}_{jk} \leq \epsilon$ and $\left| b_j - b_k^* \right| \leq \epsilon$

Main Question

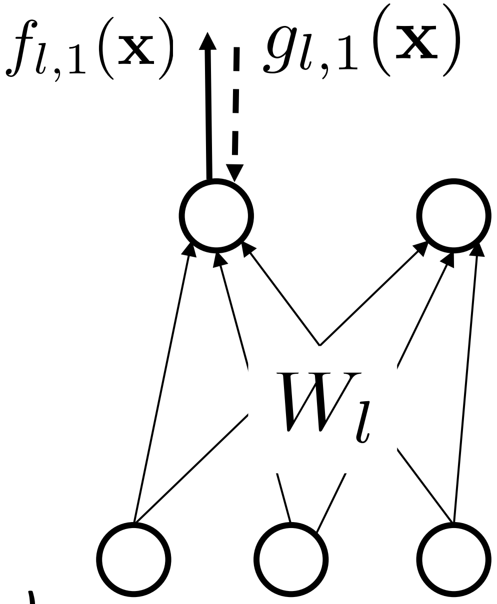
Small gradient
at every training sample
during training

→ ? ? ? →

Student aligns
with the teacher

→ Small training error leads to good generalization

Notation



Layer l
(n_l nodes)

Layer $l-1$
(n_{l-1} nodes)

Activation

$$\mathbf{f}_l(\mathbf{x}) = \begin{bmatrix} f_{l,1}(\mathbf{x}) \\ f_{l,2}(\mathbf{x}) \end{bmatrix}$$

Gradient

$$\mathbf{g}_l(\mathbf{x}) = \begin{bmatrix} g_{l,1}(\mathbf{x}) \\ g_{l,2}(\mathbf{x}) \end{bmatrix}$$

Weight update rule: $\dot{W}_l = \mathbb{E}_{\mathbf{x}}\left[\mathbf{f}_{l-1}(\mathbf{x})\mathbf{g}_l^{\intercal}(\mathbf{x})\right]$

GD: expectation taken over the entire dataset
SGD: expectation taken over a batch

Lemma1: Recursive Gradient Rule

For layer l, there exists $A_l(x)$ and $B_l(x)$ so that:

$$\mathbf{g}_l(\mathbf{x}) = D_l(\mathbf{x}) \left[A_l(\mathbf{x})\mathbf{f}_l^*(\mathbf{x}) - B_l(\mathbf{x})\mathbf{f}_l(\mathbf{x})\right]$$

Student gradient

Student gating

Teacher mixture

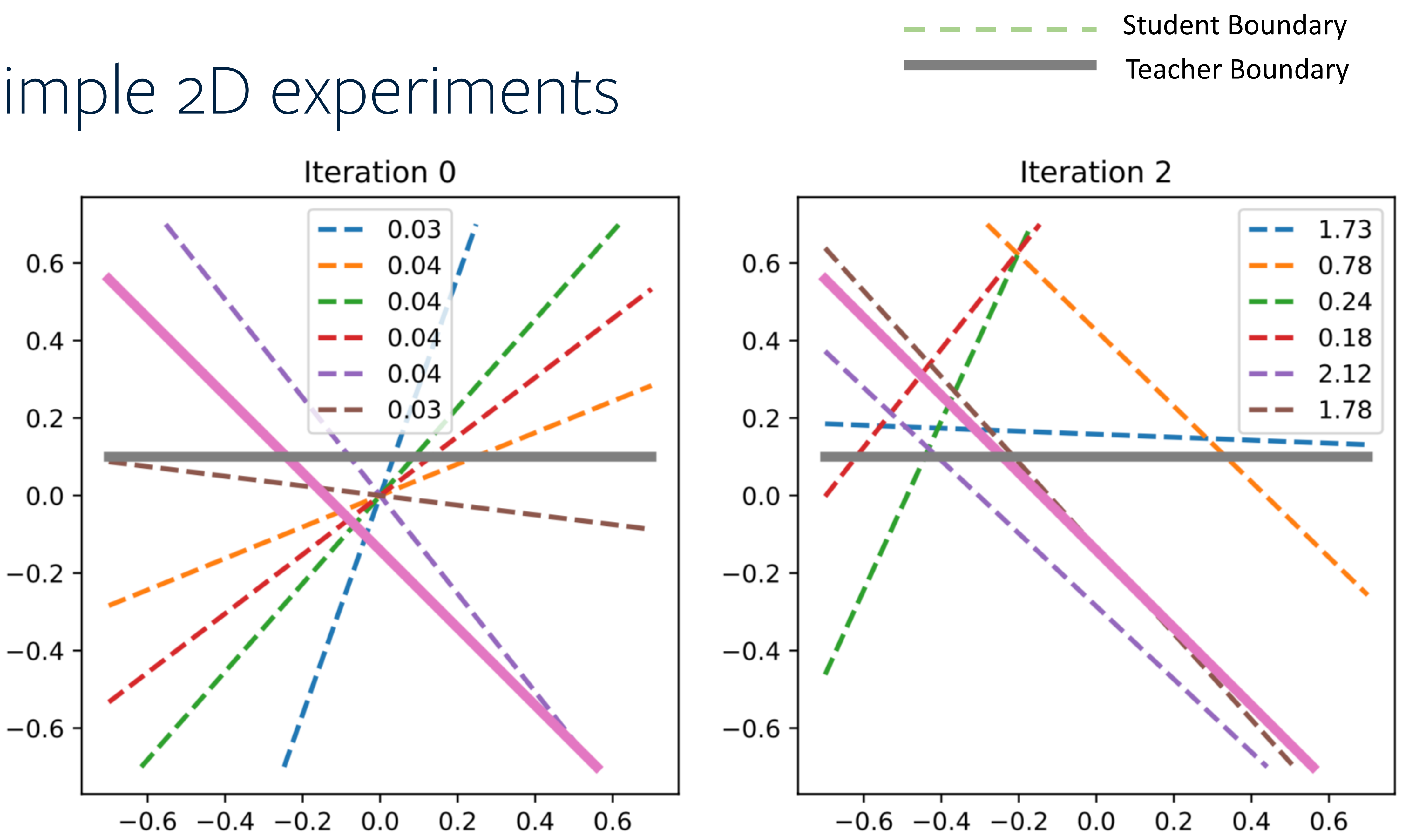
Student mixture

$A_l(x)$ and $B_l(x)$ are **piece-wise constant.**

A Demonstrative Case:

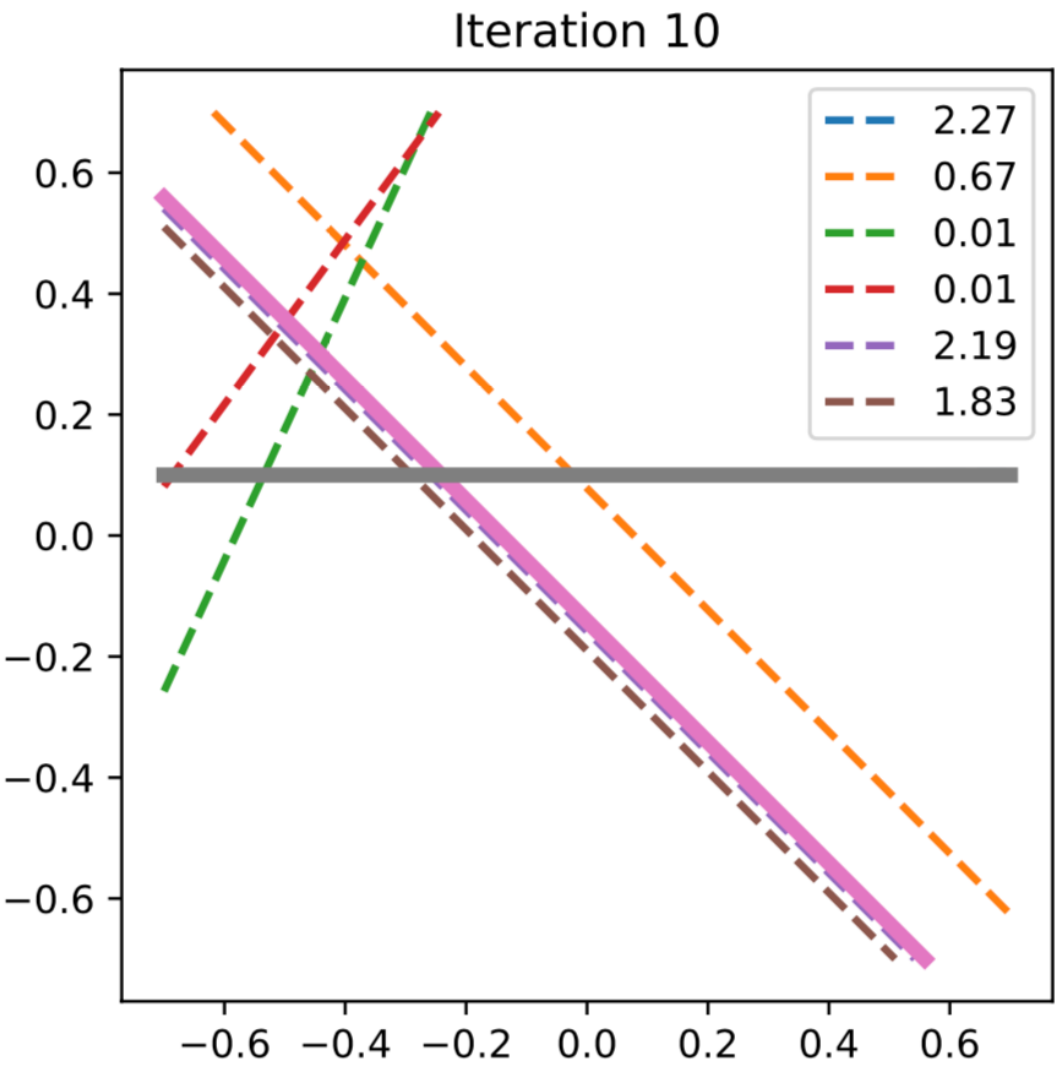
Two-layer Network, **Zero Gradient** and **Infinite** Samples

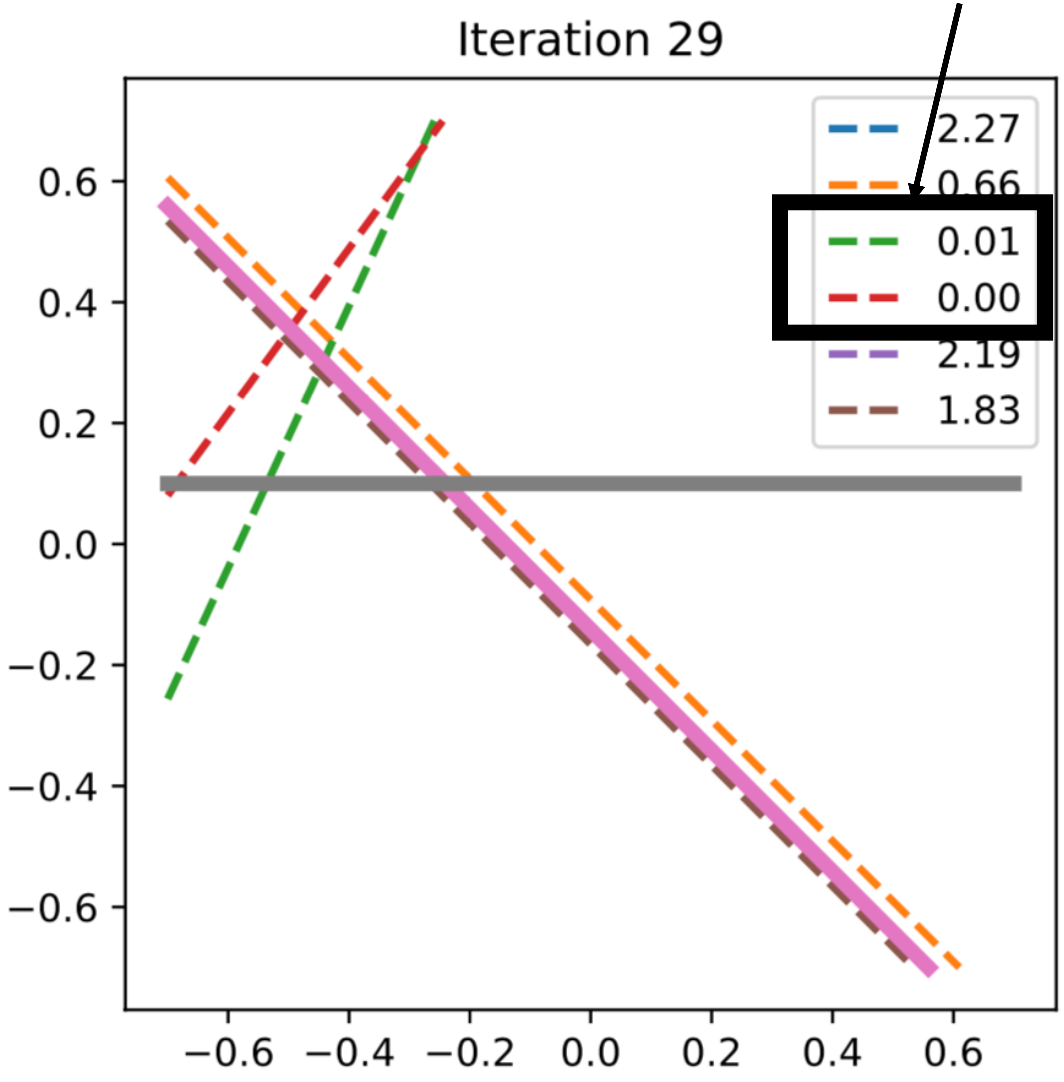
Simple 2D experiments



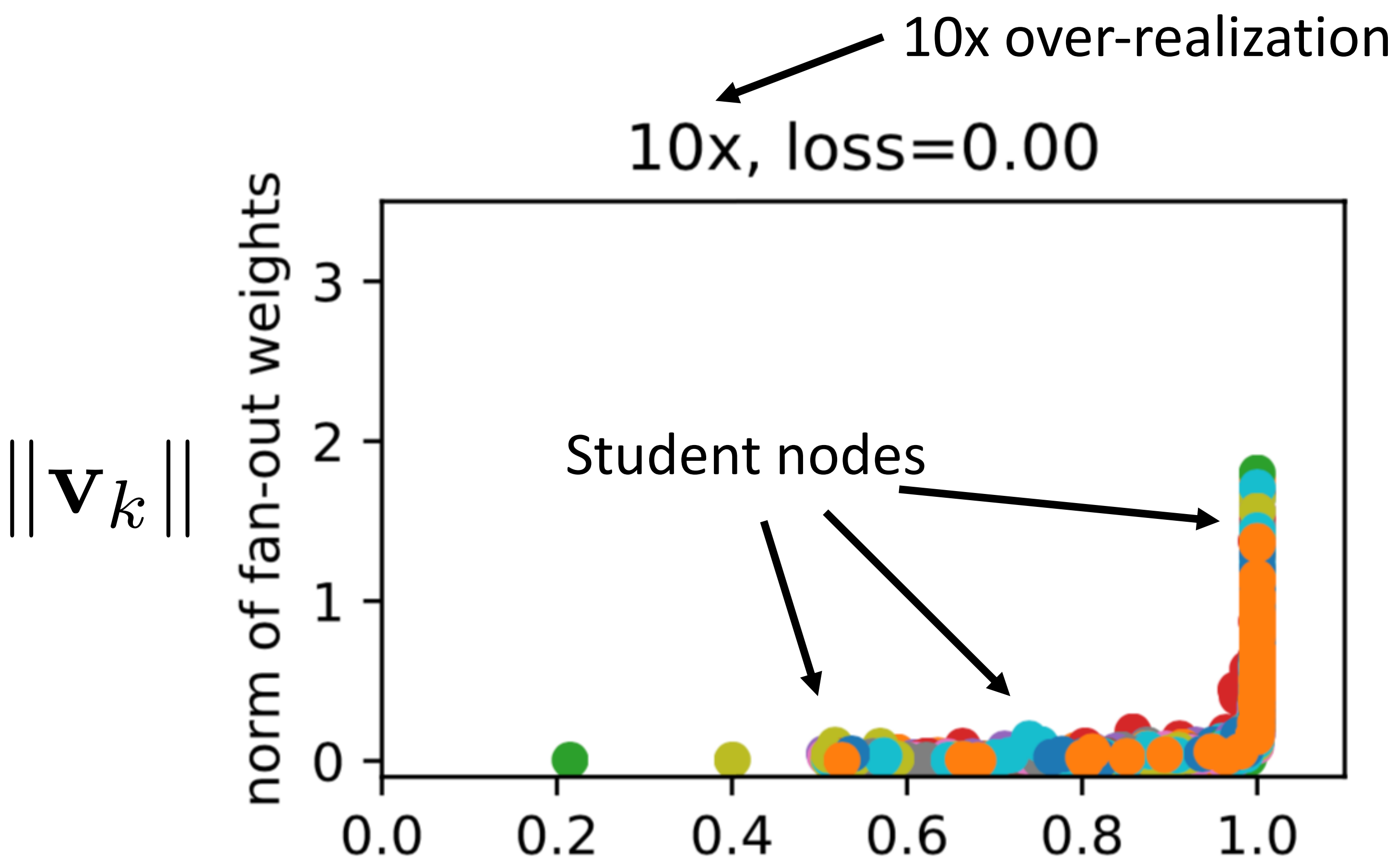
Simple 2D experiments

Norm of the fan-out weights.



