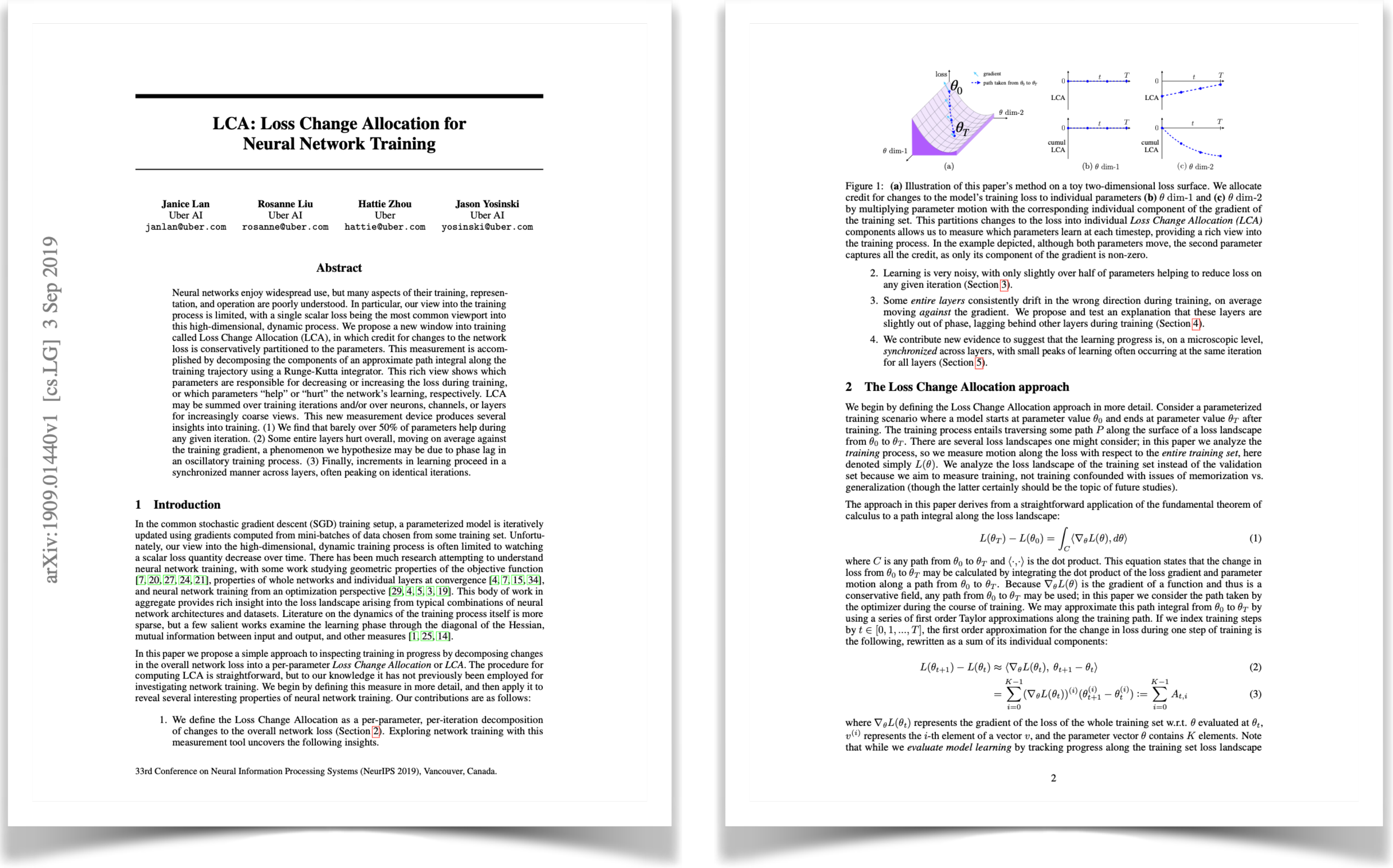
# Two small ideas:
# Approximate LCA, Computational Ethics

MLC Research Jam
28 May 2025

Jason Yosinski
jason@yosinski.com

# Idea #1: Approximate LCA (Loss Change Allocation)


LCA: Loss Change Allocation for Neural Network Training

Janice Lan, Uber AI, janlan@uber.com
Rosanne Liu, Uber AI, rosanne@uber.com
Hattie Zhou, Uber, hattie@uber.com
Jason Yosinski, Uber AI, yosinski@uber.com

Abstract

1 Introduction

2 The Loss Change Allocation approach

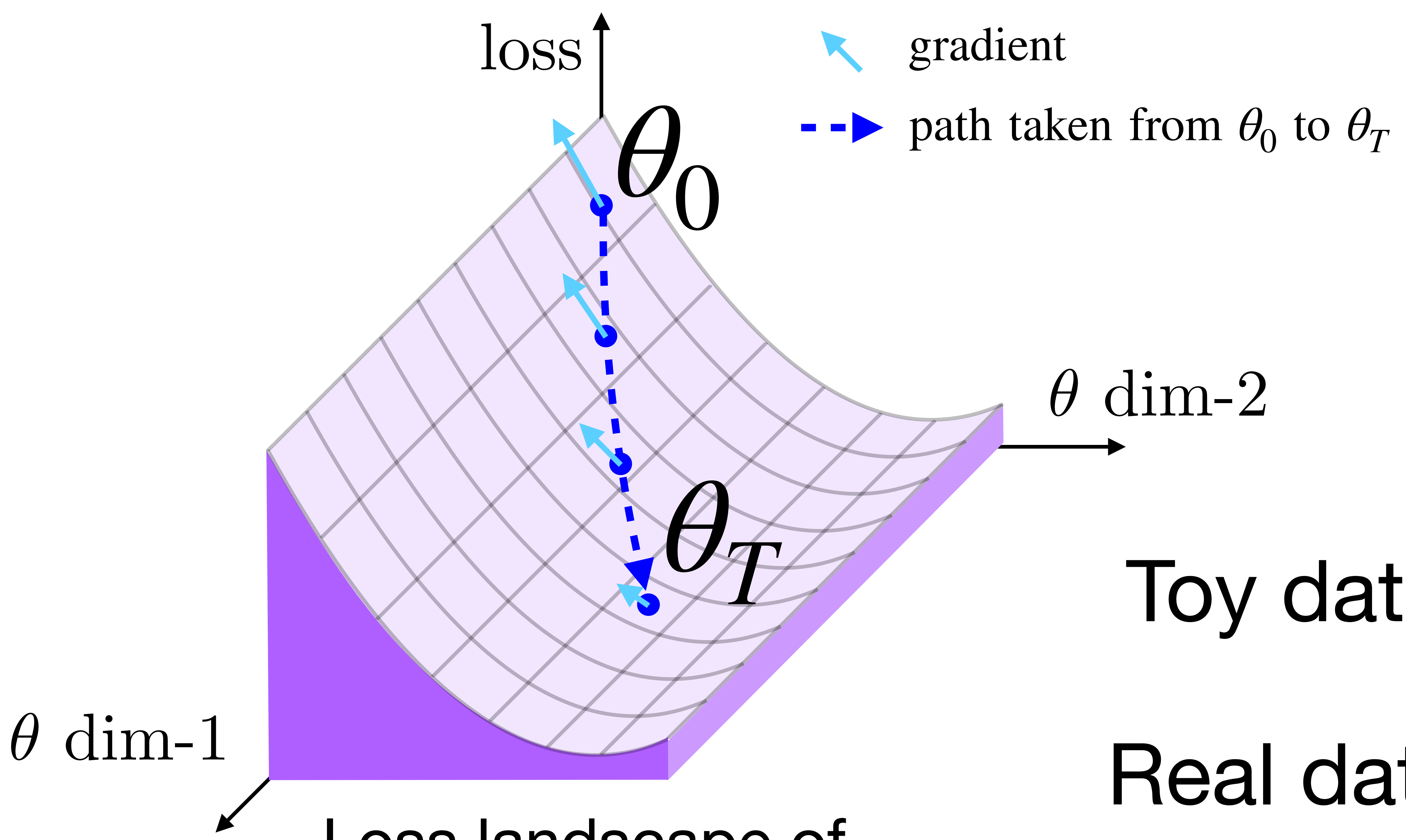
Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. LCA: Loss Change Allocation for Neural Network Training, NeurIPS 2019.

😍 See training in progress!

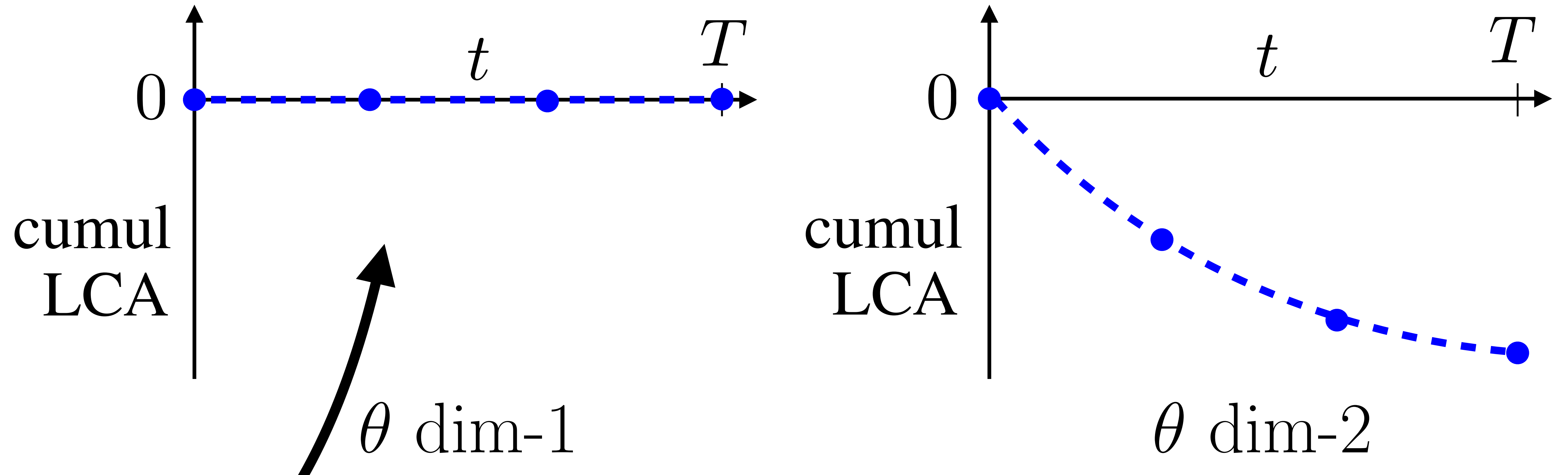
😭 Very slow

If we use aggressive approximation to make it faster, can it still provide useful visibility?