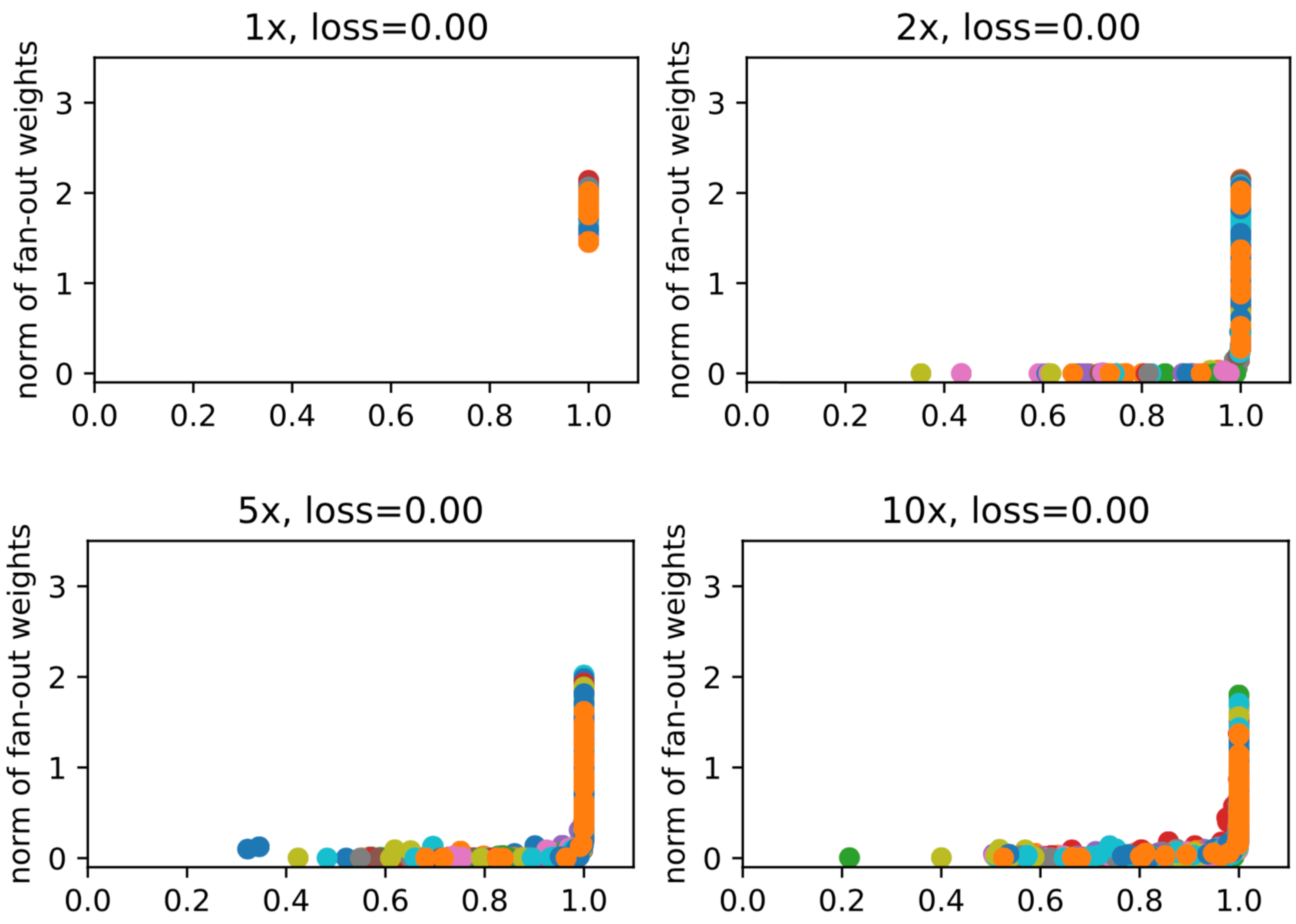


L-shape curve at convergence



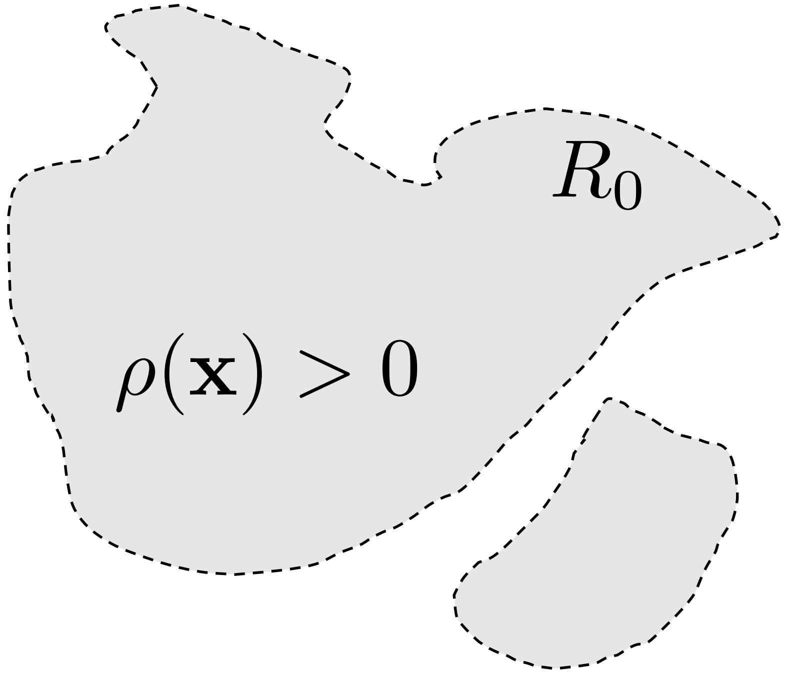
Normalized correlation of a student node
to its best correlated teacher (degree of specialization)

L-shape curve at convergence



Assumption of the dataset

No parametrized assumptions



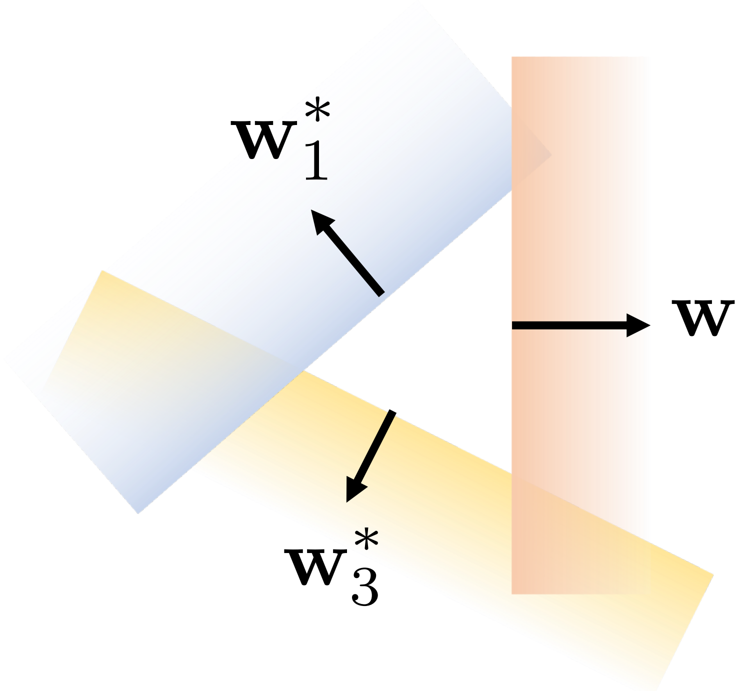
Infinite dataset!

(Region needs to have interiors)

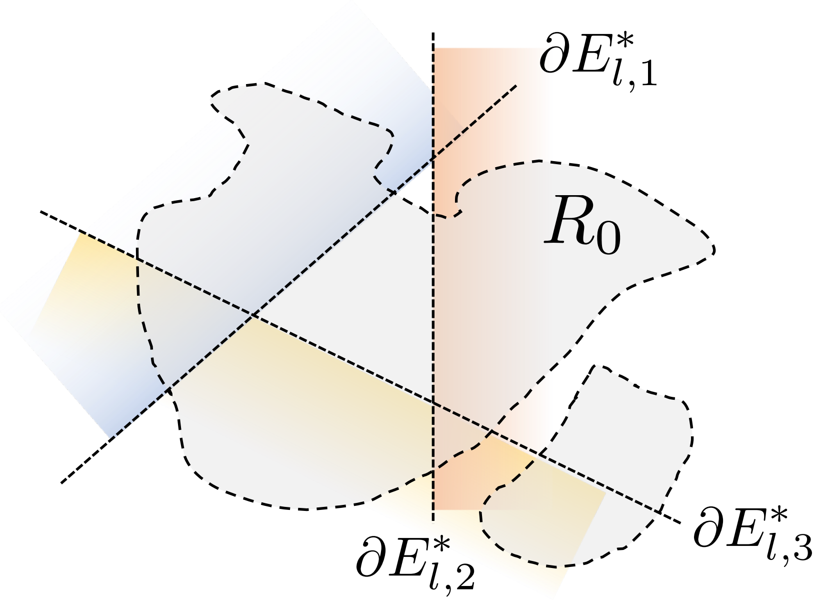
Full rank

Assumptions on Teacher Network

- Cannot reconstruct arbitrary teachers
  - e.g., all ReLU nodes are dead

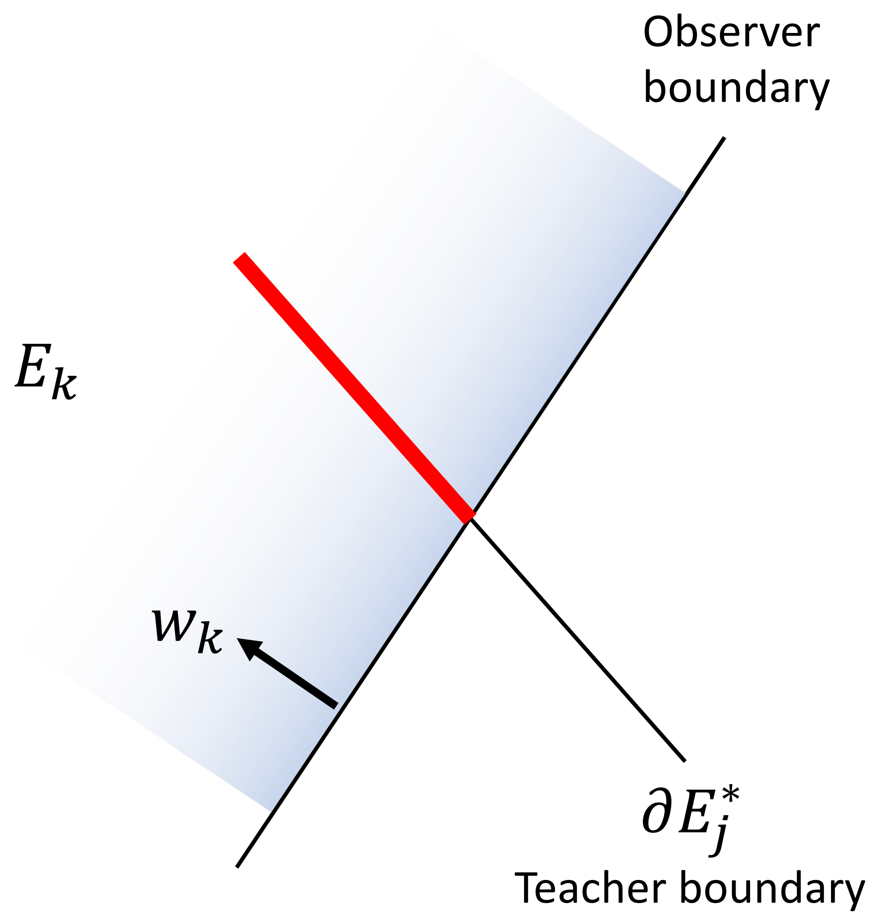


Distinct teacher nodes



Teacher's ReLU boundary are **visible** in the dataset

Definition of "Observation"



E_k: Activation region of node *k*

$$\partial E_j^* \cap E_k \neq \emptyset$$

Teacher *j* is **observed** by a student *k*

Main results: Alignment could happen!

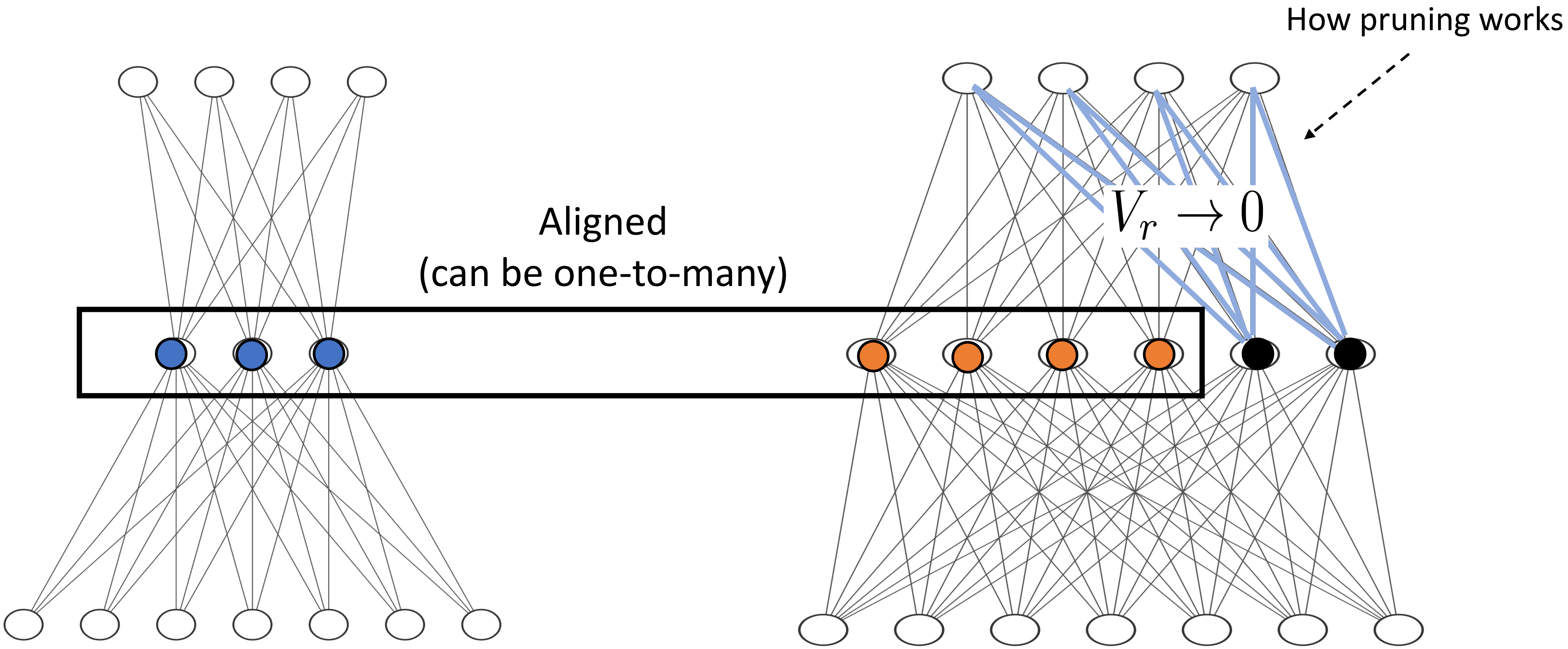
$\boldsymbol{g}_1(x) = \mathbf{0}$ for all $x \in R_0$
(all input gradients at layer 1 is *zero* at all training samples)

Teacher node *j* is **observed** by a student node *k*

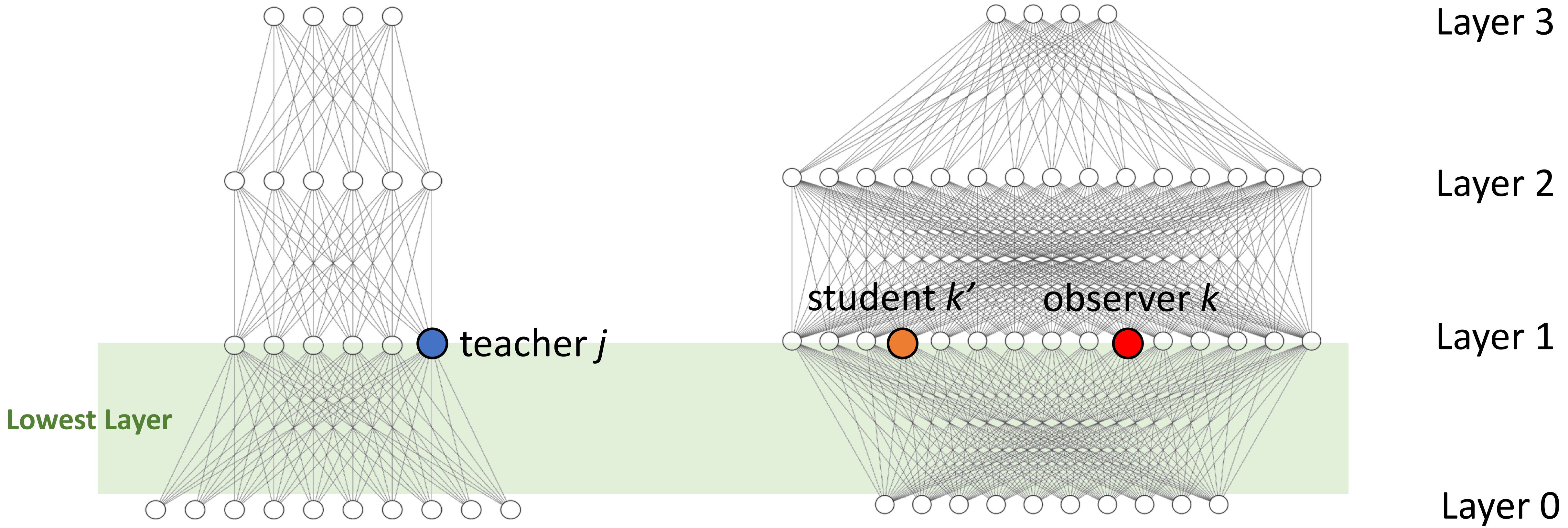
→

Teacher *j* is **aligned with** at least one student *k'*

What happens to unaligned students?



Multi-Layer case: Alignment could happen!



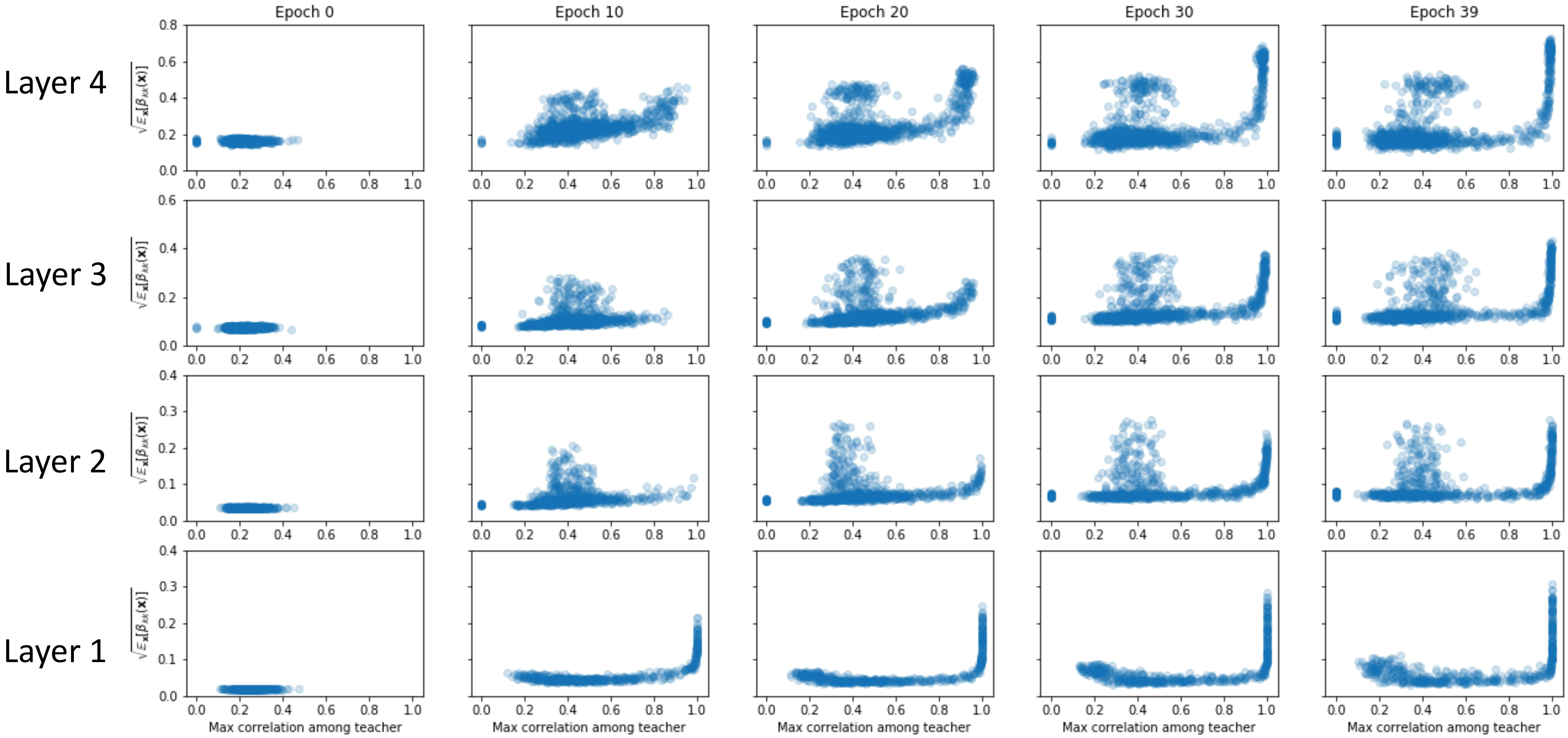
$$\boldsymbol{\alpha}_k^\intercal(\mathbf{x})\mathbf{f}^*(\mathbf{x}) - \boldsymbol{\beta}_k^\intercal(\mathbf{x})\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

Piece wise constant, apply the same logic **per region!**

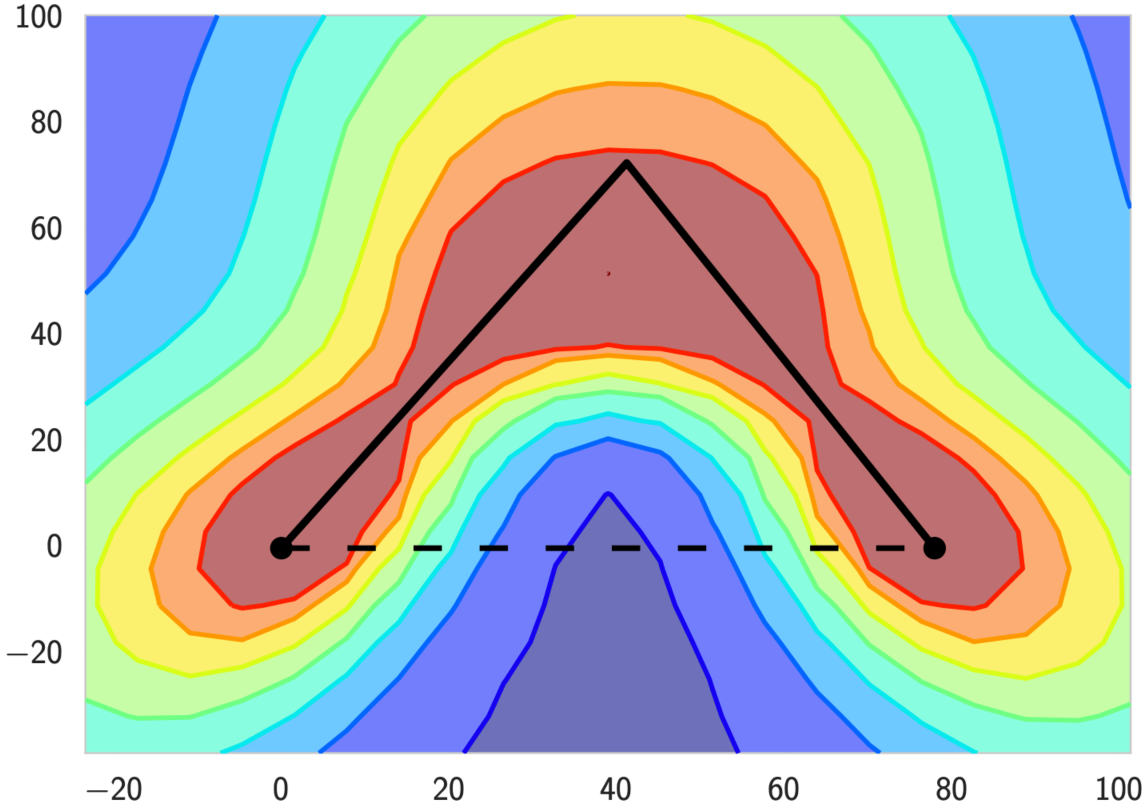
For 2-layer:

$$\sqrt{\mathbb{E}_{\mathbf{x}}\left[\beta_{kk}(\mathbf{x})\right]} = \|\mathbf{v}_k\|$$

Training Progresses



Solutions can be connected by line segments

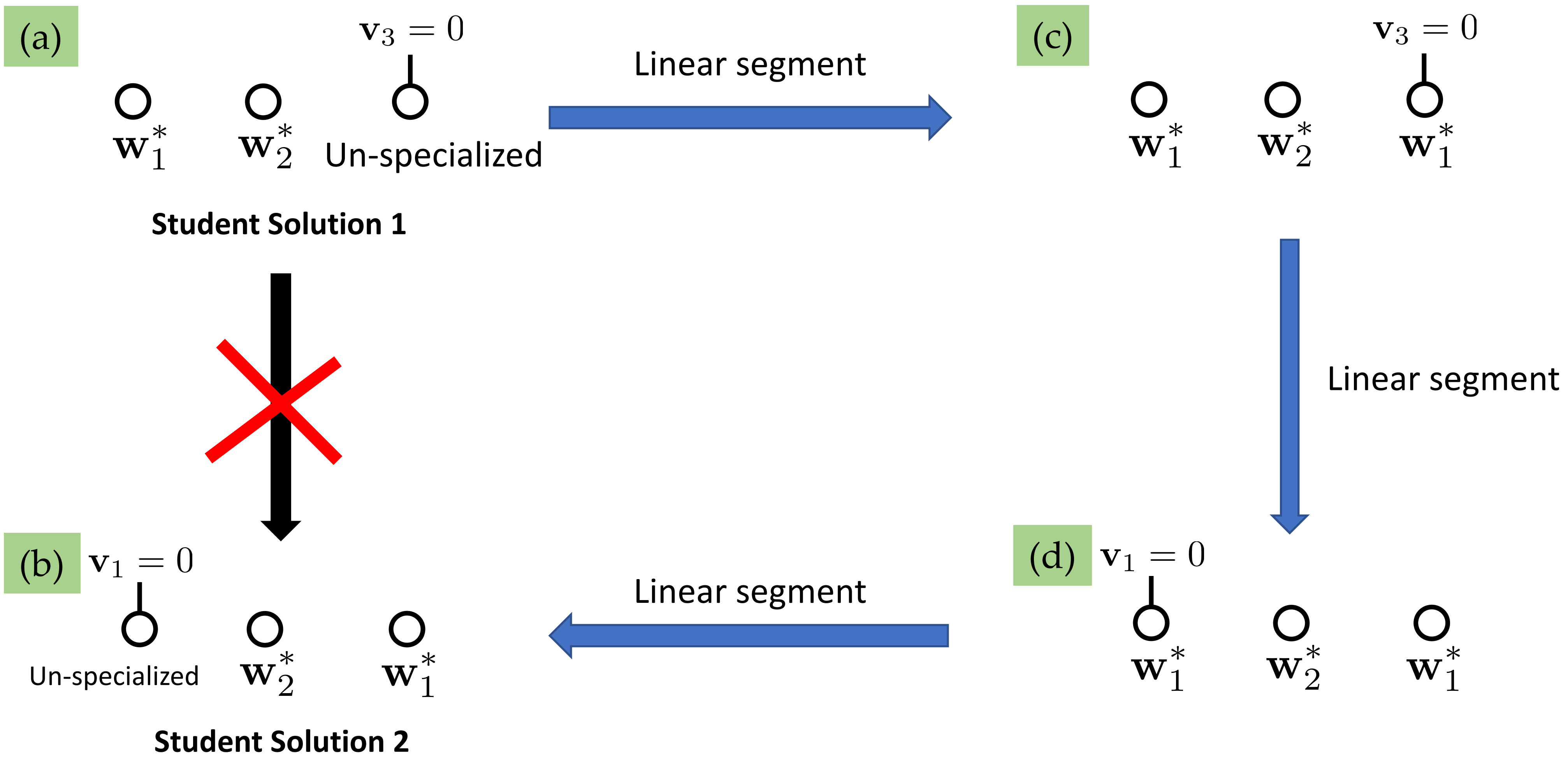


[Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, Garipov et al. NeurIPS 2018]

[Essentially No Barriers in Neural Network Energy Landscape, Draxler et al, 2018]

[Explaining Landscape Connectivity of Low-cost Solutions for Multilayer Nets, Kuditipudi et al, 2019]

Our Explanation



Realistic Case

Student Specialization with **2-layers** ReLU nets, **Small** Gradient and **Finite** Samples

Polynomial Complexity for 2-layered Network

To achieve ϵ-alignment between a teacher *j* and student *k*

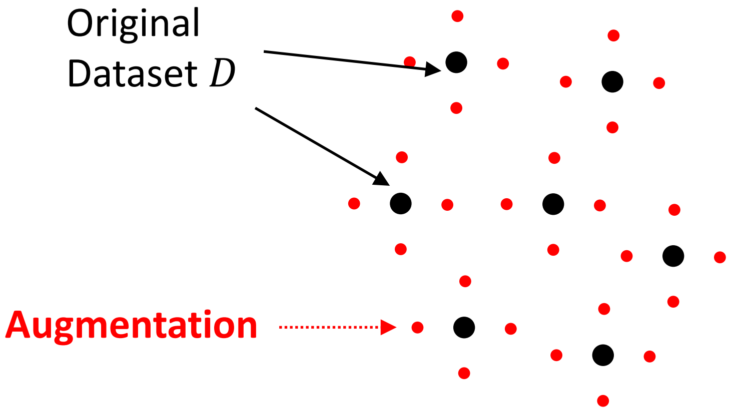
$$K_1 = m_1 + n_1$$

Small gradient

$$\|\mathbf{g}_1(\mathbf{x}, \hat{\mathcal{W}})\|_\infty \leq \frac{\alpha_{kj}}{5K_1^{3/2}\sqrt{d}}\epsilon,\ \mathbf{x} \in D'$$

Sample Complexity of original Dataset D

$$N = \mathcal{O}(K_1^{5/2} d^2 \epsilon^{-1} \kappa^{-1})$$

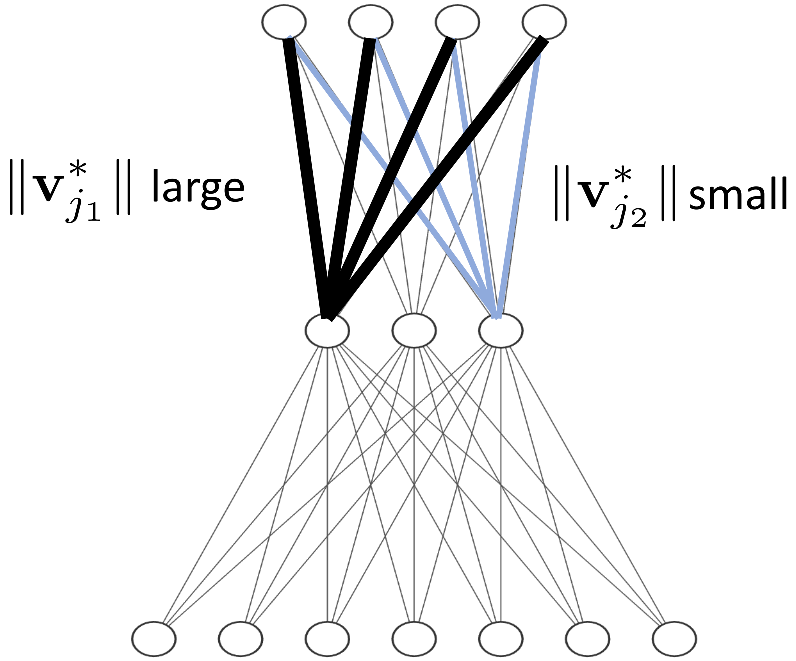


Teacher-agnostic augmentation

D’ = Aug(D)

|D’| = (2d+1)|D|

Lesson 1: Stronger teacher node learns faster



Gradient Condition:

$$\alpha_{kj} := \boldsymbol{v}_k^{\mathrm{T}} \boldsymbol{v}_j^*$$

$$\|\mathbf{g}_1(\mathbf{x}, \hat{\mathcal{W}})\|_\infty \leq \frac{\boxed{\alpha_{kj}}}{5K_1^{3/2}\sqrt{d}}\epsilon, \ \mathbf{x} \in D',$$

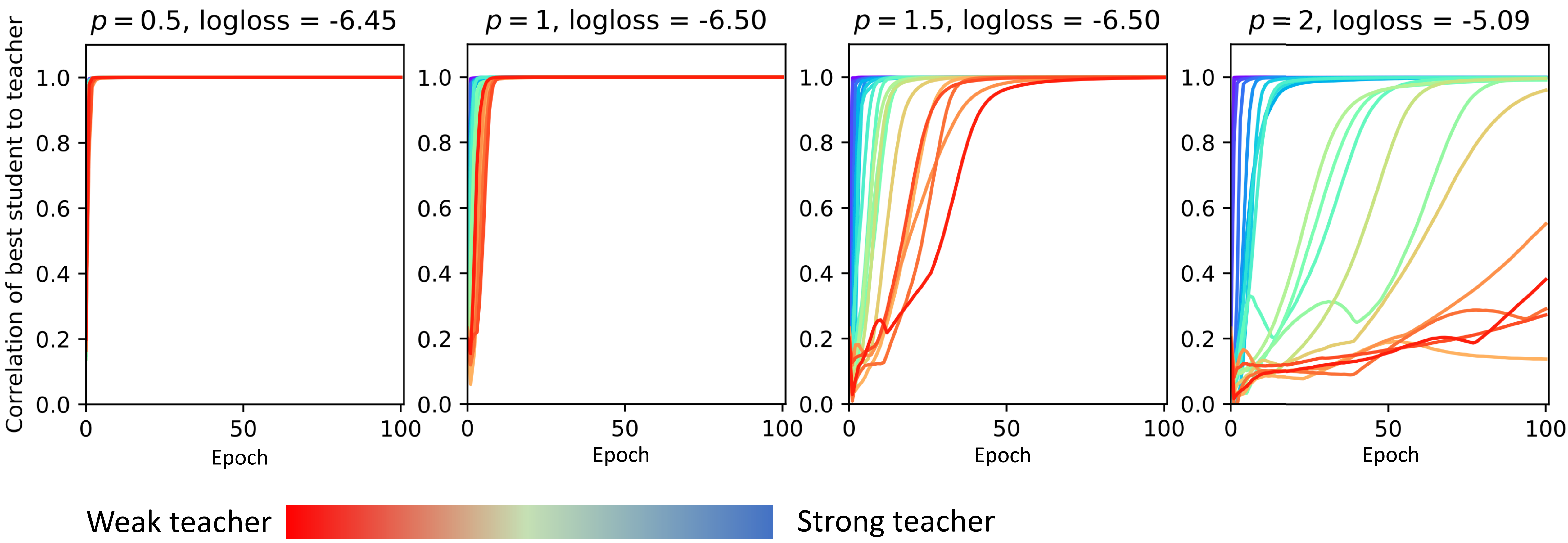
Strong teacher nodes are learned faster

1. Robust to Noise! 😃
2. Hard to learn weak teacher nodes 😢

Weak teacher nodes are slow to learn

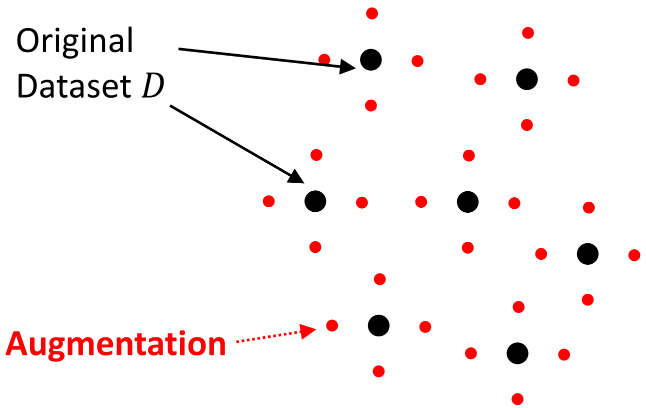
Teacher *j*:

$$\|\mathbf{v}_j^*\| \propto 1/j^p$$

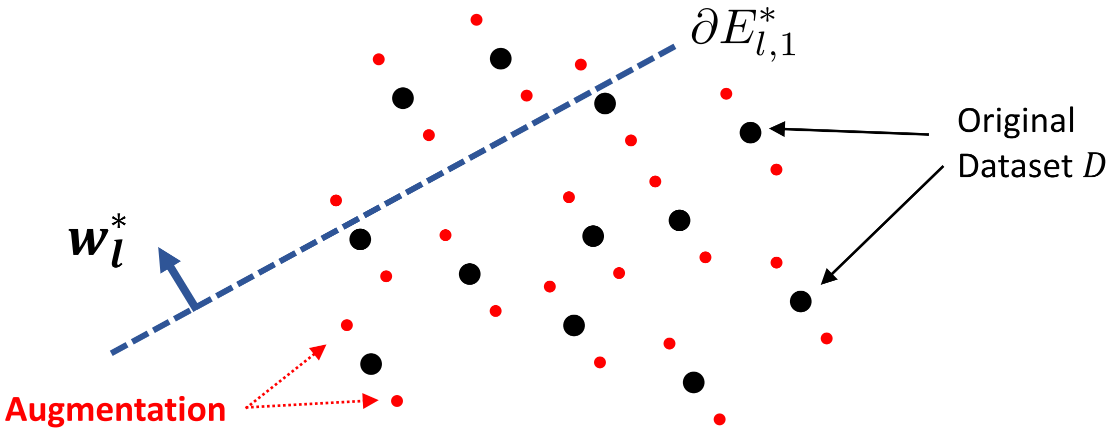


Lesson 2: Data augmentation Matters

Teacher-agnostic augmentation



Teacher-aware augmentation



Small gradient

$$\|\mathbf{g}_1(\mathbf{x}, \hat{\mathcal{W}})\|_\infty \le \frac{\alpha_{kj}}{5K_1^{3/2}\sqrt{d}}\epsilon$$

$$\|\mathbf{g}_1(\mathbf{x}, \hat{\mathcal{W}})\|_\infty \le \frac{\alpha_{kj}}{5K_1^{3/2}}\epsilon$$