# LCA: Loss Change Allocation for Neural Network Training

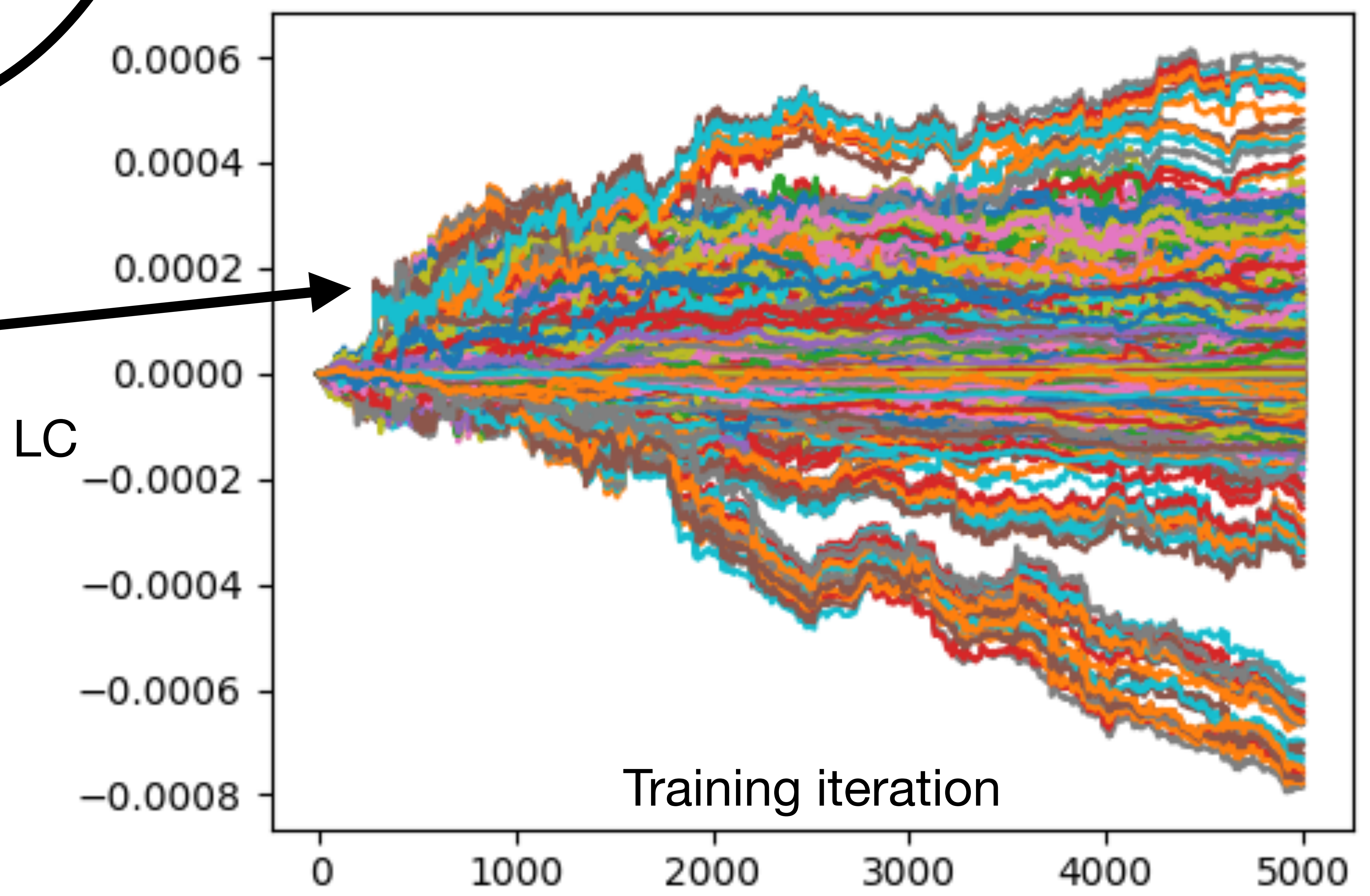



Loss landscape of entire train set




Toy data

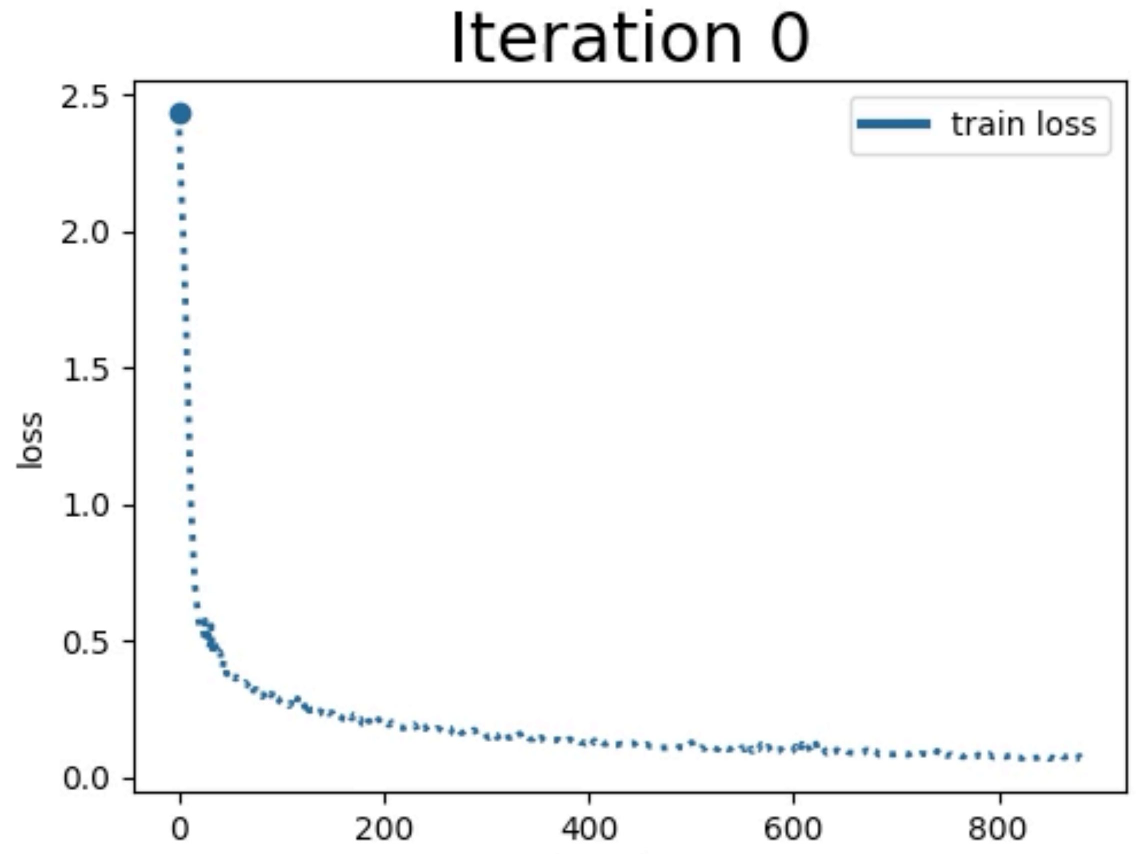
Real data

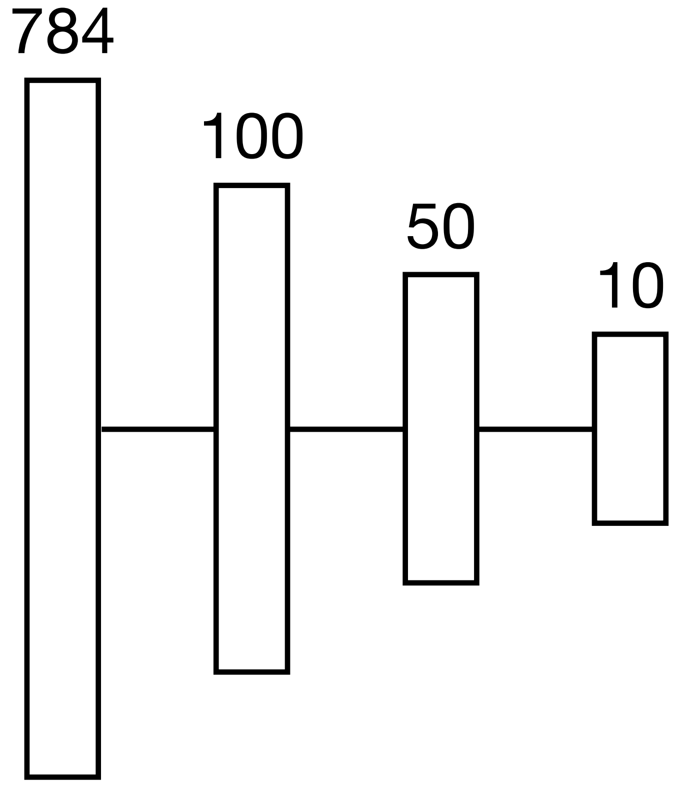



Simple approach → New visibility into training

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

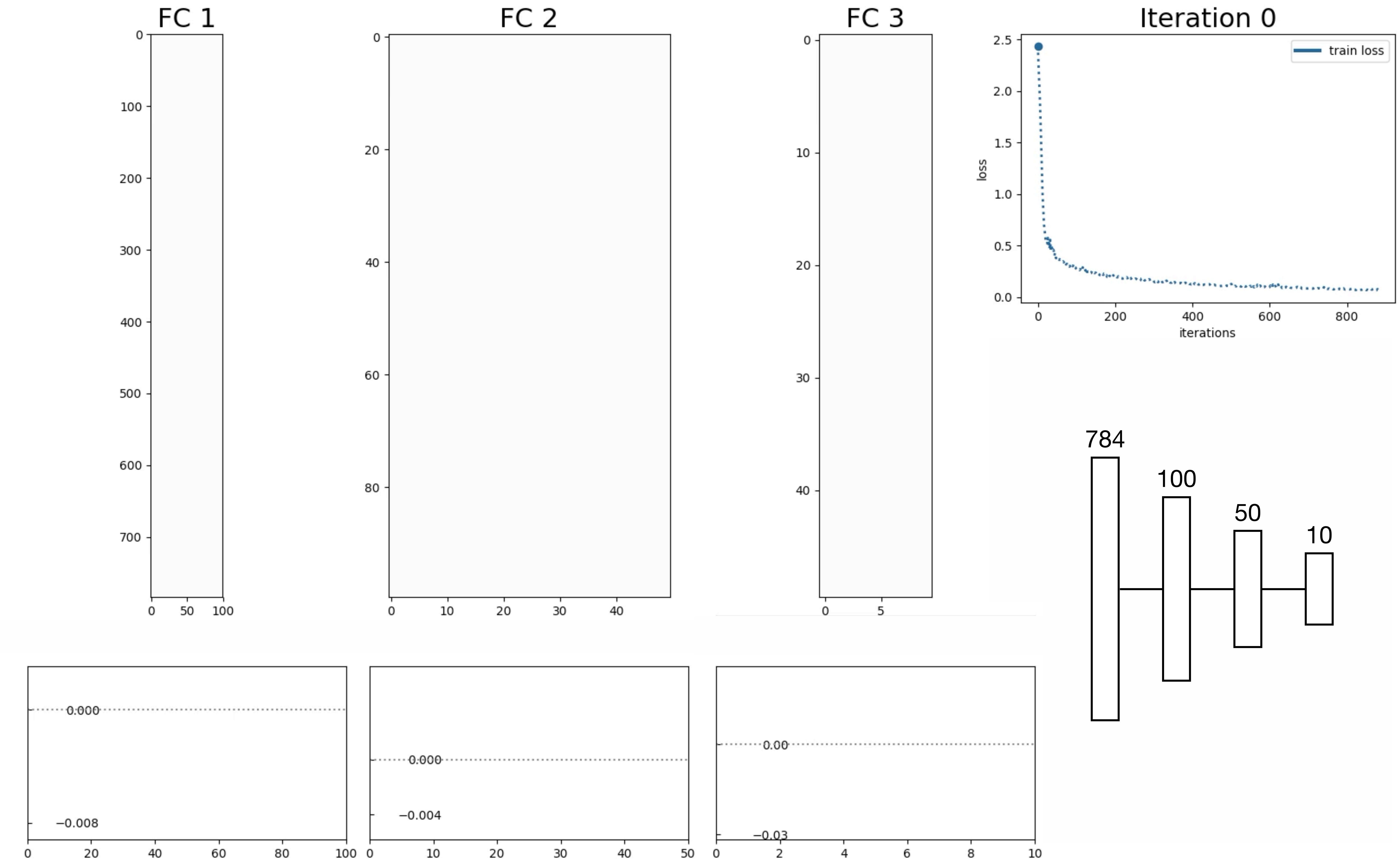
# LCA: Loss Change Allocation for Neural Network Training



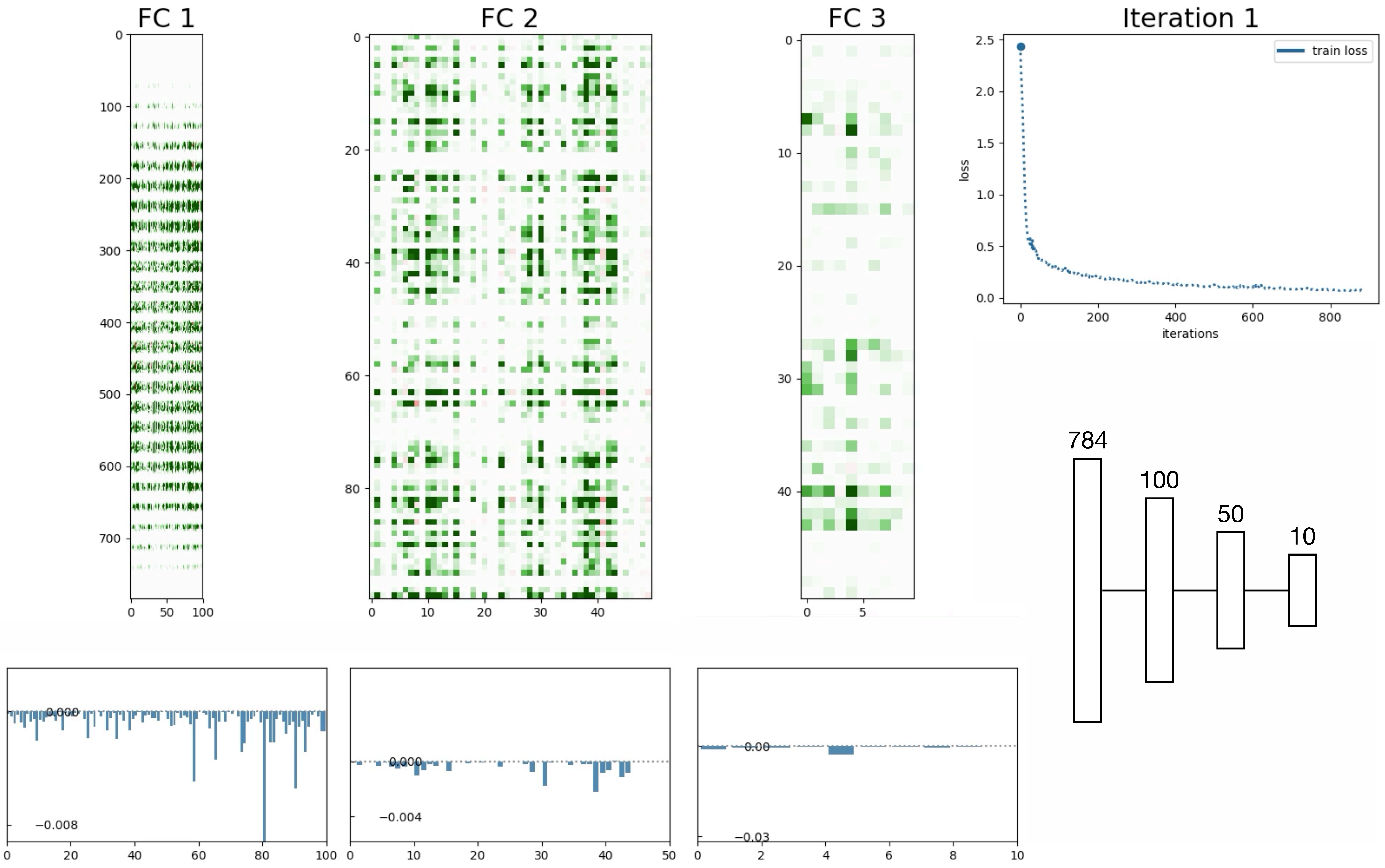

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

# LCA: Loss Change Allocation for Neural Network Training




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

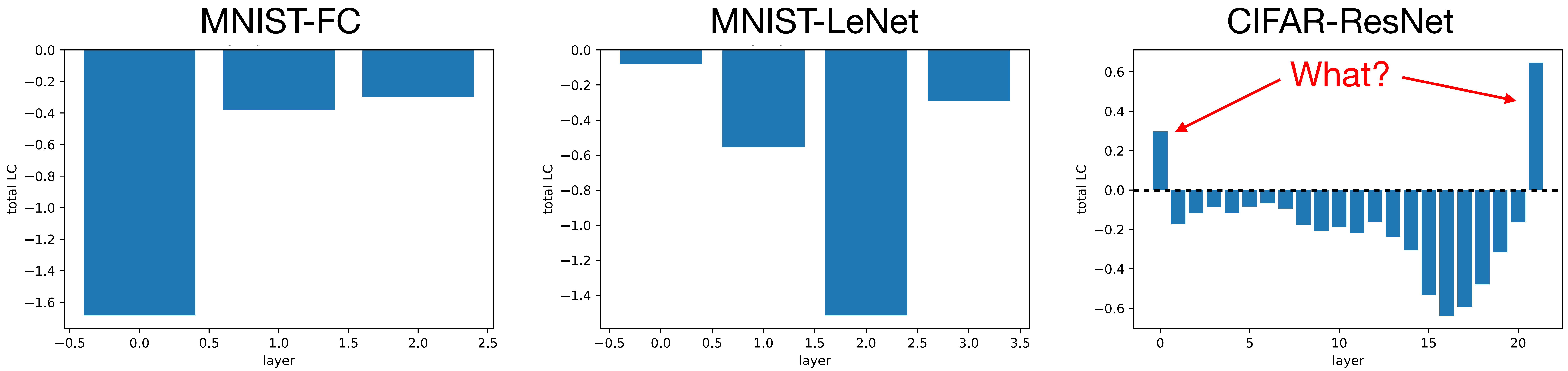
# LCA: Loss Change Allocation for Neural Network Training