No $\sqrt{d}$

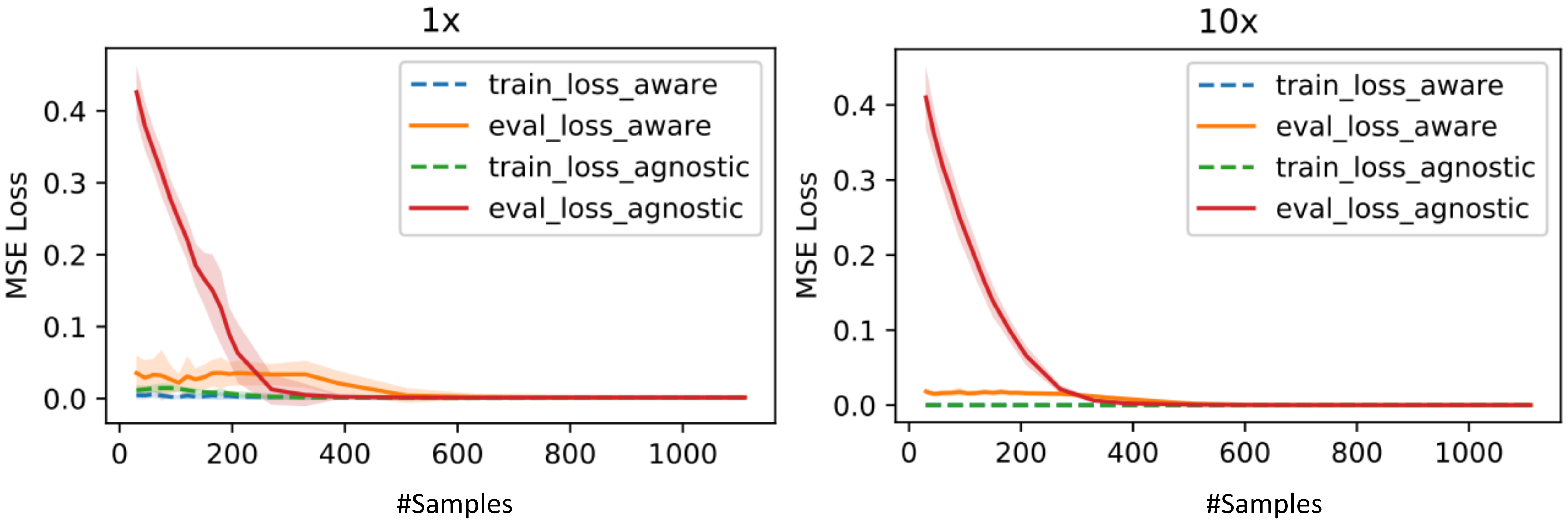
Sample Complexity of original Dataset D

$$N = \mathcal{O}(K_1^{5/2} d^2 \epsilon^{-1} \kappa^{-1})$$

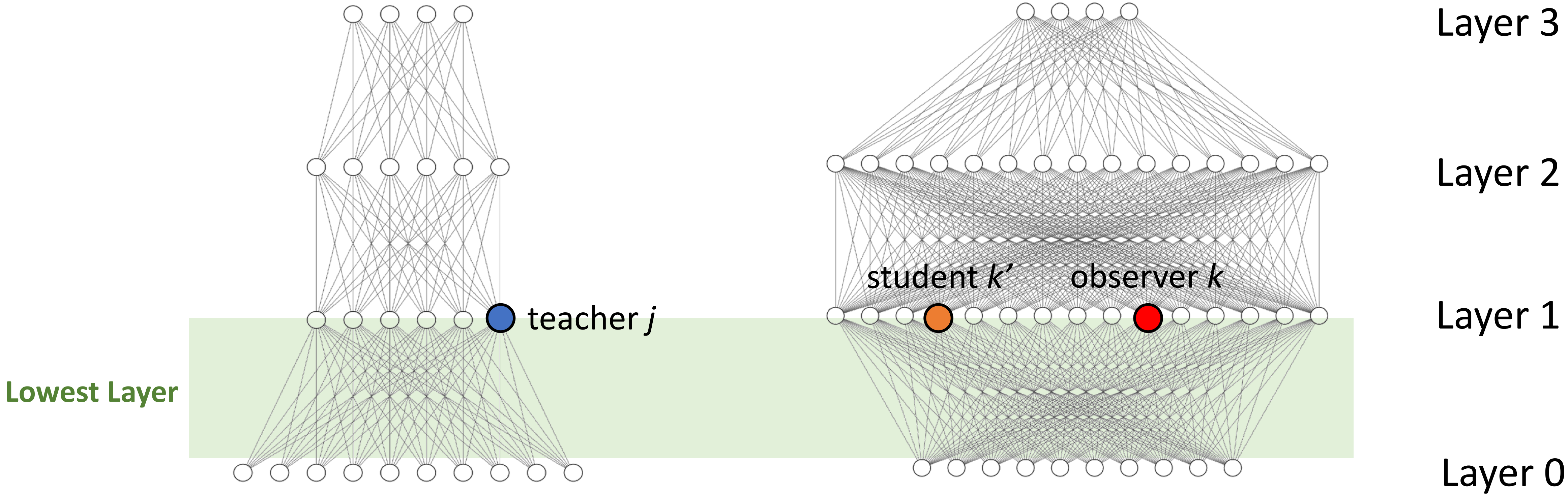
$$\mathcal{O}(K_1^{5/2} d \epsilon^{-1} \kappa^{-1})$$

Linear w.r.t d

Teacher-Agnostic versus Teacher-aware



Multi-layer case



Small gradient

$$\|\mathbf{g}_1(\mathbf{x}, \hat{\mathcal{W}})\|_\infty \leq \frac{\min_{R\in\mathcal{R}} \alpha_{kj}(R)}{5Q^{3/2}\sqrt{d}}\epsilon, \textit{ for } \mathbf{x} \in D'$$

Sample Complexity of original Dataset D

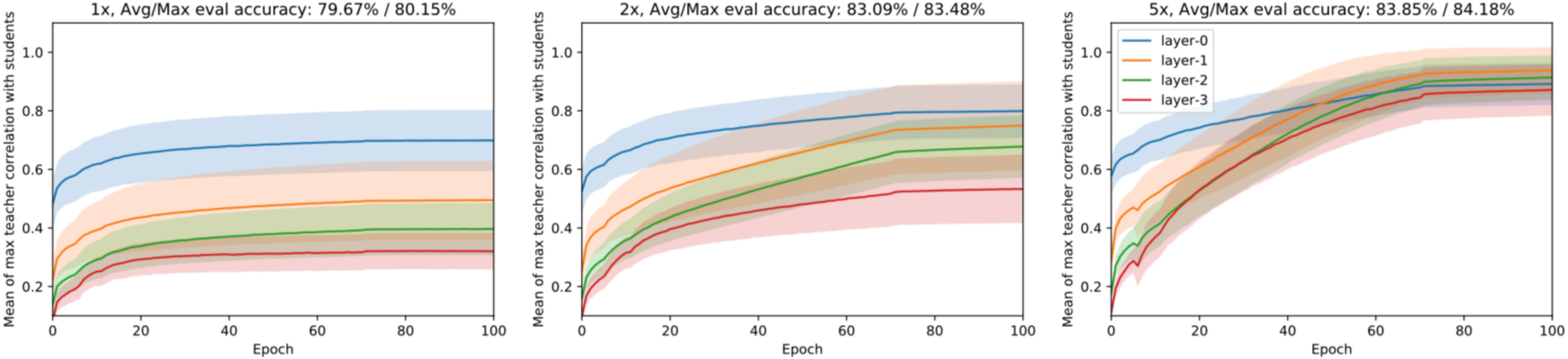
$$\mathcal{O}(Q^{5/2}d^2\epsilon^{-1}\kappa^{-1})$$

Q: #boundaries of hyperplanes (w.r.t network depth)

CIFAR 10

1. Train a conv teacher network of size 64-64-64-64.
2. **[Construct Oracle]** Prune the teacher network to [45-32-32-20]
3. Then train a student network to mimic teacher‘s output (before softmax)

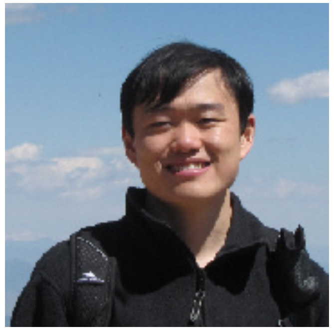
The student network has more parameters

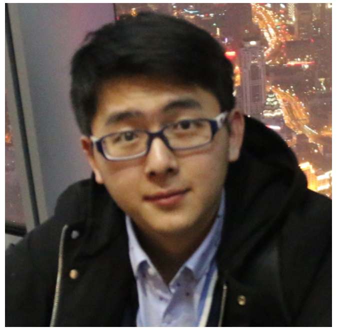


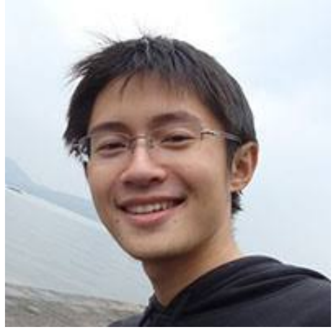
A theoretical framework that explains

1. Why **self-supervised learning** with deep ReLU models works
2. Why a good representation is learned without supervision
3. Why BYOL doesn't need negative samples

Understanding Self-supervised Learning with Dual Deep Networks


Yuandong Tian

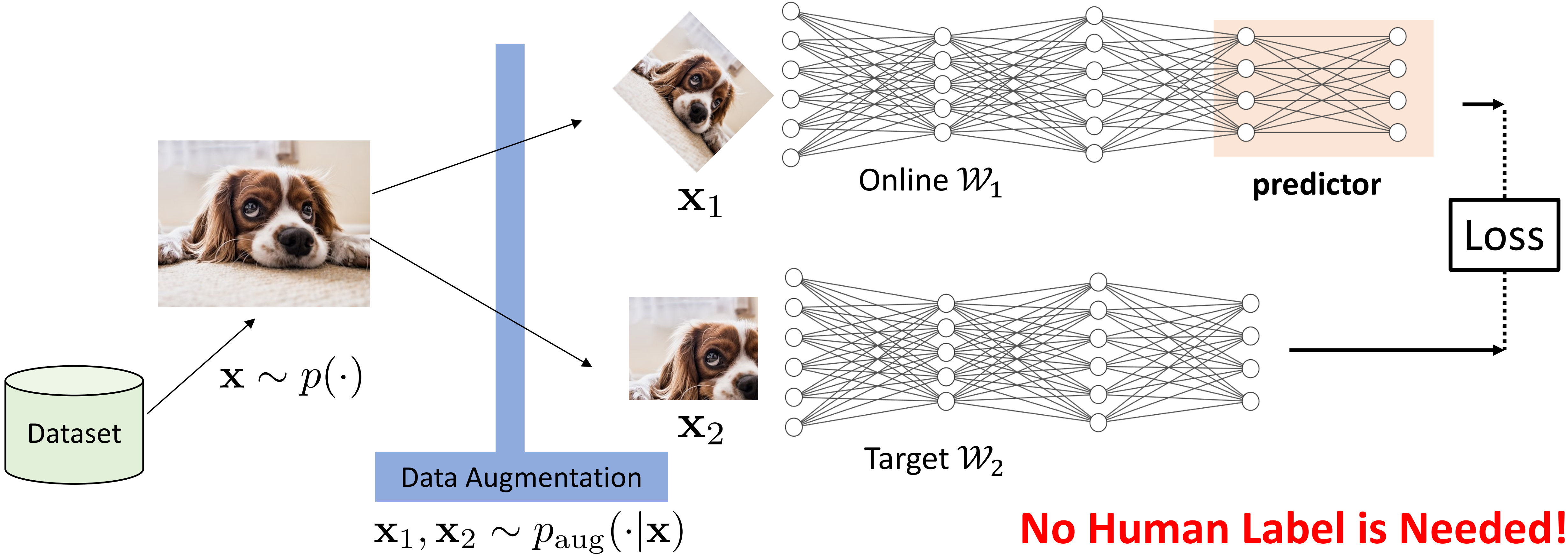

Lantao Yu


Xinlei Chen


Surya Ganguli

Submitted to ICLR 2021

Self-supervised Learning (SSL)

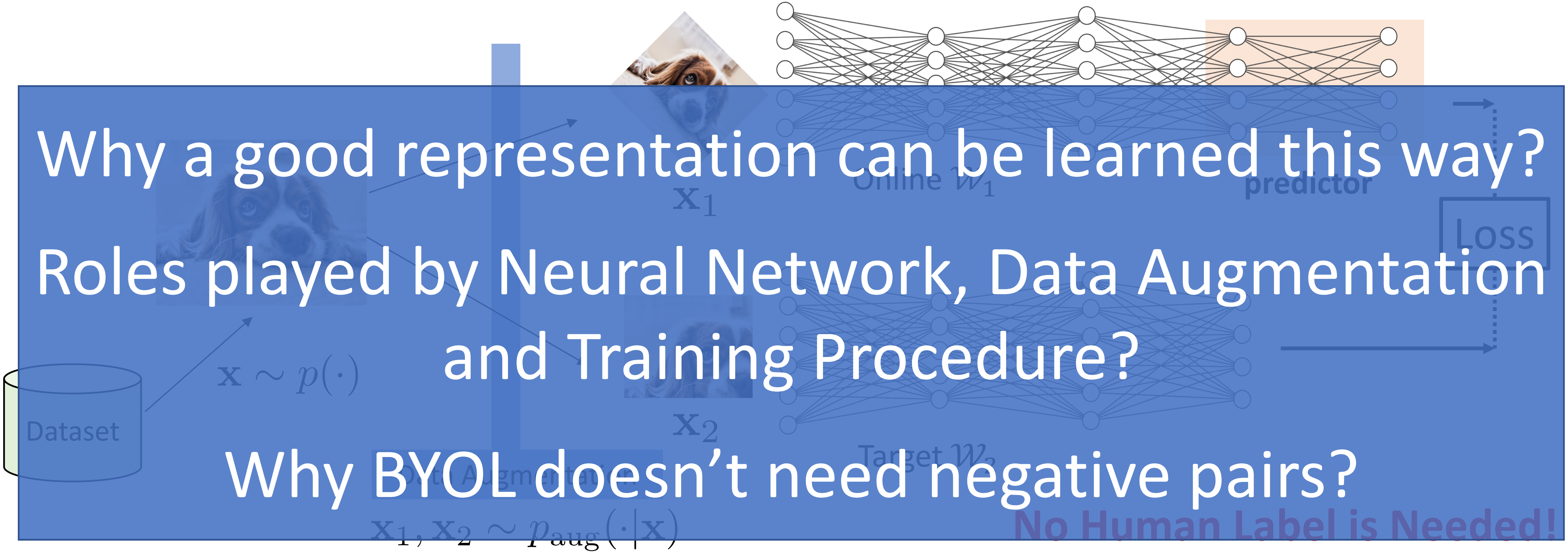


No Human Label is Needed!

BYOL: *[J. Grill, Bootstrap your own latent: A new approach to self-supervised Learning, arXiv 2020]*

SimCLR: *[T. Chen, A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020]*

Self-supervised Learning (SSL)



Why a good representation can be learned this way?

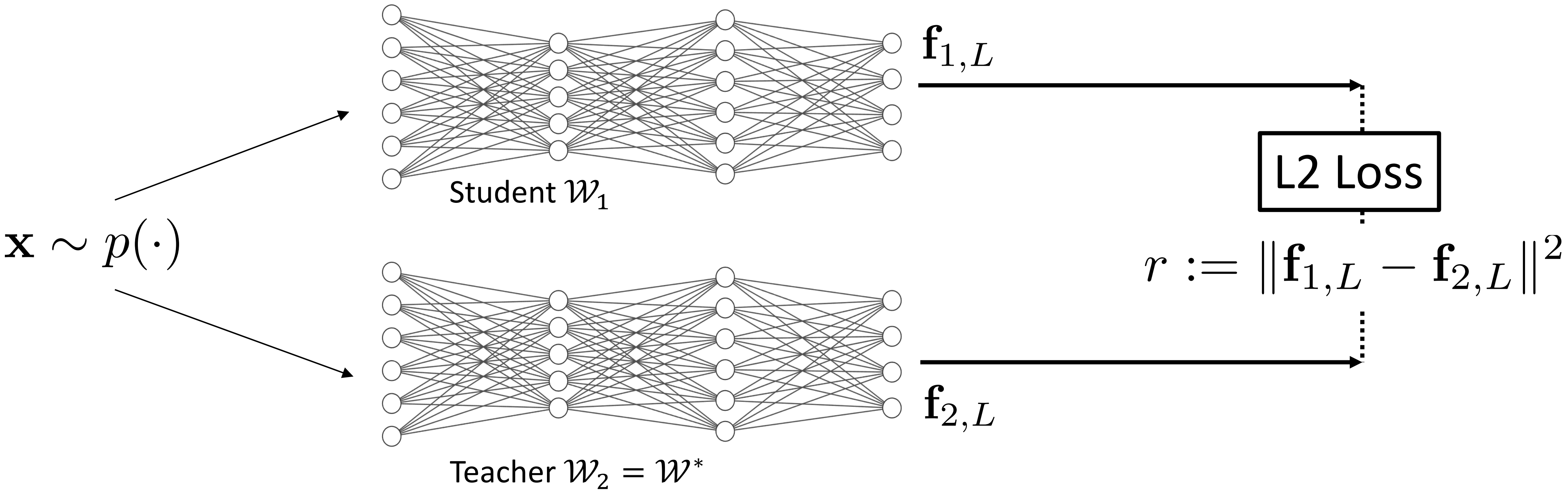
Roles played by Neural Network, Data Augmentation and Training Procedure?

Why BYOL doesn't need negative pairs?