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

# LCA: Loss Change Allocation for Neural Network Training

Sum over layers




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

# Request for Plot: Approximate LCA

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:
    Compute per batch: 1x

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:
Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:

Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

Approximate LCA:

$$\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:

Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

Approximate LCA:

$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)}$ Biased

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:

Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

Approximate LCA:

$$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)}$$ Biased

$$\nabla L_{\text{batch i+1}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:

Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

Approximate LCA:

$$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)}$$ Biased

$$\nabla L_{\text{batch i+1}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1x

# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:
  Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

  Compute per batch: 1000x

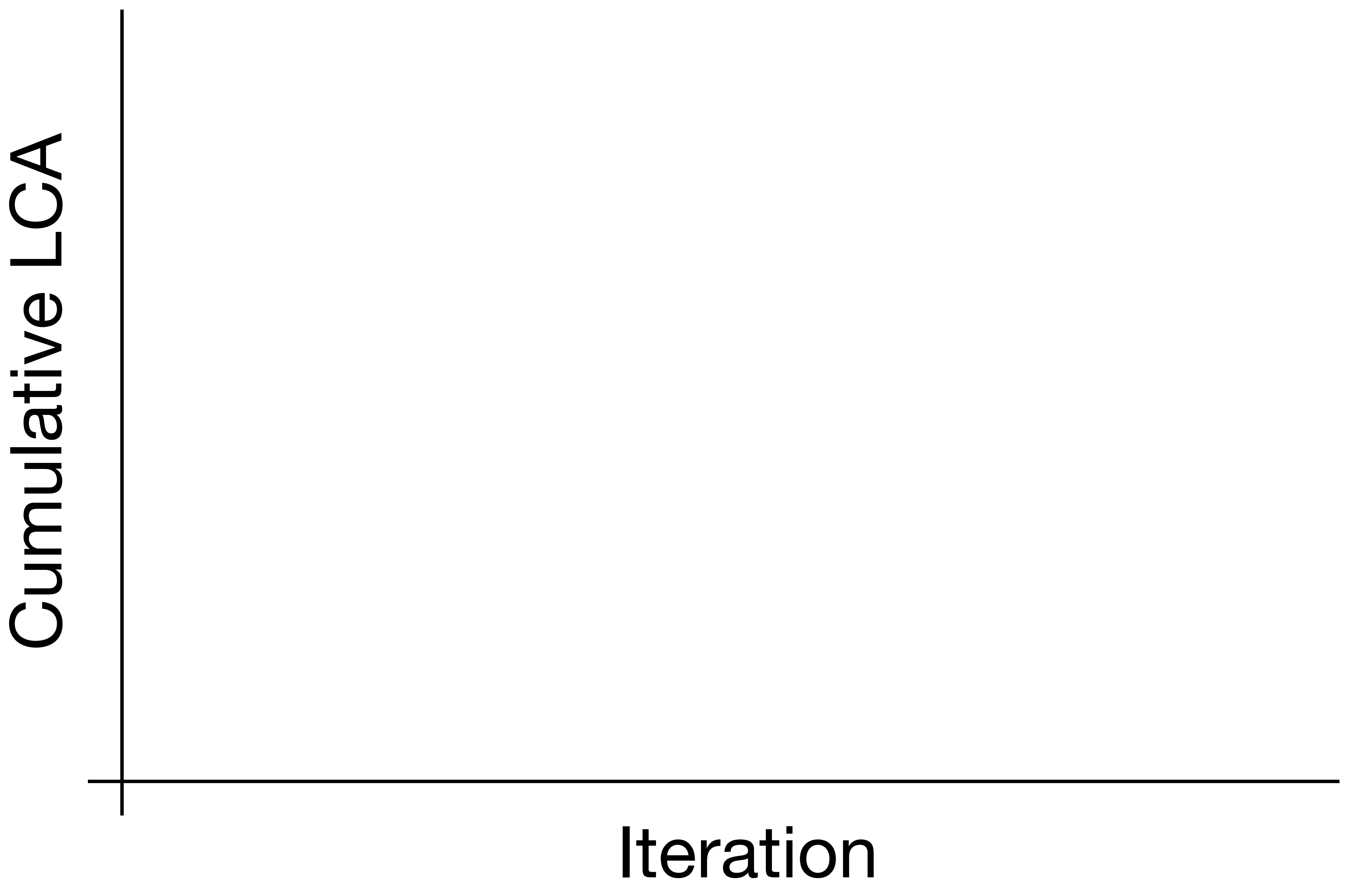
Approximate LCA:

$$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)} \quad \text{Biased}$$

$$\nabla L_{\text{batch i+1}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

  Compute per batch: 1x

Is this approximation good?



# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:

Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

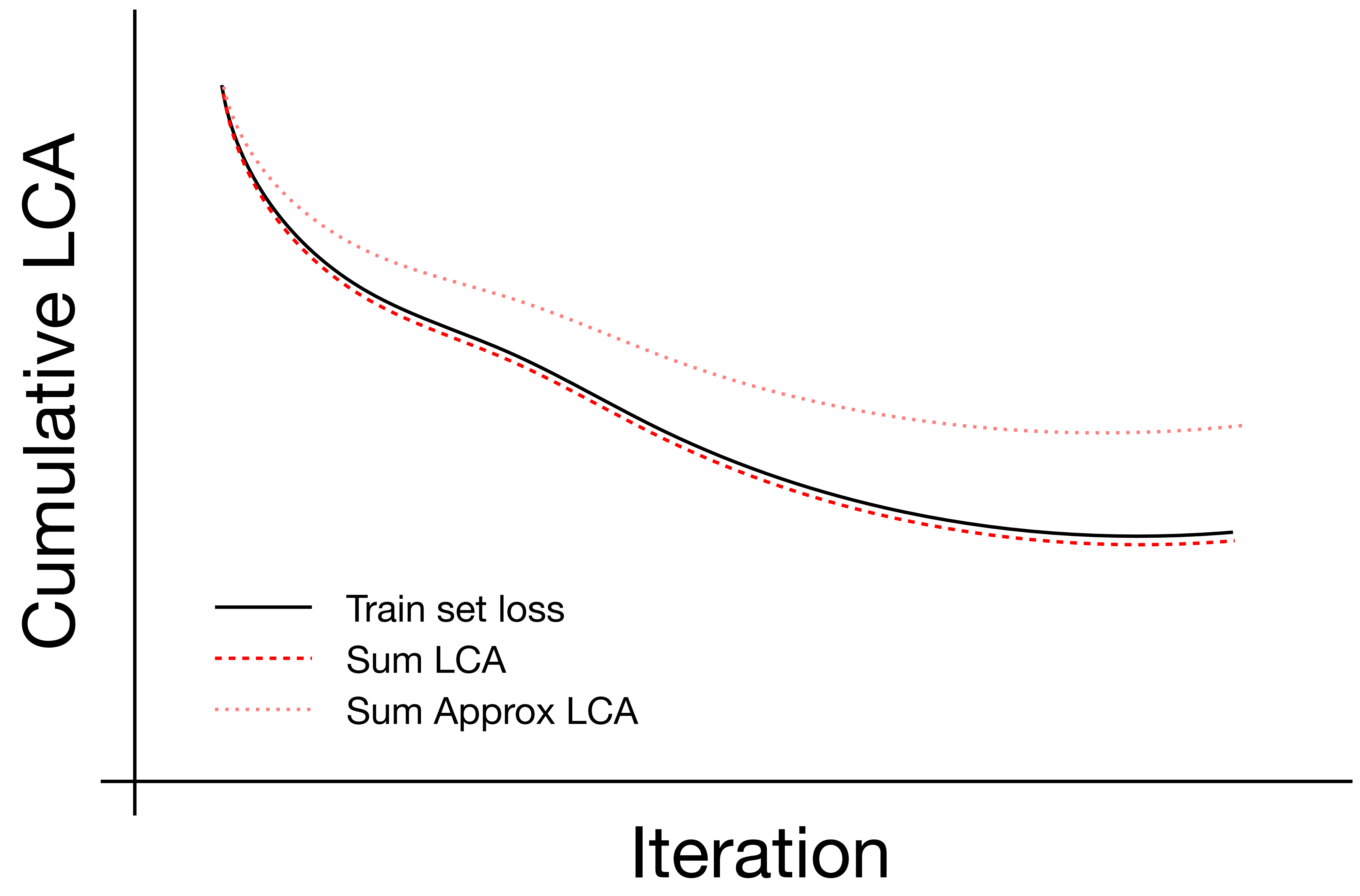
Approximate LCA:

$$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)}$$ Biased

$$\nabla L_{\text{batch i+1}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1x

Is this approximation good?