BYOL: *[J. Grill, Bootstrap your own latent: A new approach to self-supervised Learning, arXiv 2020]*

SimCLR: *[T. Chen, A Simple Framework for Contrastive Learning of Visual Representations, ICML 2020]*

Similarity with Teacher Student Setting



The mathematical framework is similar!

[Y. Tian, Student Specialization in Deep ReLU Networks With Finite Width and Input Dimension, ICML 2020]

SimCLR

Data

$\mathbf{x} \sim p(\cdot)$ → $\mathbf{x}_1$, $\mathbf{x}_+$

$\mathbf{x}' \sim p(\cdot)$ → $\mathbf{x}_{k-}$

Different sample from the same distribution

Data Augmentation

Positive pair

$$r_+ := \left\|\mathbf{f}_{1,L} - \mathbf{f}_{+,L}\right\|^2$$

Negative pair

$$r_{k-} := \left\|\mathbf{f}_{1,L} - \mathbf{f}_{k-,L}\right\|^2$$

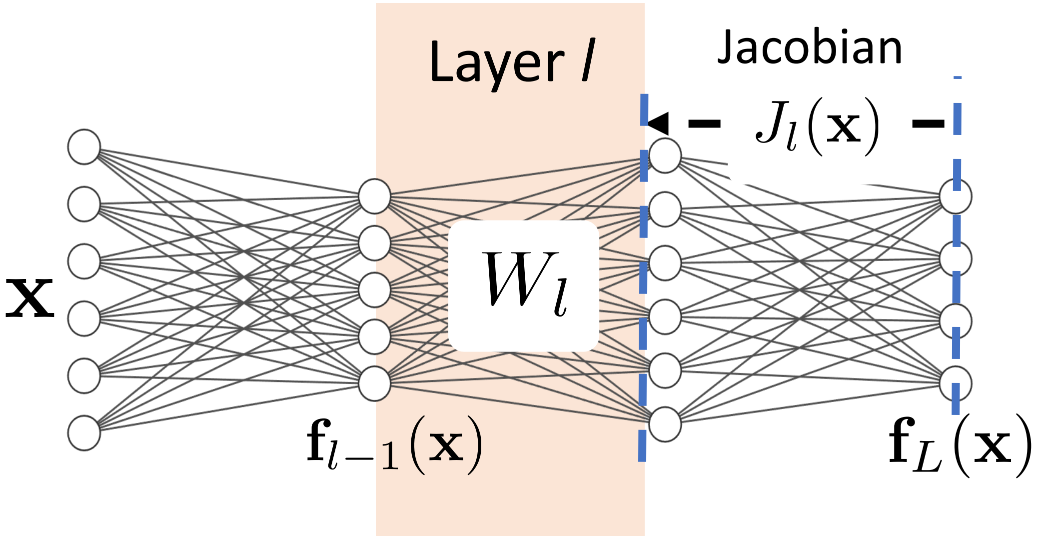
InfoNCE

$$L(r_+, r_{1-}, r_{2-}, \ldots, r_{K-}) \quad := \quad -\log \frac{e^{-r_+/\tau}}{e^{-r_+/\tau} + \sum_{k=1}^{H} e^{-r_{k-}/\tau}}$$

If $|u| = |v| = 1$, then the formulation is the same as SimCLR's formulation

Since $-r = -|u - v|^2 = 2\mathrm{sim}(u, v) - 2$

The Covariance Operator



Weight Update for SimCLR at layer *l*:

$$W_l(t+1) = W_l(t) + \alpha \Delta W_l(t)$$

Learning rate

Connection

$$K_l(\mathbf{x}) := \mathbf{f}_{l-1}(\mathbf{x}) \otimes J_l^{\intercal}(\mathbf{x})$$

Augment-mean Connection

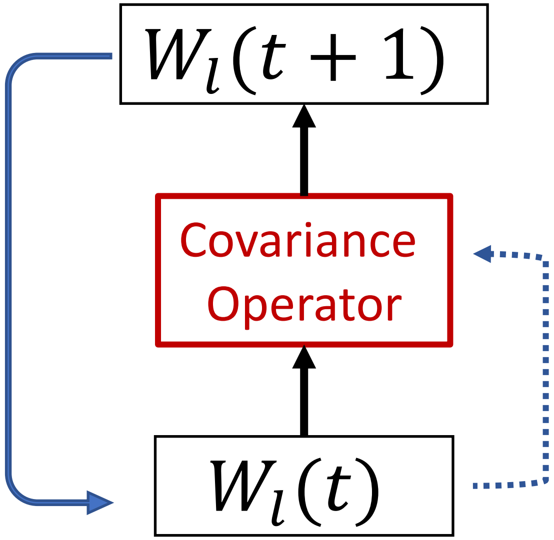
$$\bar{K}_l(\mathbf{x}) := \mathbb{E}_{\mathbf{x}' \sim p_{\mathrm{aug}}(\cdot|\mathbf{x})}[K_l(\mathbf{x}')]$$

Covariance operator (PSD)

$$\mathrm{vec}(\Delta W_l(t)) = \beta \mathbb{V}_{\boldsymbol{x}}[\bar{K}_l(\boldsymbol{x})] \mathrm{vec}(W_l(t))$$

Positive number related to Contrastive loss

What does it mean?



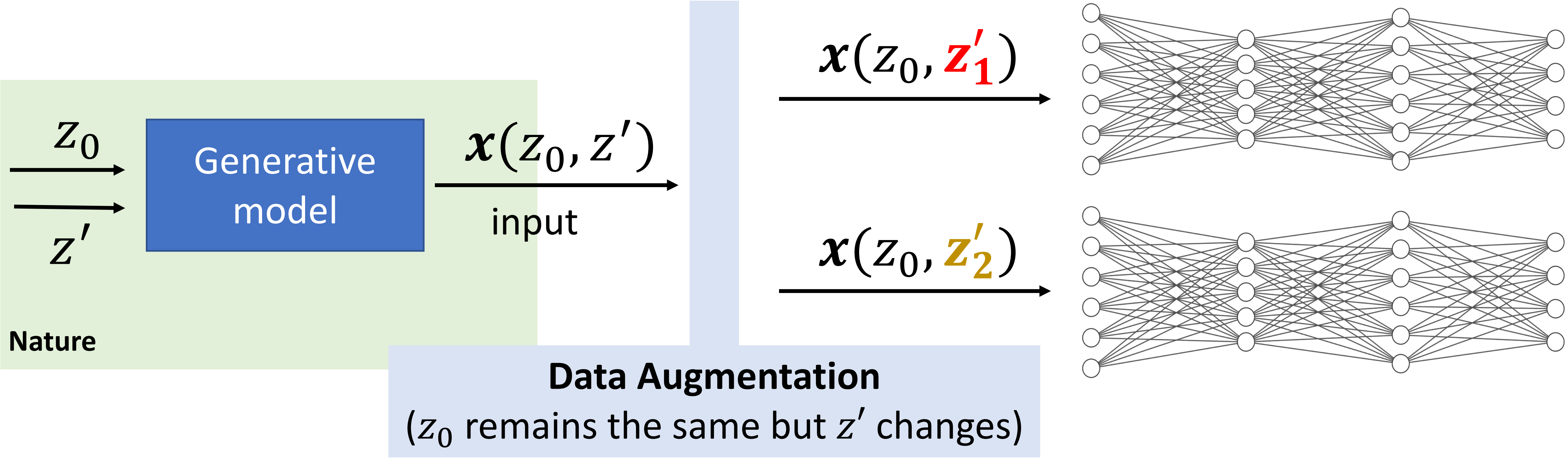
The Covariance Operator $\mathbb{V}_{\mathbf{x}}[\bar{K}_l(\mathbf{x}; \mathcal{W}(t))]$

- Always PSD at any stage of training
- Weight at each layer undergoes a PSD transformation
- Strong eigen mode leads strong weight growth along that direction

What are the strong eigen models in the covariance operator?

To understand that, we need a generative model of the data.

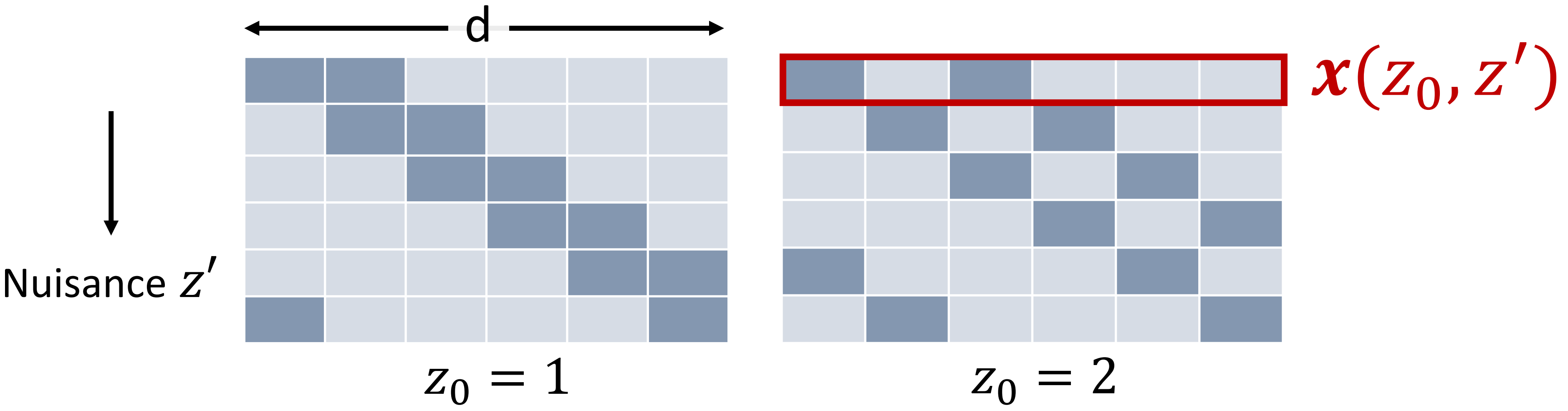
Using Generative Models to understand Covariance Operator



$\boldsymbol{z_0}$: Class (sample) label
$\boldsymbol{z'}$: Nuisance Transformations given by Data Augmentation

One-layer one-neuron example

Two objects **11** and **101** translating in 1D space



$$\boxed{\mathbb{V}_{z_0}\left[\bar{K}(z_0)\right] = \frac{1}{4d^2}\mathbf{u}\mathbf{u}^\intercal}$$

$$\mathbf{u} := \mathbf{x}_{11} + \mathbf{x}_{00} - \mathbf{x}_{01} - \mathbf{x}_{10}$$

Feature to represent pattern 10

Linear neuron: Nothing is learned.

ReLU neuron: Enforce what is initialized!

A two-layer example

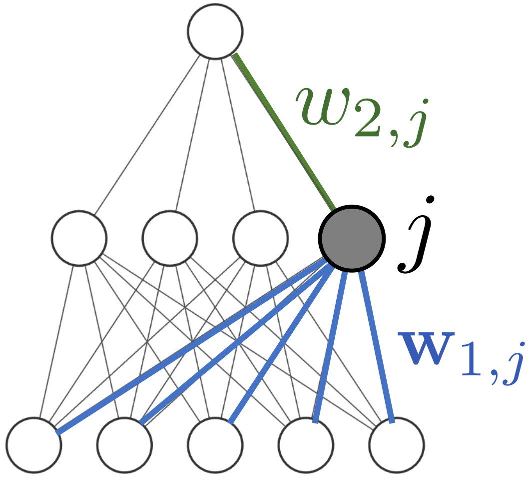
Augment-Average Connection for both layers:

$$\bar{K}_1(z) = [w_{2,1}\mathbf{u}_1, \ldots, w_{2,n_1}\mathbf{u}_{n_1}] \qquad \bar{K}_2(z) = [\mathbf{w}_{1,1}^\intercal\mathbf{u}_1, \ldots, \mathbf{w}_{1,n_1}^\intercal\mathbf{u}_{n_1}]$$

where $\boldsymbol{u}_j(z) := \mathbb{E}_{z'|z}\left[\boldsymbol{x}(z,z')\mathbb{I}(\boldsymbol{w}_{1,j}^\intercal\boldsymbol{x}(z,z') \geq 0)\right]$

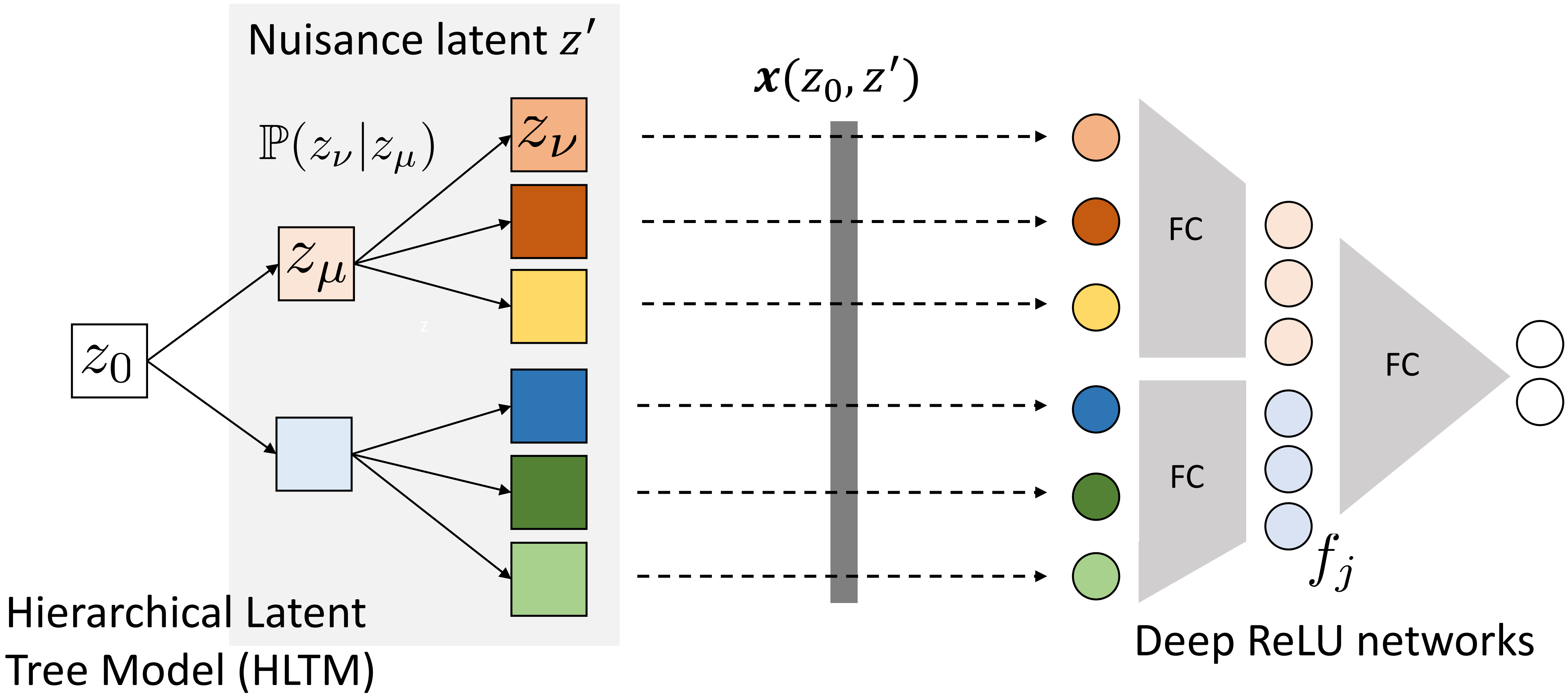
Theorem 4. *If* $\mathrm{Cov}_z[\boldsymbol{u}_j, \boldsymbol{u}_k] = 0$ *for* $j \neq k$, *then the time derivative of* $w_{2,j}$ *and* $\boldsymbol{w}_{1,j}$ *satisfies:*

$$\boxed{\dot{w}_{2,j} = w_{2,j}\boldsymbol{w}_{1,j}^\intercal A_j \boldsymbol{w}_{1,j}, \quad \dot{\boldsymbol{w}}_{1,j} = w_{2,j}^2 A_j \boldsymbol{w}_{1,j},} \quad \textit{where } A_j := \mathbb{V}_z[\boldsymbol{u}_j(z)].$$

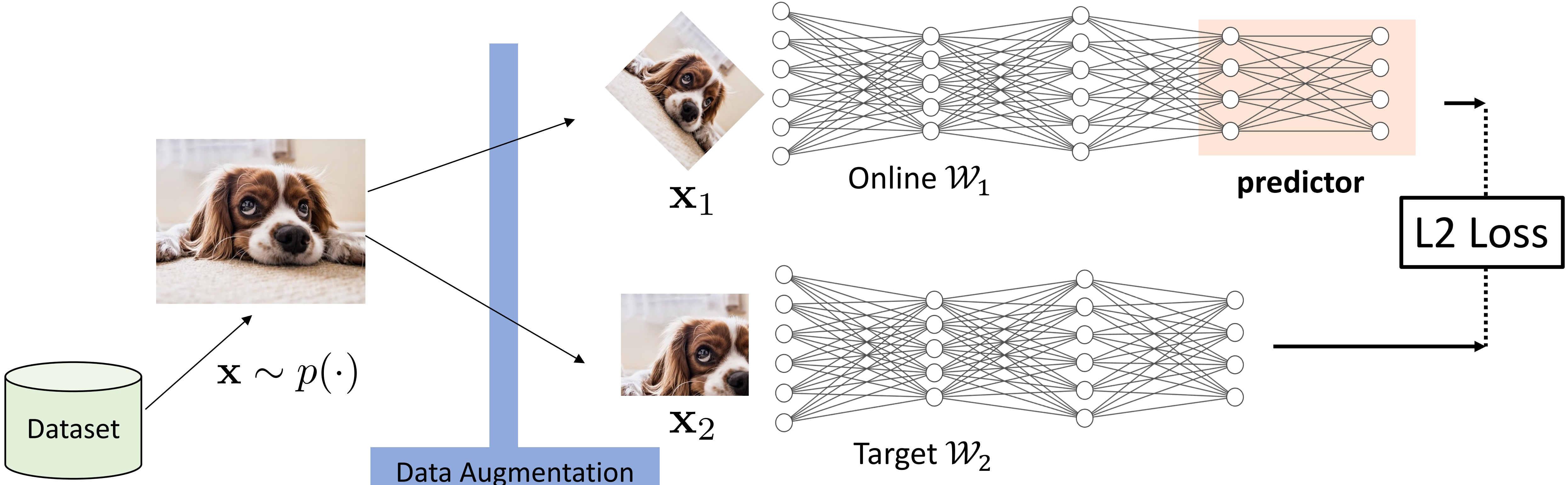


Weights of two layer are ***enforcing*** each other

Hierarchical Latent Tree Models (HLTM)



BYOL Setting

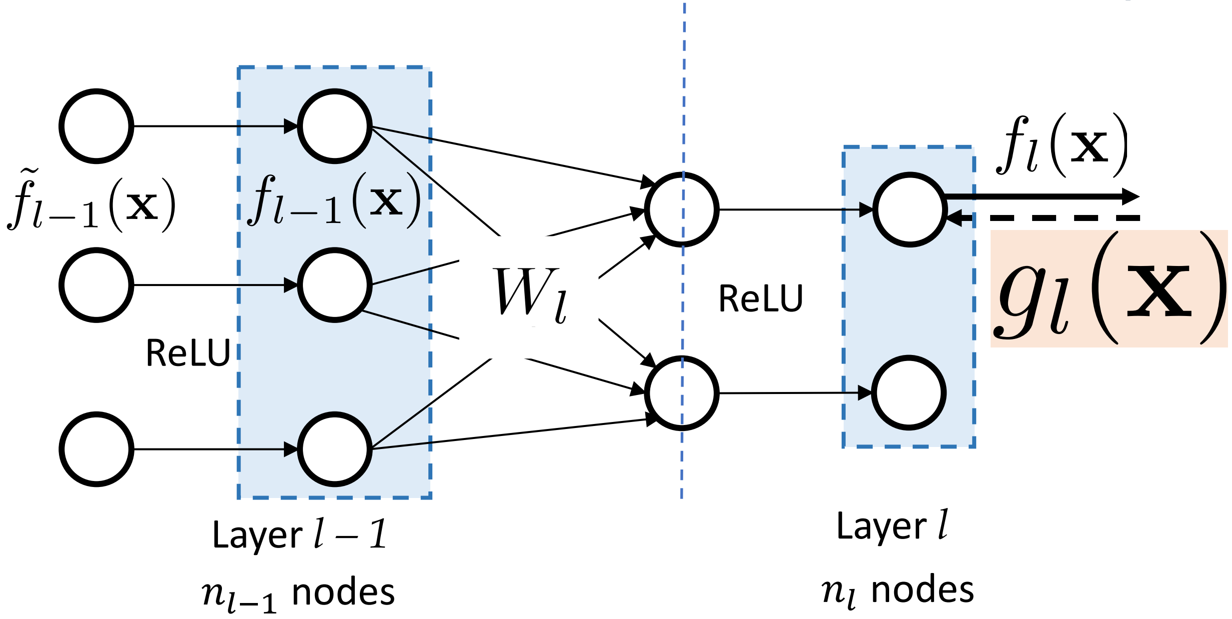


$\mathbf{x}_1, \mathbf{x}_2 \sim p_{\text{aug}}(\cdot|\mathbf{x})$

No Negative Pairs !!!

[J. Grill et al, Bootstrap your own latent: A new approach to self-supervised Learning, arXiv]

Is BatchNorm the Secrete Weapon?



Zero-mean property.

After BN, Backpropagated Gradient is zero-mean in each minibatch:

$$\tilde{\boldsymbol{g}}_l^i := \boldsymbol{g}_l^i - \frac{1}{|B|}\sum_{i\in B}\boldsymbol{g}_l^i = \boldsymbol{g}_l^i - \bar{\boldsymbol{g}}_l$$

With Normalization and $\mathcal{W} \neq \mathcal{W}'$

Without normalization

Correction due to normalization

$$\mathrm{vec}\left(\widetilde{\Delta W_l}\right) = \mathrm{vec}(\Delta W_l) + \mathrm{vec}(\delta W_l^{\mathrm{BN}})$$

$$= \mathrm{vec}(\Delta W_l)_{\mathrm{sym}}$$

Enforce similar representation within data augmentation

$$- \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right] \mathrm{vec}(W_l) + \mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x}; \mathcal{W}')\right] \mathrm{vec}(W_l')$$

$$\mathrm{vec}(\Delta W_l)_{\mathrm{sym}} = -\mathbb{E}_{\boldsymbol{x}}\left\{\mathbb{V}_{\boldsymbol{x}'}[K_l(\boldsymbol{x}')]\right\} \mathrm{vec}(W_l)$$

*Some assumption is need to get to here, see paper for the details.

Why can BYOL work?

If $\mathcal{W}$ has an extra predictor $\left\| -\mathbb{V}_{\mathbf{x}}[\bar{K}_l(\mathbf{x})] \right\| \ll \left\| \mathrm{Cov}_{\mathbf{x}}[\bar{K}_l(\mathbf{x}), \bar{K}_l(\mathbf{x}; \mathcal{W}')] \right\|$

$$
\begin{aligned}
\mathrm{vec}\left(\widetilde{\Delta W_l}\right) &= \mathrm{vec}(\Delta W_l) + \mathrm{vec}(\delta W_l^{\mathrm{BN}}) \\
&= \mathrm{vec}(\Delta W_l)_{\mathrm{sym}} \\
&- \mathbb{V}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x})\right] \mathrm{vec}(W_l) + \mathrm{Cov}_{\boldsymbol{x}}\left[\bar{K}_l(\boldsymbol{x}), \bar{K}_l(\boldsymbol{x}; \mathcal{W}')\right] \mathrm{vec}(W_l')
\end{aligned}
$$

Negated covariance operator
(but second order w.r.t the predictor!!)

Approximate **covariance operator**

*Some assumption is need to get to here, see paper for the details.

BYOL Setting (Top-1 Performance in STL-10)

No predictor, things do not work

-	EMA	BN	EMA, BN
38.7 ± 0.6	39.3 ± 0.9	33.0 ± 0.3	32.8 ± 0.5

EMA is helping

BN is critical

P	P, EMA	P, BN	P, EMA, BN
39.5 ± 3.1	44.4 ± 3.2	63.6 ± 1.06	$\mathbf{78.1} \pm 0.3$

Zero-mean Gradient matters.

Ablation Study of Batch components

-	μ	σ	μ, σ	$\mu^{\nparallel}$
43.9 ± 4.2	64.8 ± 0.6	72.2 ± 0.9	$\mathbf{78.1} \pm 0.3$	44.2 ± 7.0

$\sigma^{\nparallel}$	$\mu^{\nparallel}, \sigma$	$\mu, \sigma^{\nparallel}$	$\mu^{\nparallel}, \sigma^{\nparallel}$
54.2 ± 0.6	48.3 ± 2.7	76.3 ± 0.4	47.0 ± 8.1

μ x = x - x.mean(0)

σ x = x / x.std(0)

$\mu^{\nparallel}$ x = x - x.mean(0).detach()

$\sigma^{\nparallel}$ x = x / x.std(0).detach()

Reinitializing Predictors Works

Table 5: Top-1 performance of BYOL using reinitialization of the predictor every T epochs.

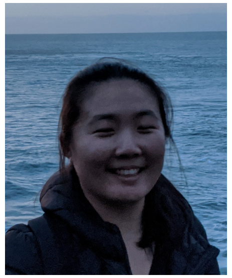
	Original BYOL	ReInit $T = 5$	ReInit $T = 10$	ReInit $T = 20$
STL-10 (100 epochs)	78.1	78.6	**79.1**	79.0
ImageNet (60 epochs)	60.9	61.9	**62.4**	**62.4**

The predictor is not necessarily "optimal" as suggested in the original BYOL paper.

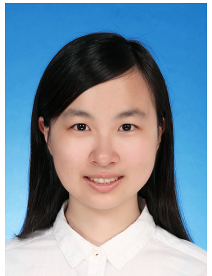
Understanding Adversarial Samples using Teacher-Student

Zhoulin Yang*

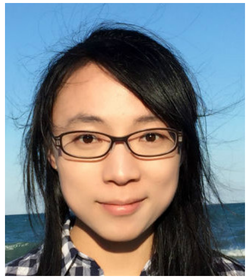
Zhaoxi Chen



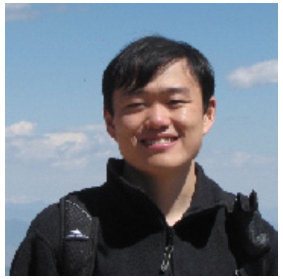
Tiffany Cai



Xinyun Chen



Bo Li

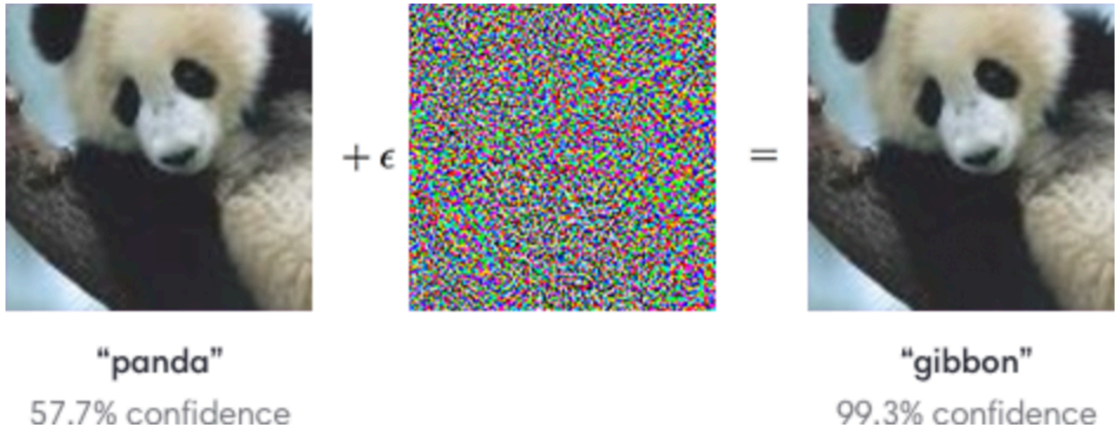


Yuandong Tian *

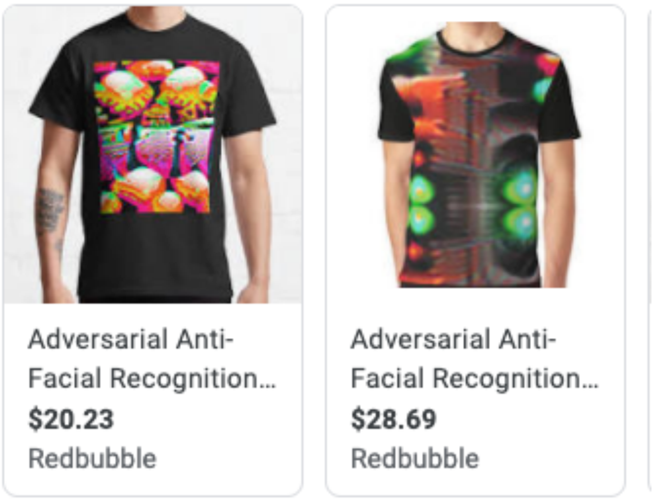
* = Equal Contribution

Submitted to AISTATS 2021

Adversarial Samples



Stop sign → a 45 mph sign



Adversarial Samples: not **Bugs** but **Features**

Training image

Robust Features: dog
Non-Robust Features: dog

Adversarial example towards "cat"

max P(cat)

Relabel as cat

Robust Features: dog
Non-Robust Features: cat

Train

Evaluate on original test set

good accuracy

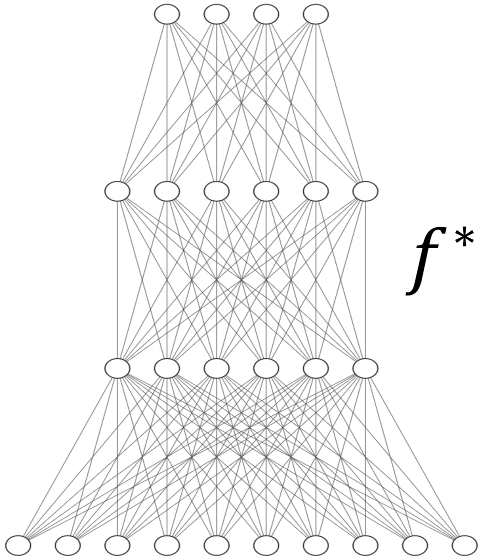
Garbage In → Signal Out 😱

[A. Ilyas, et al, Adversarial Examples Are Not Bugs, They Are Features, 2019]

A Natural Explanation

$y \rightarrow y'$

Original label | Wrong label



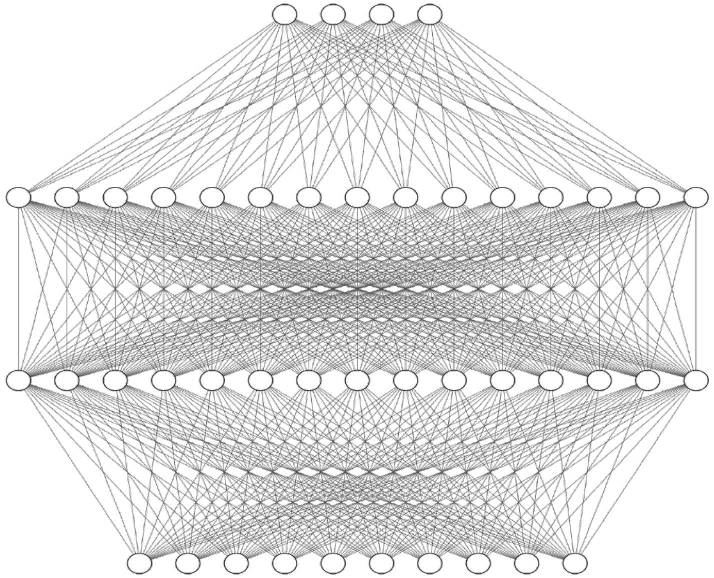
f^*

$x \rightarrow x'$

Original data | Adversarial input

Normal training: train with (x, y)

Fancy training: train with $(x', y' = f^*(x'))$
(wrong sample, correct mapping)

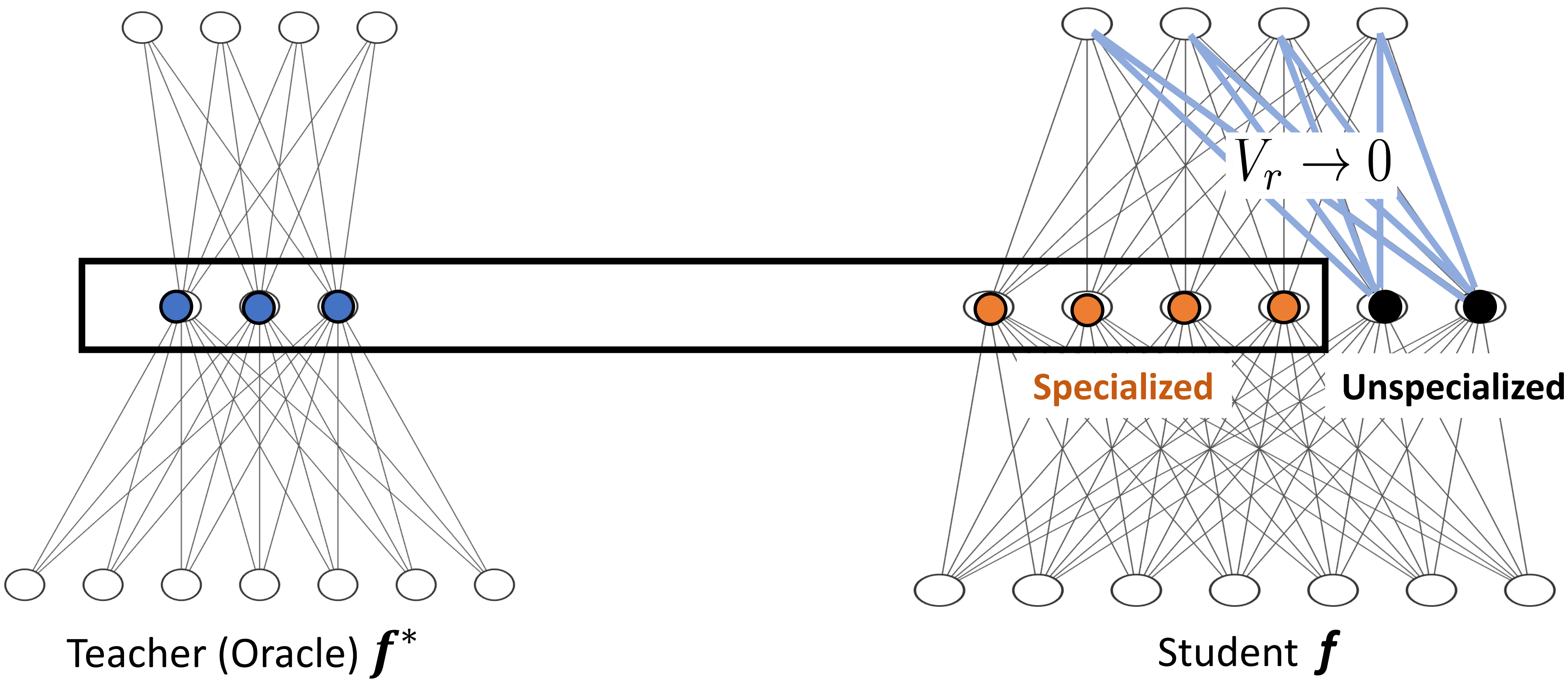


f

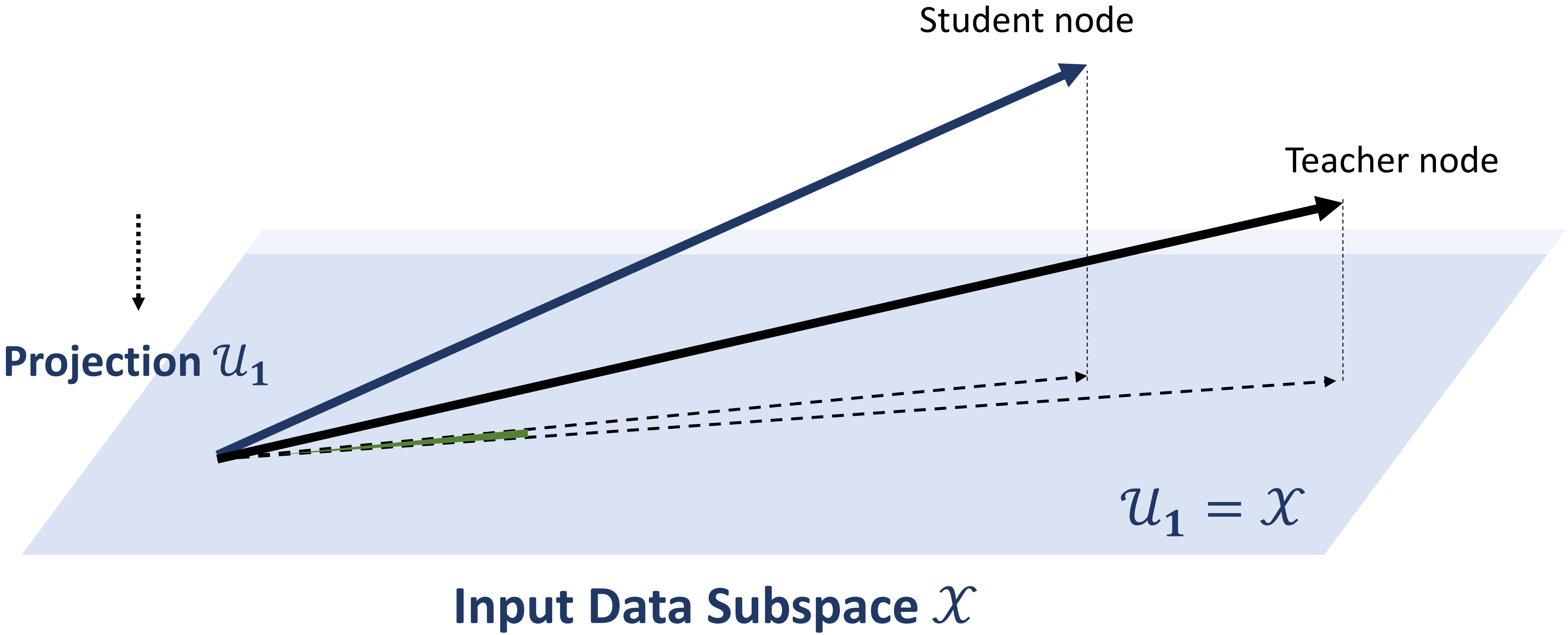
Both are valid pairs from the original network
→ They both train well.