# Request for Plot: Approximate LCA

*Example: training set of 1000 batches*

Ordinary training:
Compute per batch: 1x

Original LCA:

$$\nabla L_{\text{train}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1000x

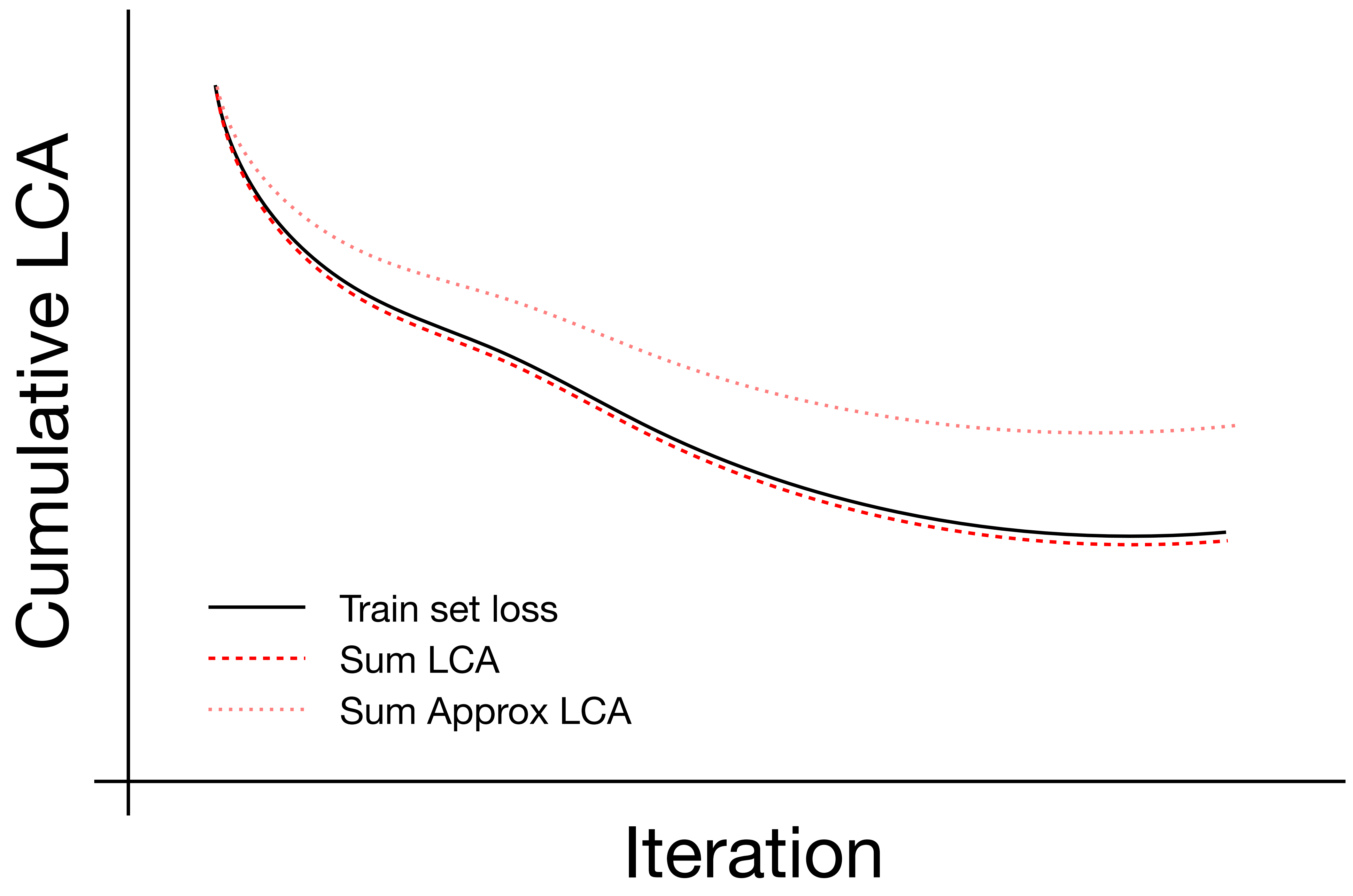
Approximate LCA:

$$\cancel{\nabla L_{\text{batch i}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)}$$ Biased

$$\nabla L_{\text{batch i+1}}(\theta_i) \cdot (\theta_{i+1} - \theta_i)$$

Compute per batch: 1x

Is this approximation good?




Next questions:
- How to incorporate RK-4?
- If this works, are per-layer sums also well approximated?

# Idea #2: Computational Ethics

## How we used to do Computer Vision

- Humans create features
- Failures → humans create smarter features
- Endless Whack-a-mole


## How we align models now

- Humans write prompts (constitutions, specs)
- Failures → humans create smarter prompts
- Endless Whack-a-mole

  See e.g. 24k token Claude system prompt: "…Claude provides emotional support alongside accurate medical or psychological information or terminology where relevant."

## What worked better

- Models learn their own features

## What might work better?

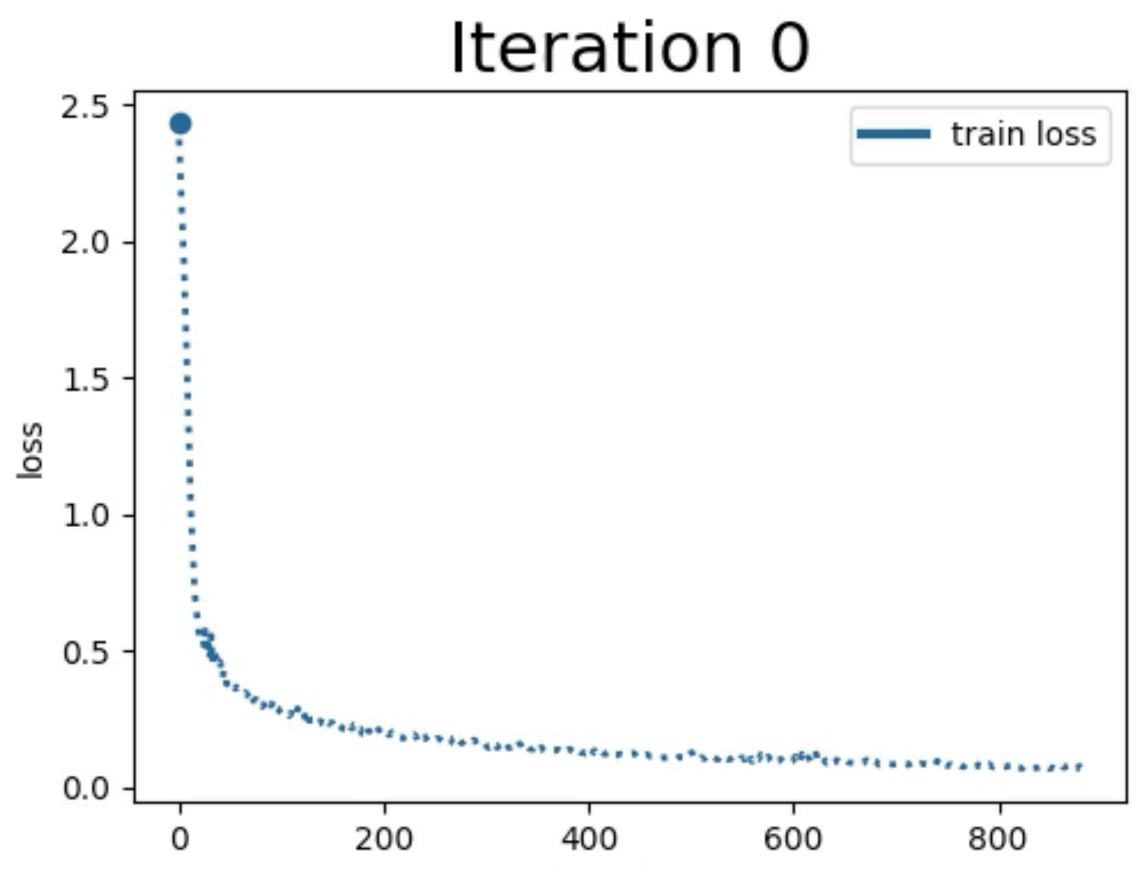
- Models learn their own ethics
- Start with using models to map the existing, diverse landscape of human ethics

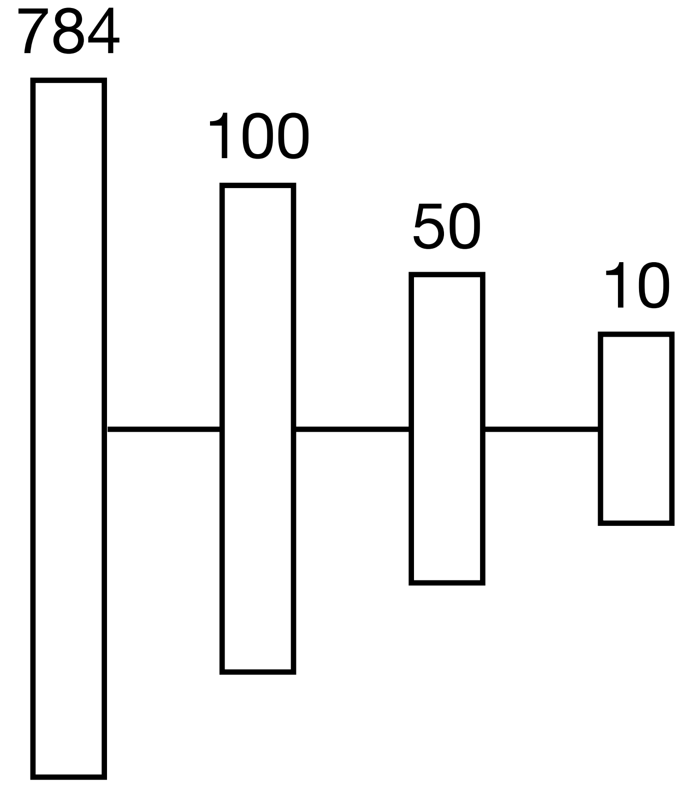
# Appendix: extra LCA slides

# LCA: Loss Change Allocation for Neural Network Training

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

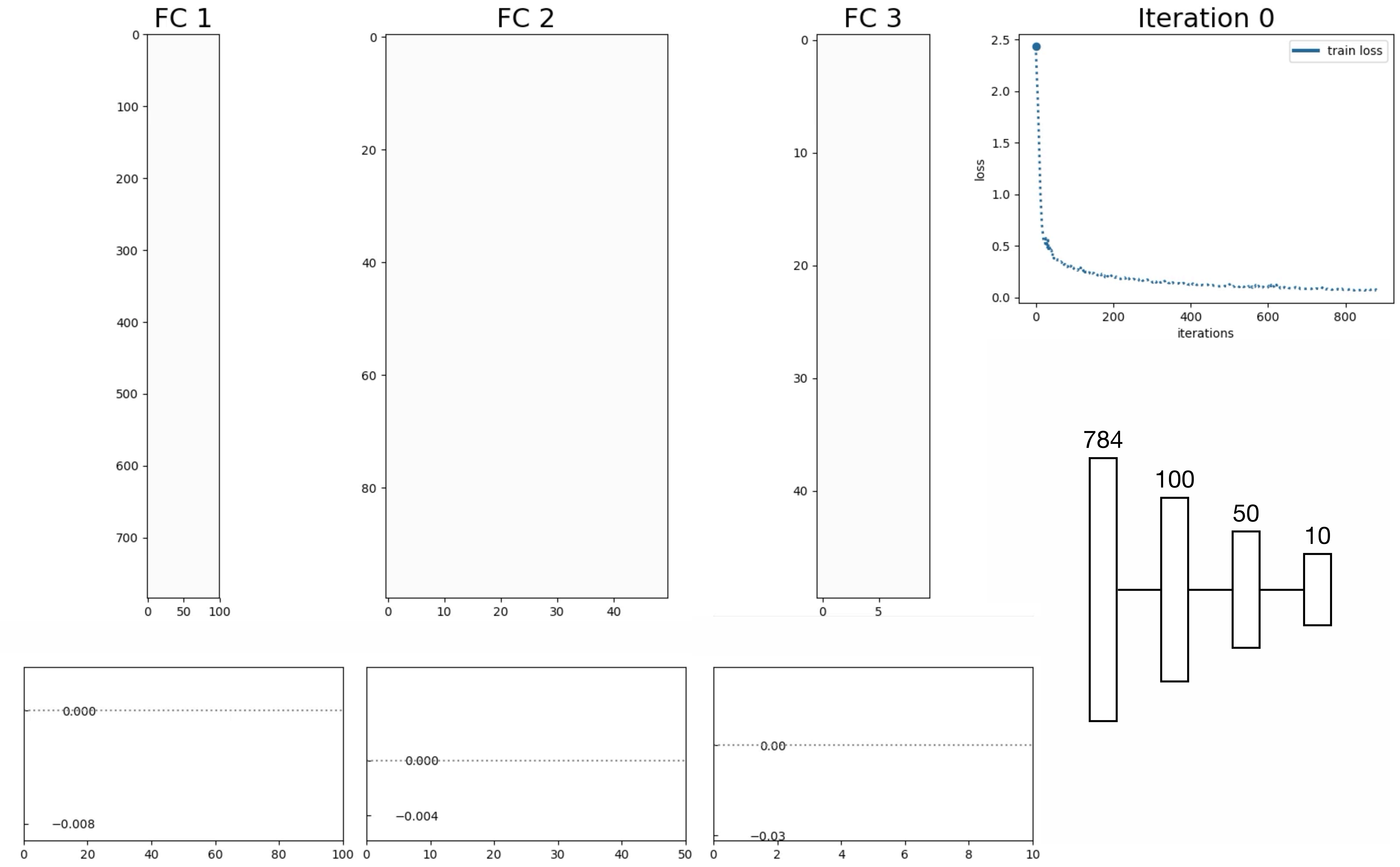
# LCA: Loss Change Allocation for Neural Network Training