Student Specialization

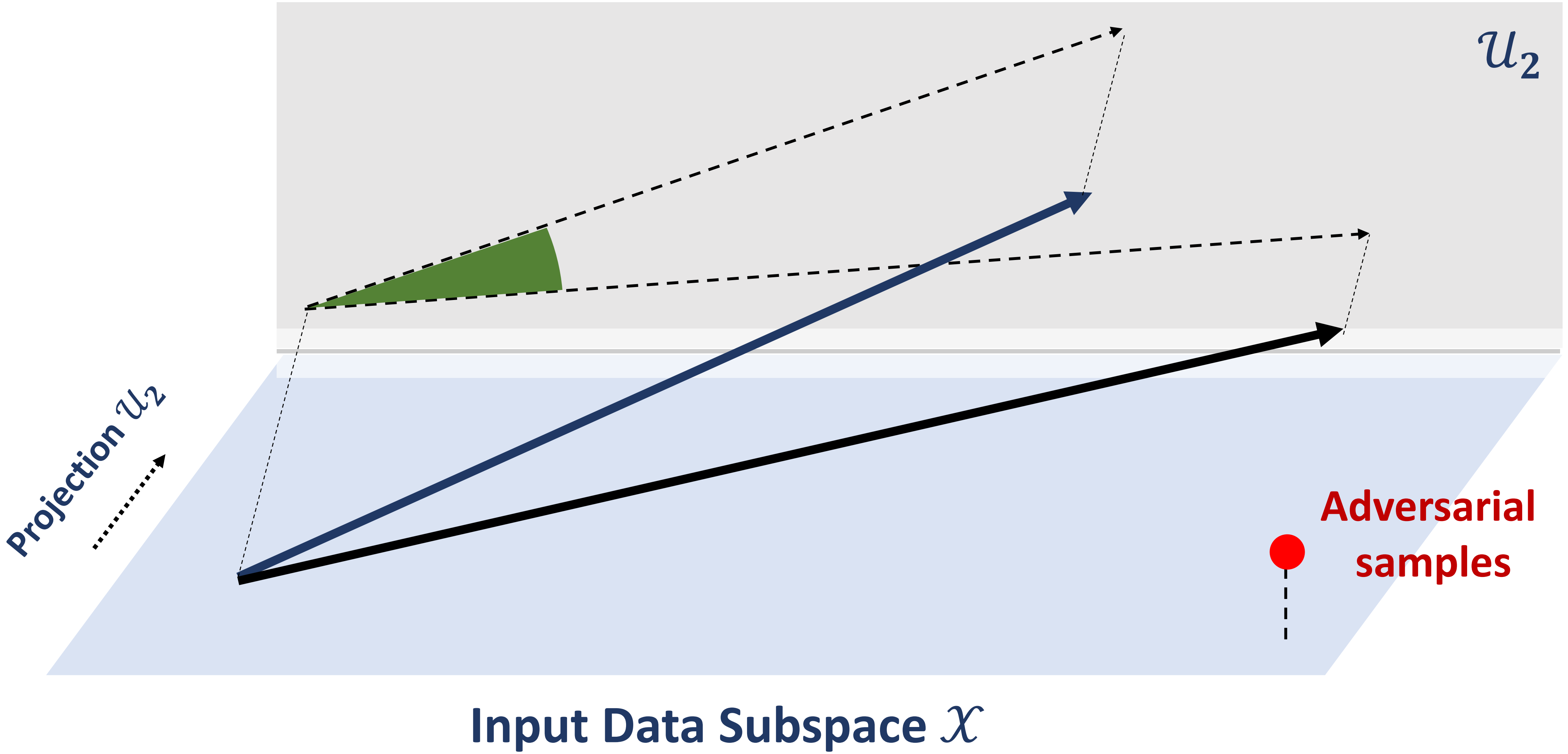
- ***Full rank input space*** → Specialization happens. Contributions of unspecialized nodes are zero.
- ***Low rank input*** → ??



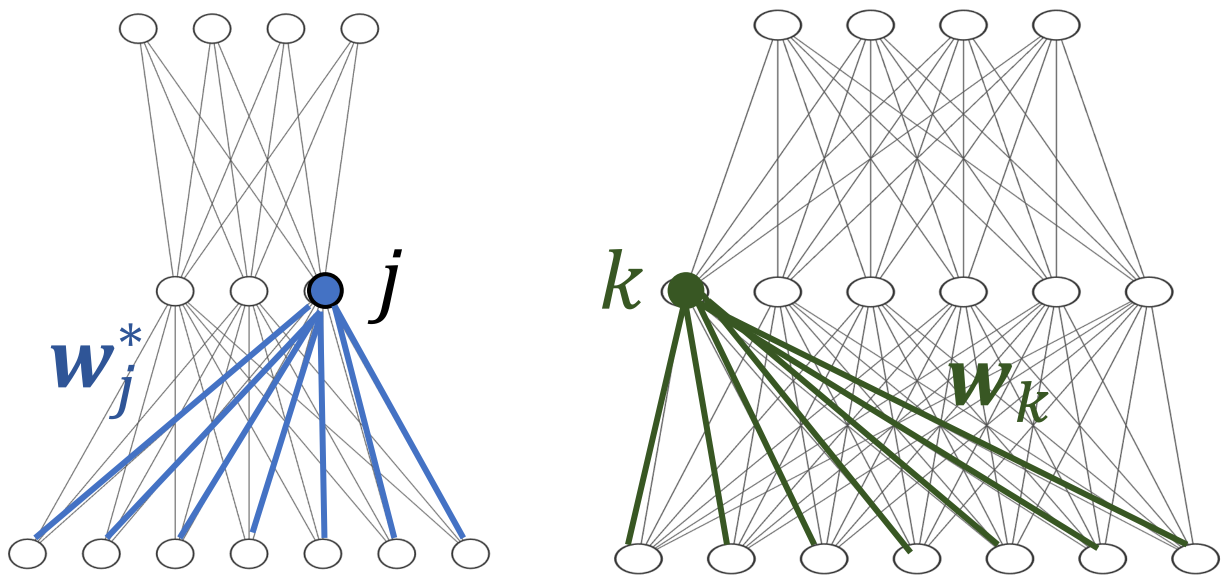
Student Specialization under Low-rank Input

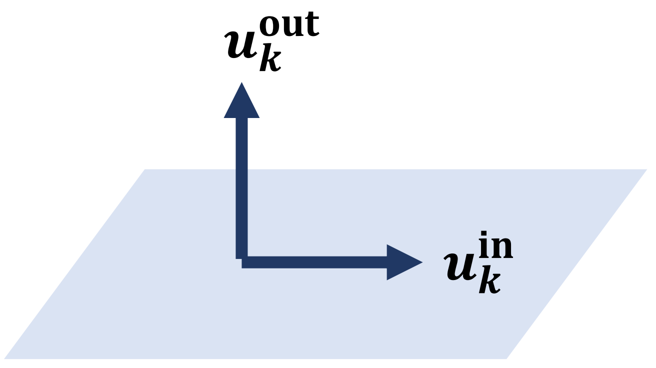


Student Specialization under Low-rank Input



An Empirical Model





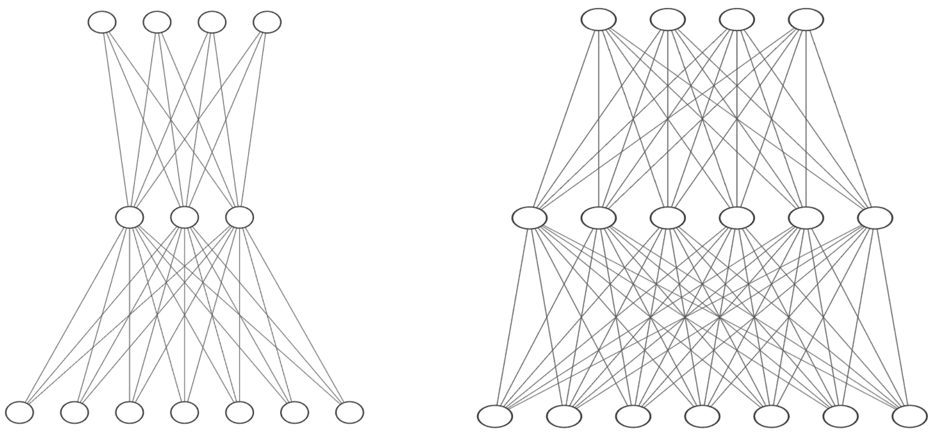
Input Data Subspace $\mathcal{X}$

$$\boldsymbol{w}_k = \boldsymbol{w}_j^* + \epsilon_{\mathrm{in}} \boldsymbol{u}_k^{\mathrm{in}} + \epsilon_{\mathrm{out}} \boldsymbol{u}_k^{\mathrm{out}}$$

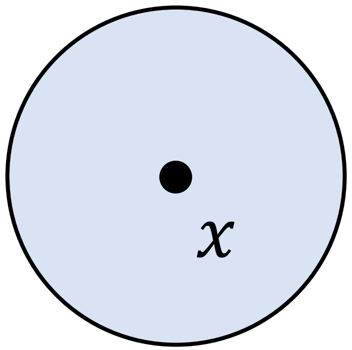
Student node k **Teacher node k**

Will adversarial training improve the robust accuracy by remove the two?

Revise Adversarial Samples



Teacher (Oracle) $\boldsymbol{f}^*$ Student $\boldsymbol{f}$



$B(x, \epsilon)$

Oracle Adversarial

$$\mathbf{x}' = \arg \max_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} L[f(\mathbf{x}'), f^*(\mathbf{x}')]$$

Data Adversarial

Fix target

$$\mathbf{x}' = \arg \max_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} L[f(\mathbf{x}'), \boxed{\mathbf{y}}]$$

Logit Training

$$\min_{\mathrm{w}} L[f(x; w), f^*(x)]$$

Label Training

$$\min_{\mathrm{w}} L[f(x; w), \operatorname{argmax} f^*(x)]$$

Experiments using CIFAR 10 images

Use CIFAR 10 images as low-dimensional and finite sample input

Use a 4-layered teacher network (trained on CIFAR10 and then pruned to be 45−32−32−20) as the *oracle*

Using L2 loss and Oracle Adversarial

$$\mathbf{x}' = \arg\max_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} \|f(\mathbf{x}') - f^*(\mathbf{x}')\|^2$$

Train with pair $\left(x', f^*(x')\right)$. i.e., ground truth label $f^*(x')$ for adversarial sample x'

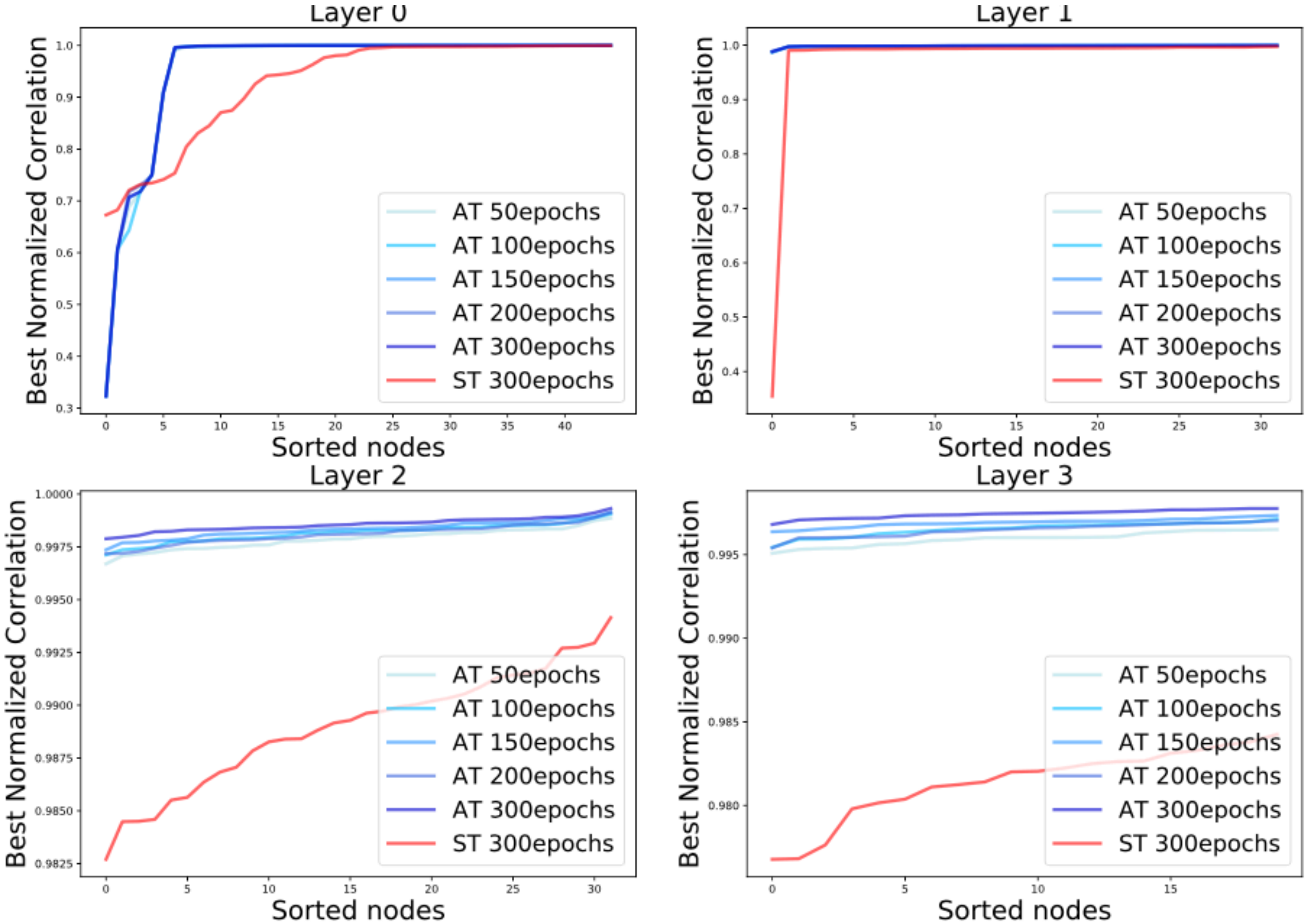
(Use PGD in practice: back-propagate f and f^* to get signed gradient, and apply to x iteratively)

Experiments using CIFAR 10 images

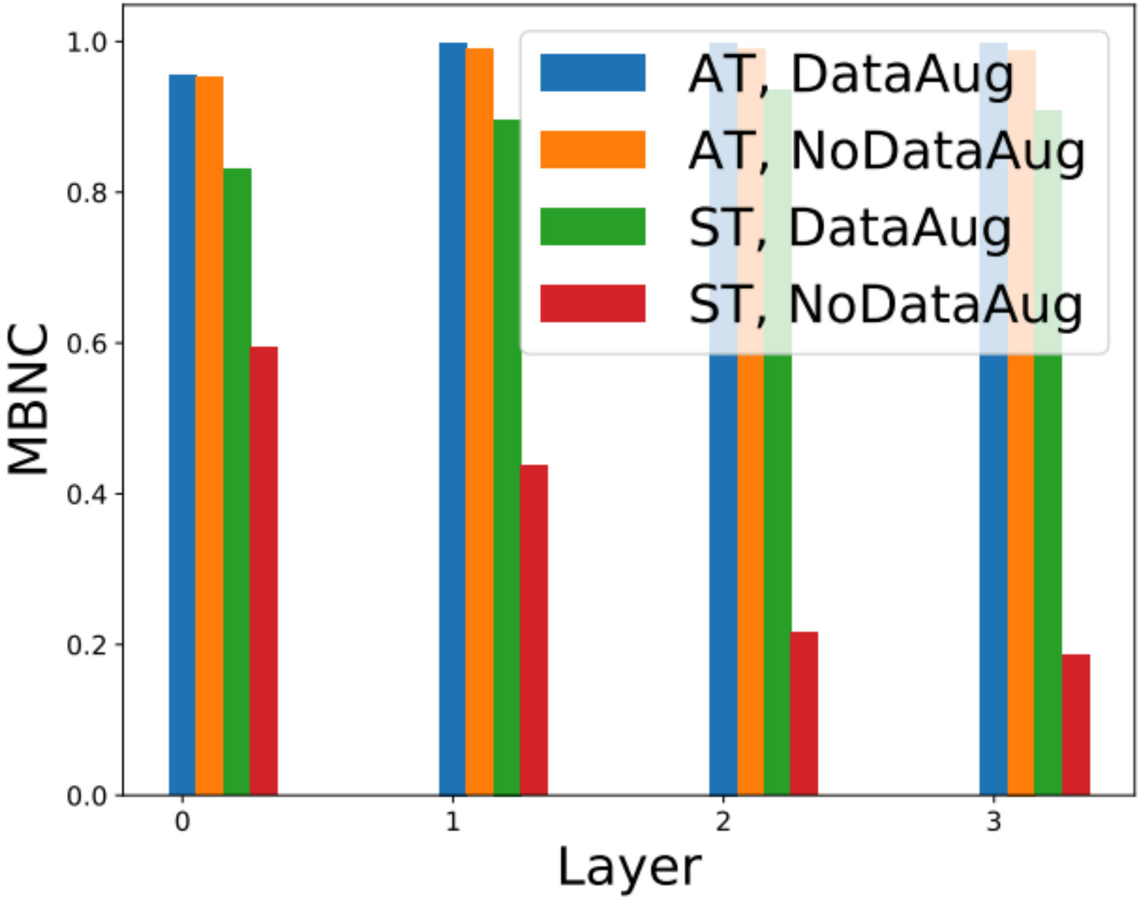
ρ_{jk}: Normalized correlation between j and k:

Best NC(j) = $\max_{student\ k} \rho_{jk}$

Sorted Best NC curve



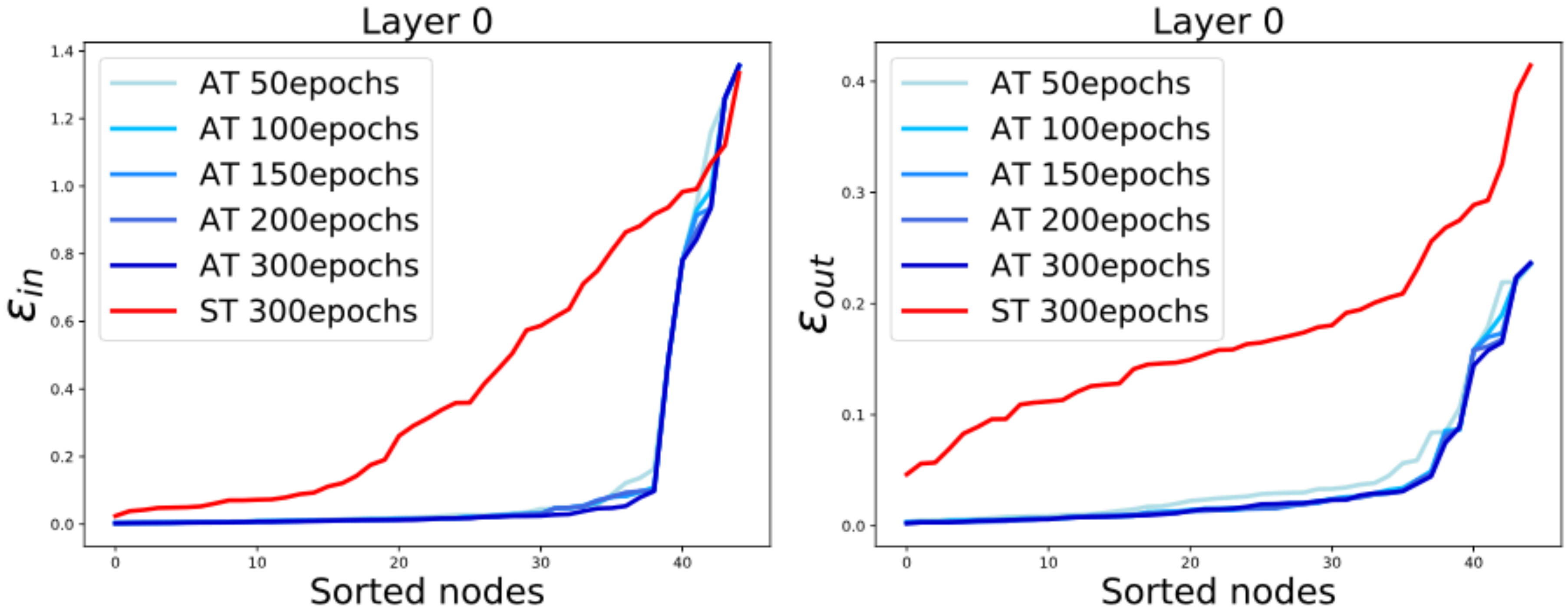
Data Augmentation matters



With data augmentation,

we get better performance

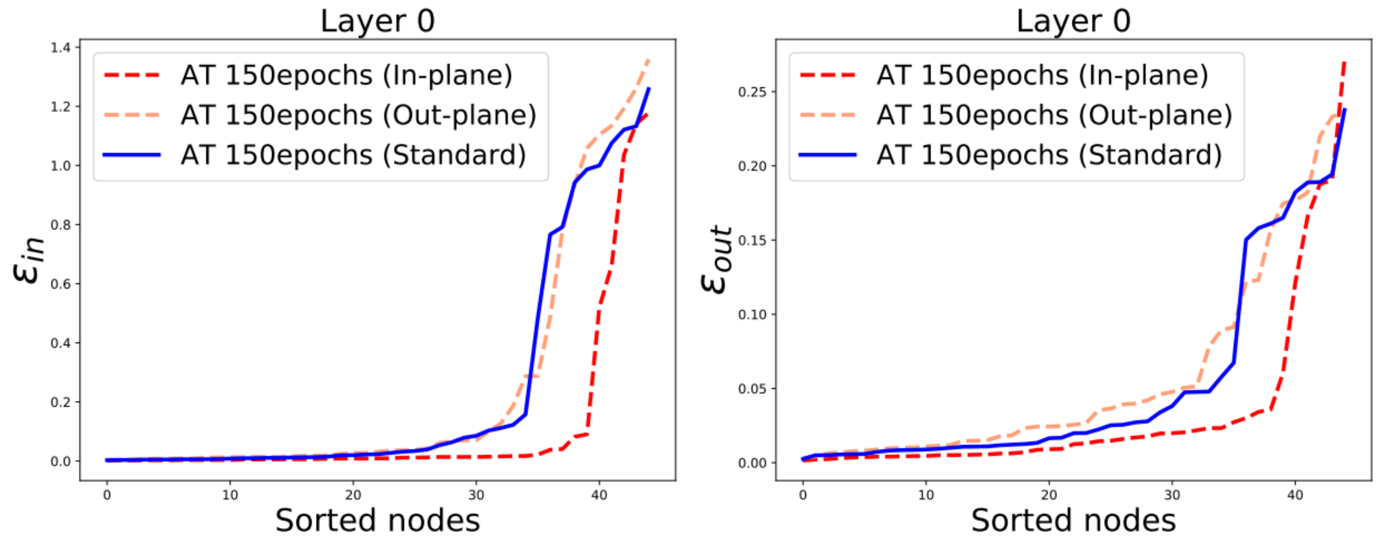
ϵ_{in} and ϵ_{out}



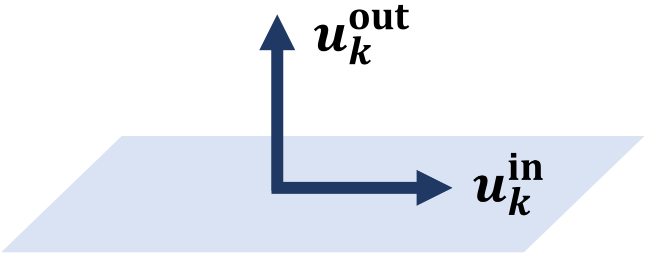
Model	AT (50)	AT (100)	AT (150)	AT (200)	AT (300)	ST (300)
Robust Acc	81.18%	82.57%	84.07%	85.35%	86.81%	62.77%

Robustness is consistent with student specialization.

Training with ϵ_{in} and ϵ_{out} only adversarial samples



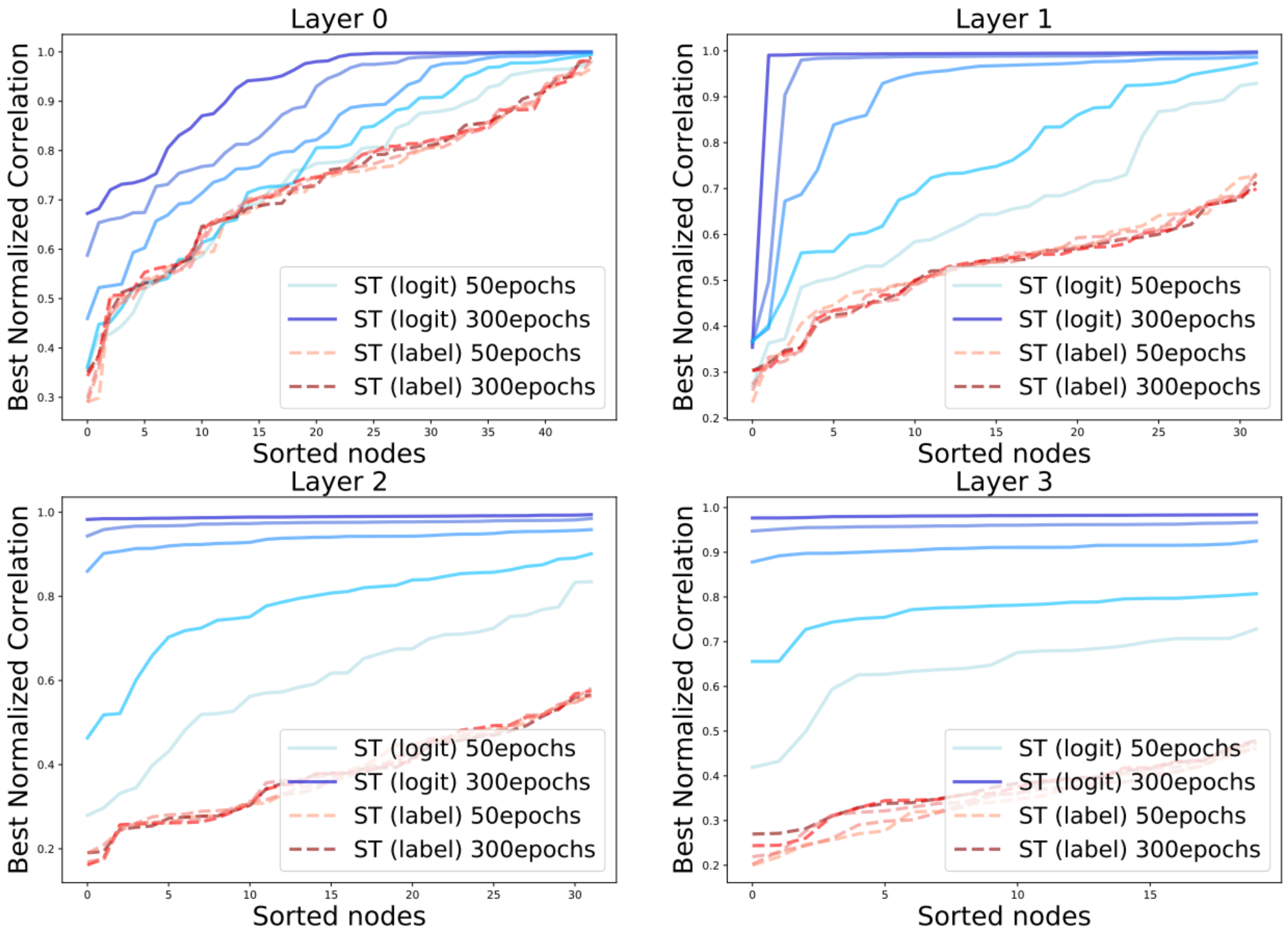
Attacks	In-plane	Out-plane	Standard
AT (In-plane)	88.86%	89.18%	89.28%
AT (Out-plane)	83.11%	83.54%	83.60%
AT (Standard)	86.87%	87.28%	87.18%



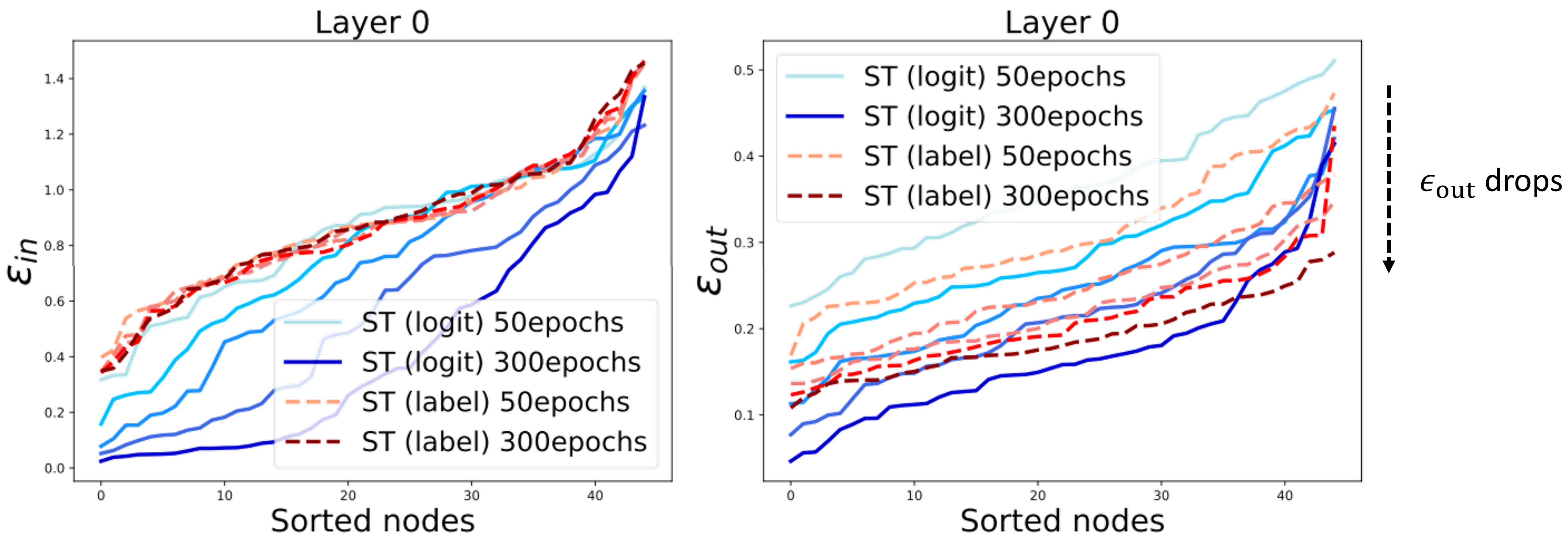
Input Data Subspace $\mathcal{X}$

ϵ_{in} and ϵ_{out} is only controlled in the first layer.

Huge Difference between logits and label training



Label training helps in ϵ_{out} but not ϵ_{in}

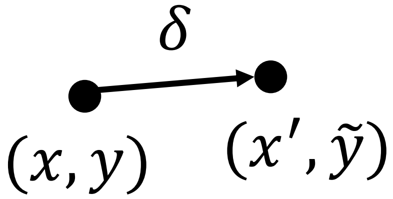


Robust Acc	ST (50)	ST (100)	ST (150)	ST (200)	ST (300)
Logit training	23.12%	30.72%	36.72%	48.52%	62.77%
Label training	19.08%	20.81%	22.34%	23.42%	25.79%

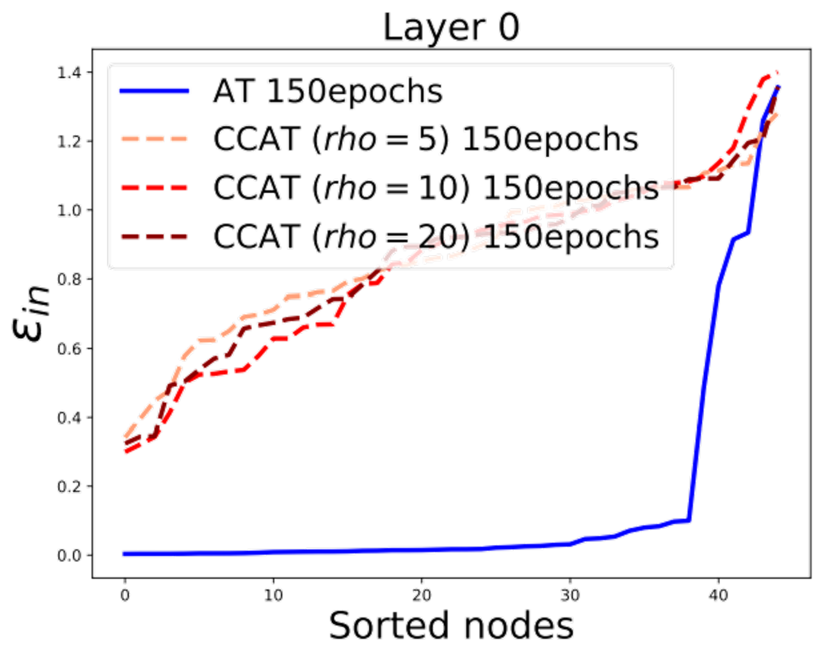
Training with CCAT (AT without true label)

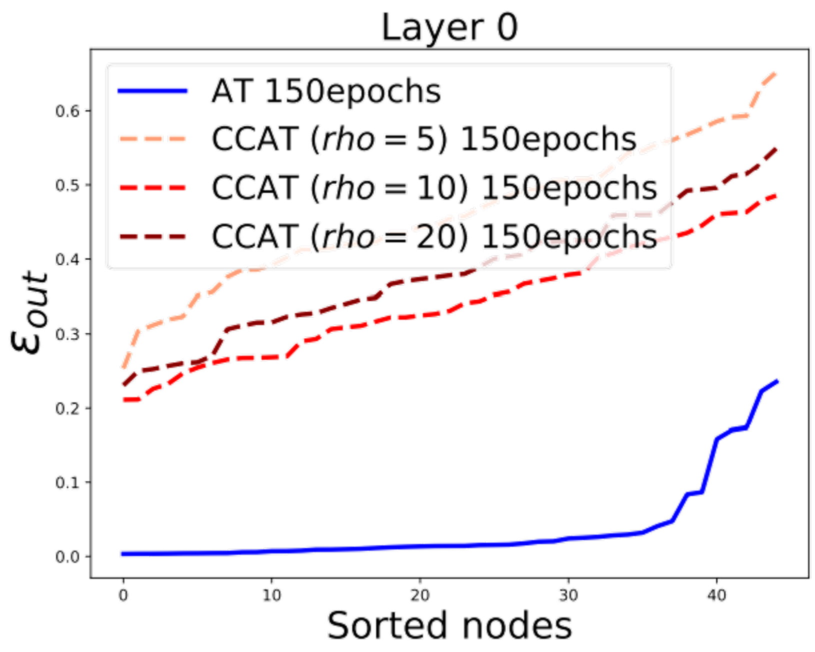
$$\tilde{y} = \lambda(\delta)\mathrm{onehot}(y) + (1 - \lambda(\delta))\frac{1}{K}$$

$$\lambda(\delta) = (1 - \min(1, ||\delta||_\infty/\epsilon))^\rho$$



Larger δ, smaller λ
more uniform label $\tilde{y}$





Model (150 epochs)	Robust Accuracy
CCAT ($\rho = 5$)	47.25%
CCAT ($\rho = 10$)	52.04%
CCAT ($\rho = 20$)	49.33%
Adversarial Training (logits)	84.07%

Thanks!