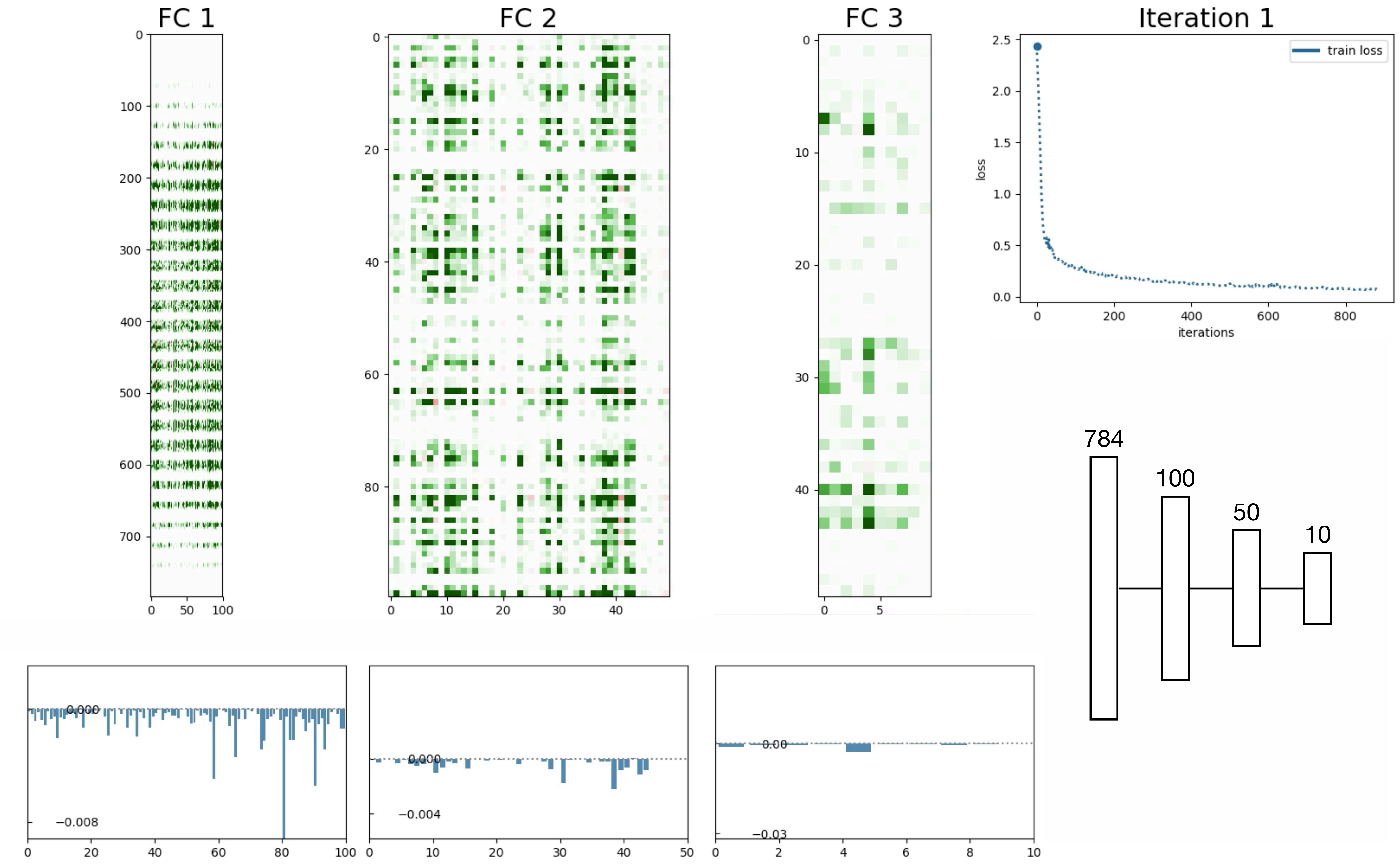

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

# LCA: Loss Change Allocation for Neural Network Training



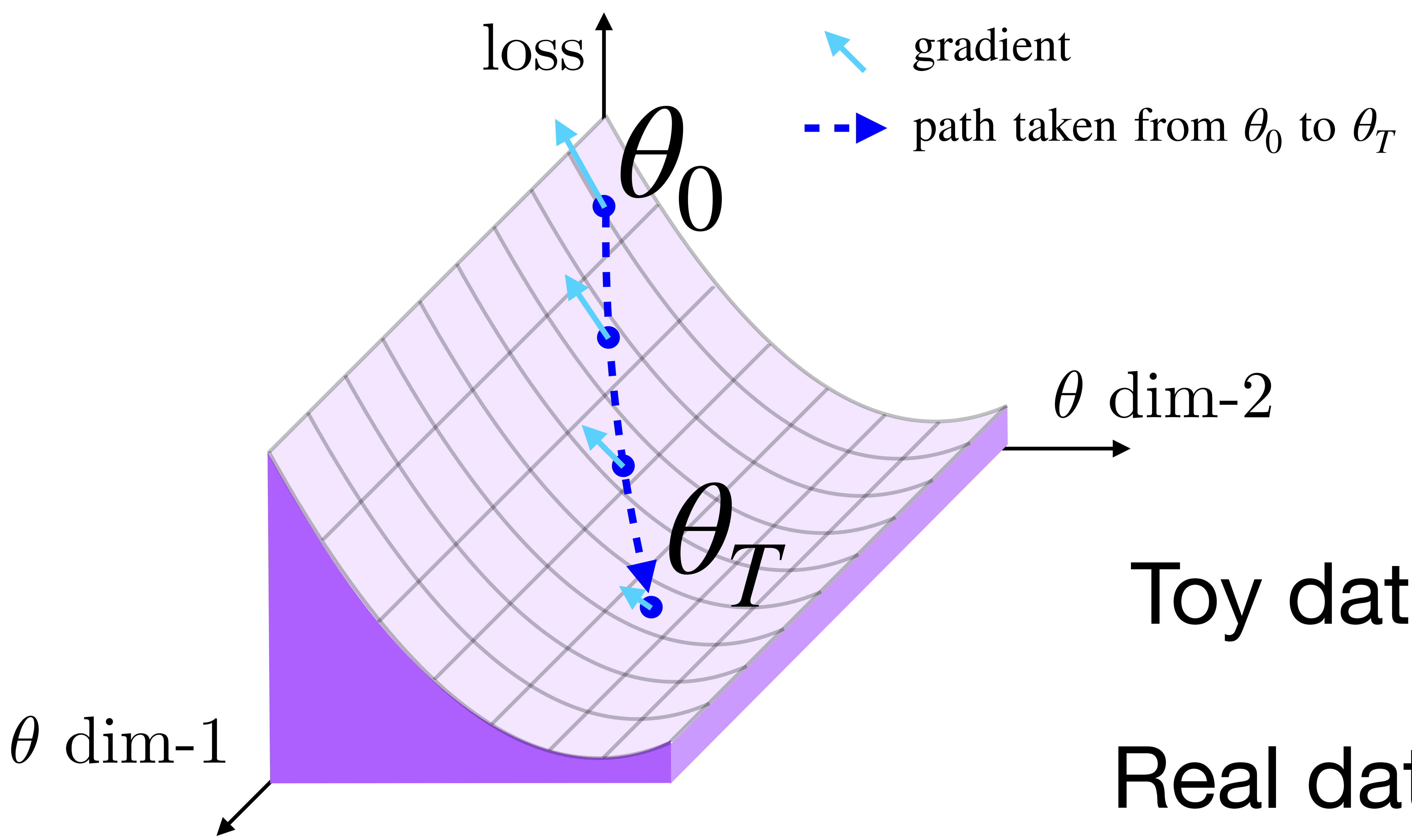

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

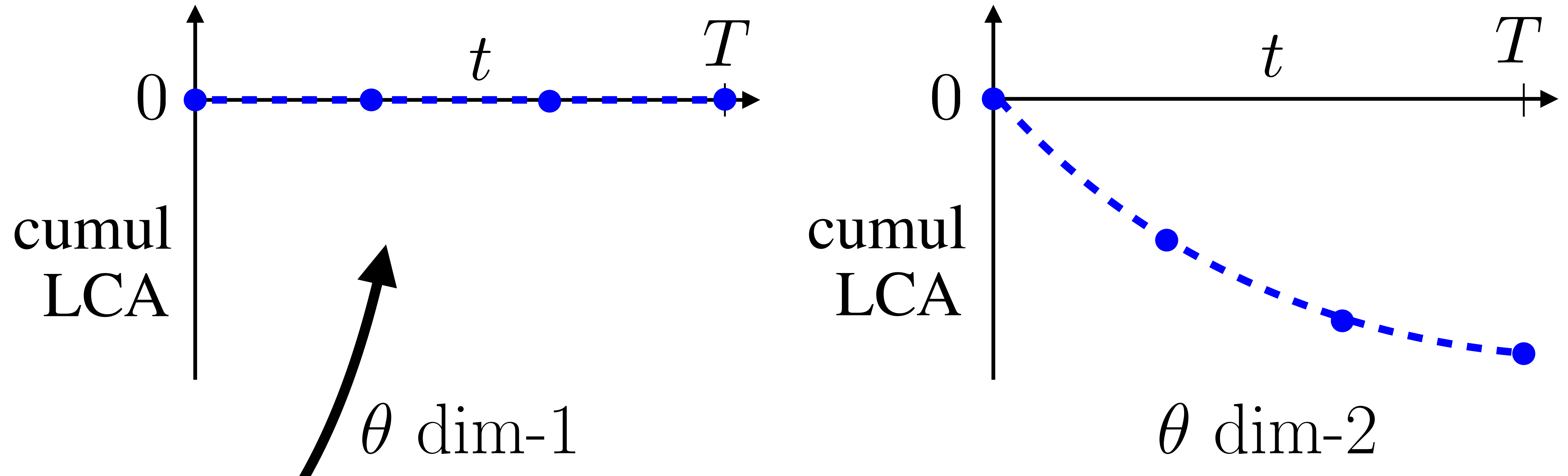
# LCA: Loss Change Allocation for Neural Network Training




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

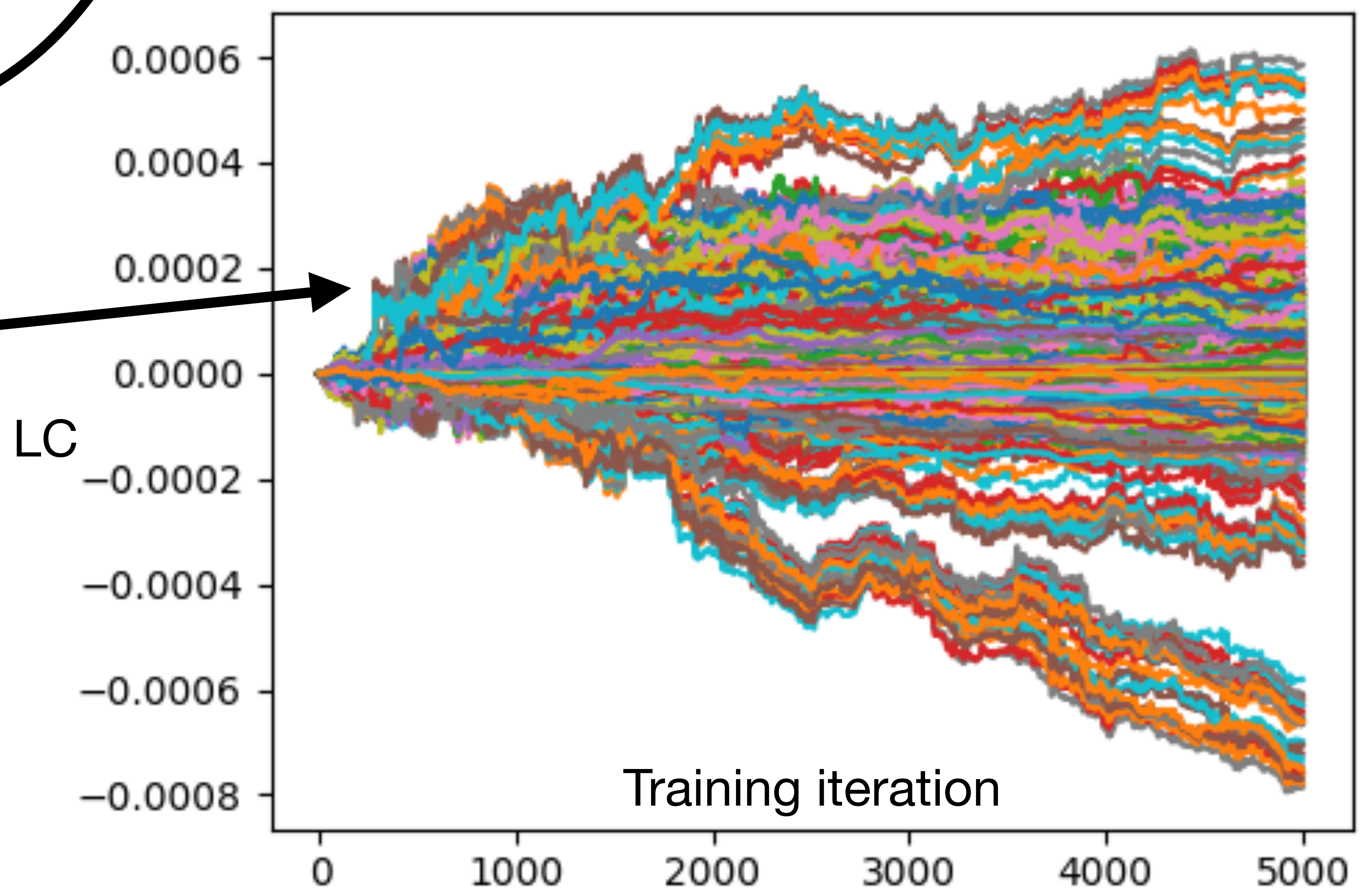
# LCA: Loss Change Allocation for Neural Network Training




Toy data

Real data



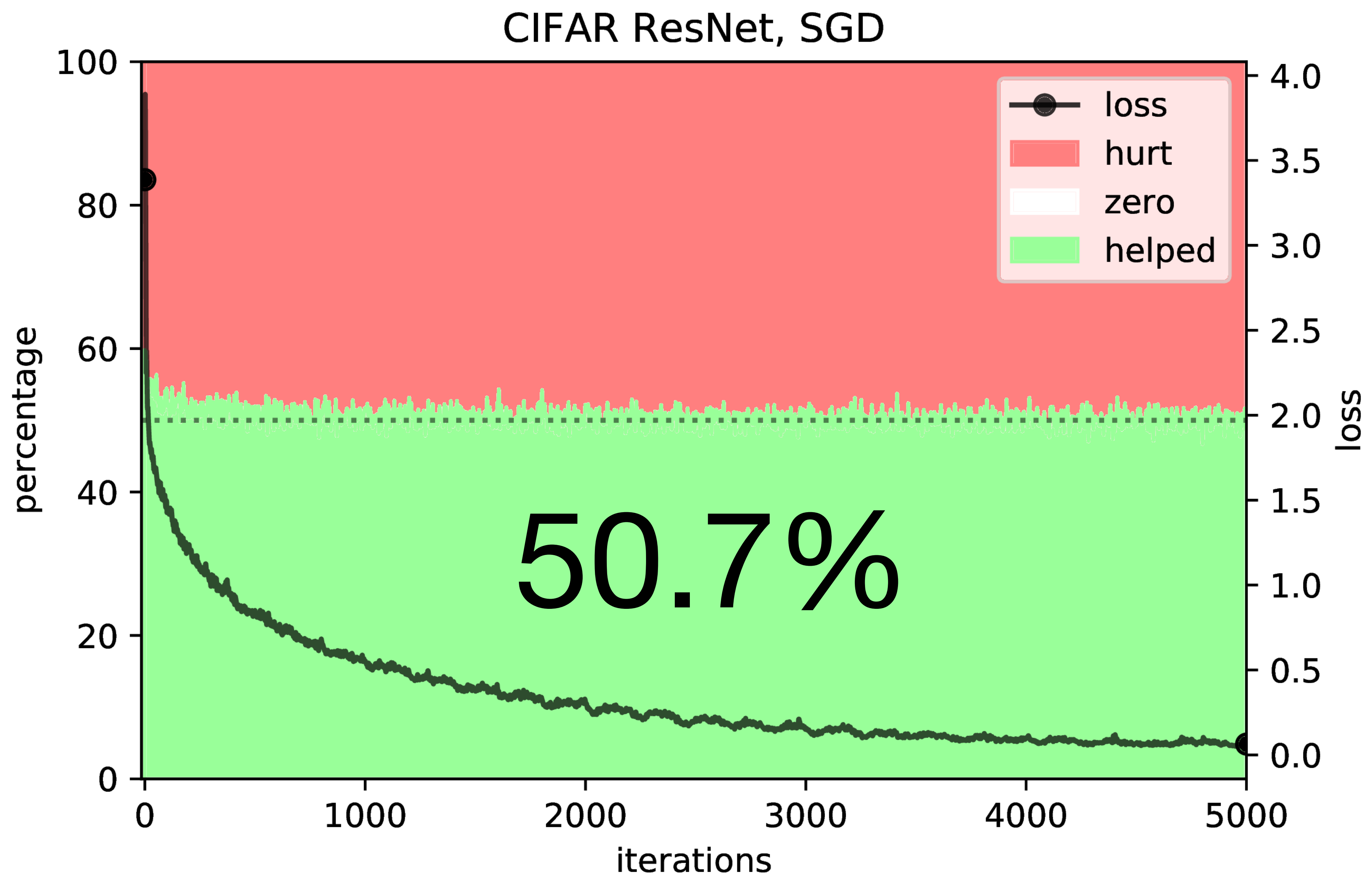

Simple approach → New visibility into training

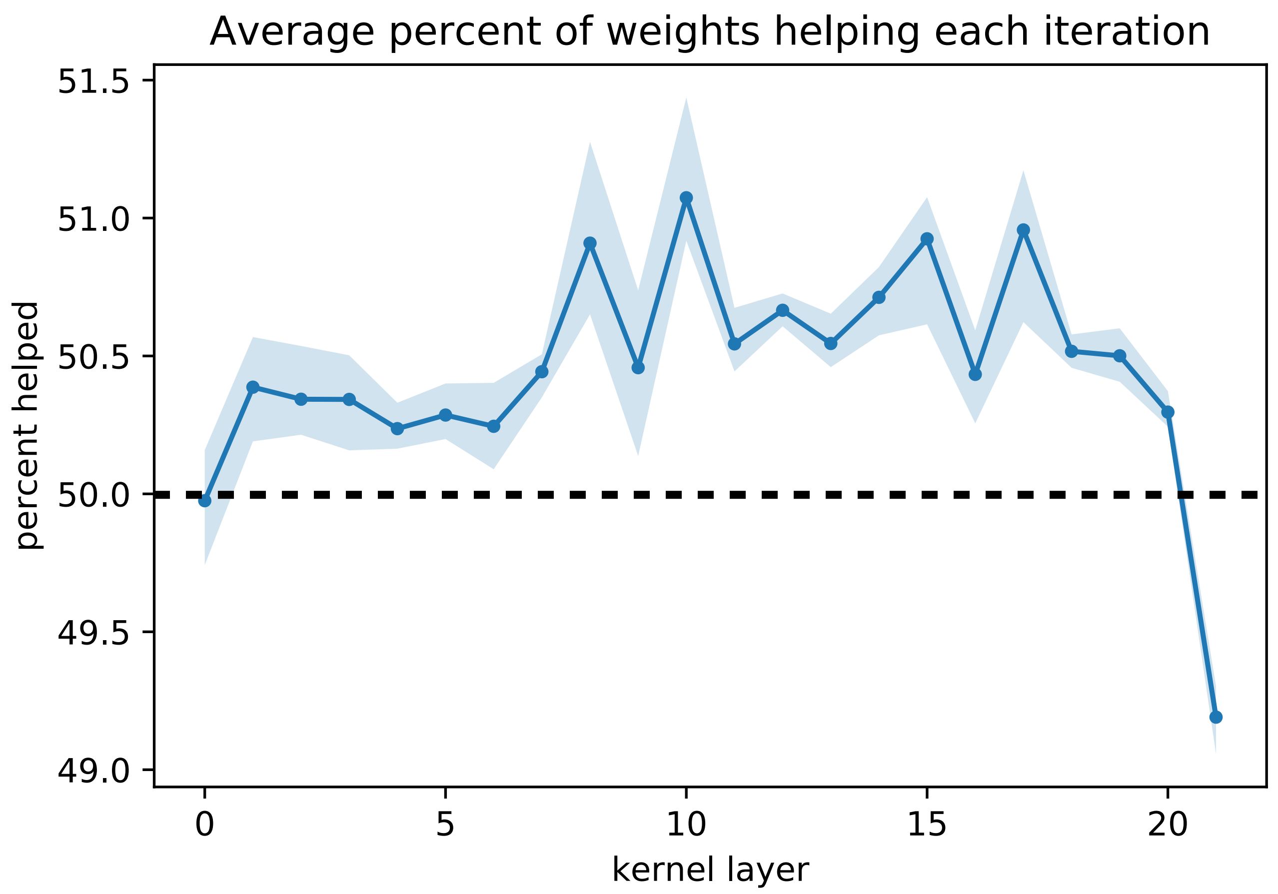
Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

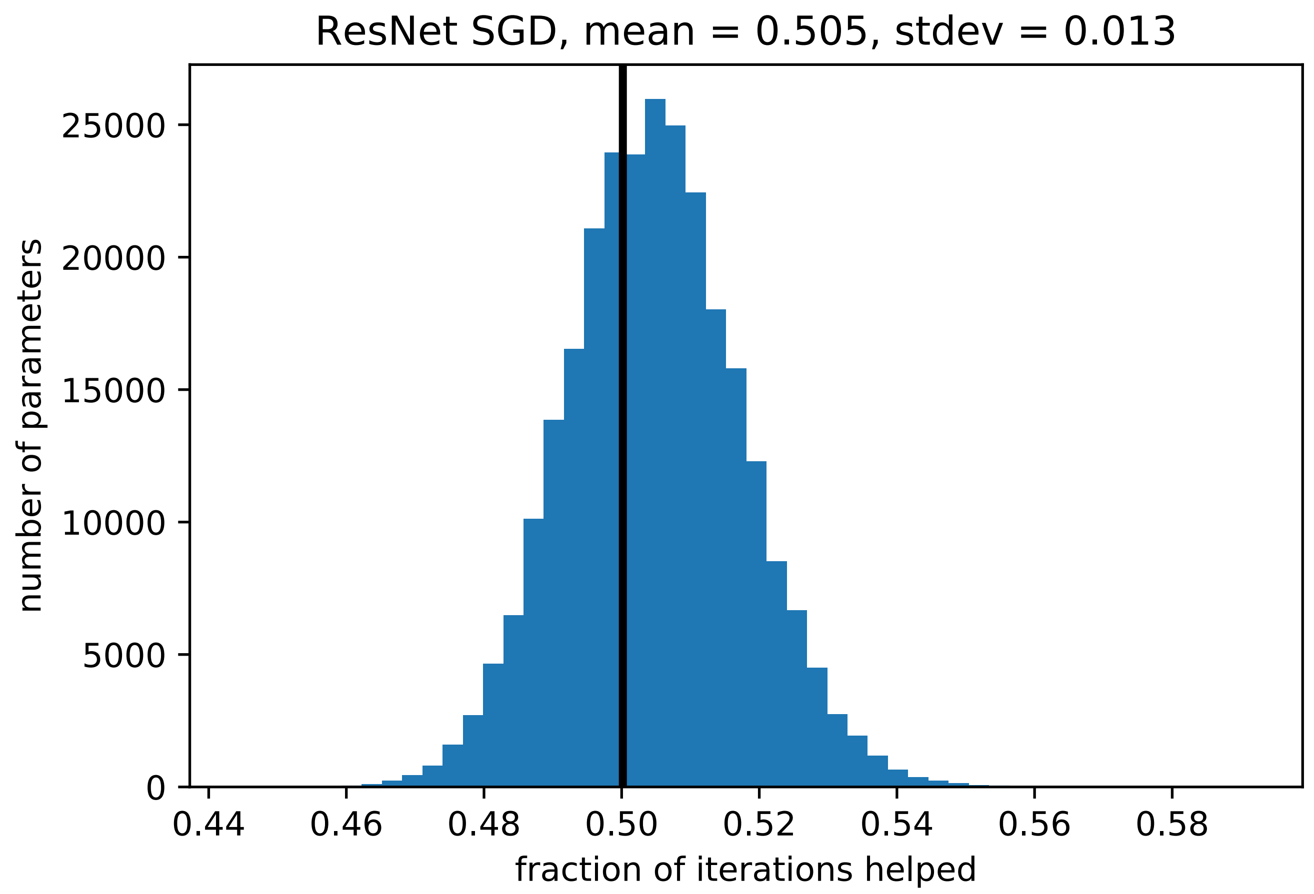
# LCA: Loss Change Allocation for Neural Network Training

## ❶ Training is noisy

- Holds for all layers
- Holds for all params
- Holds for many hyperparams
  (50.3% – 51.6%)



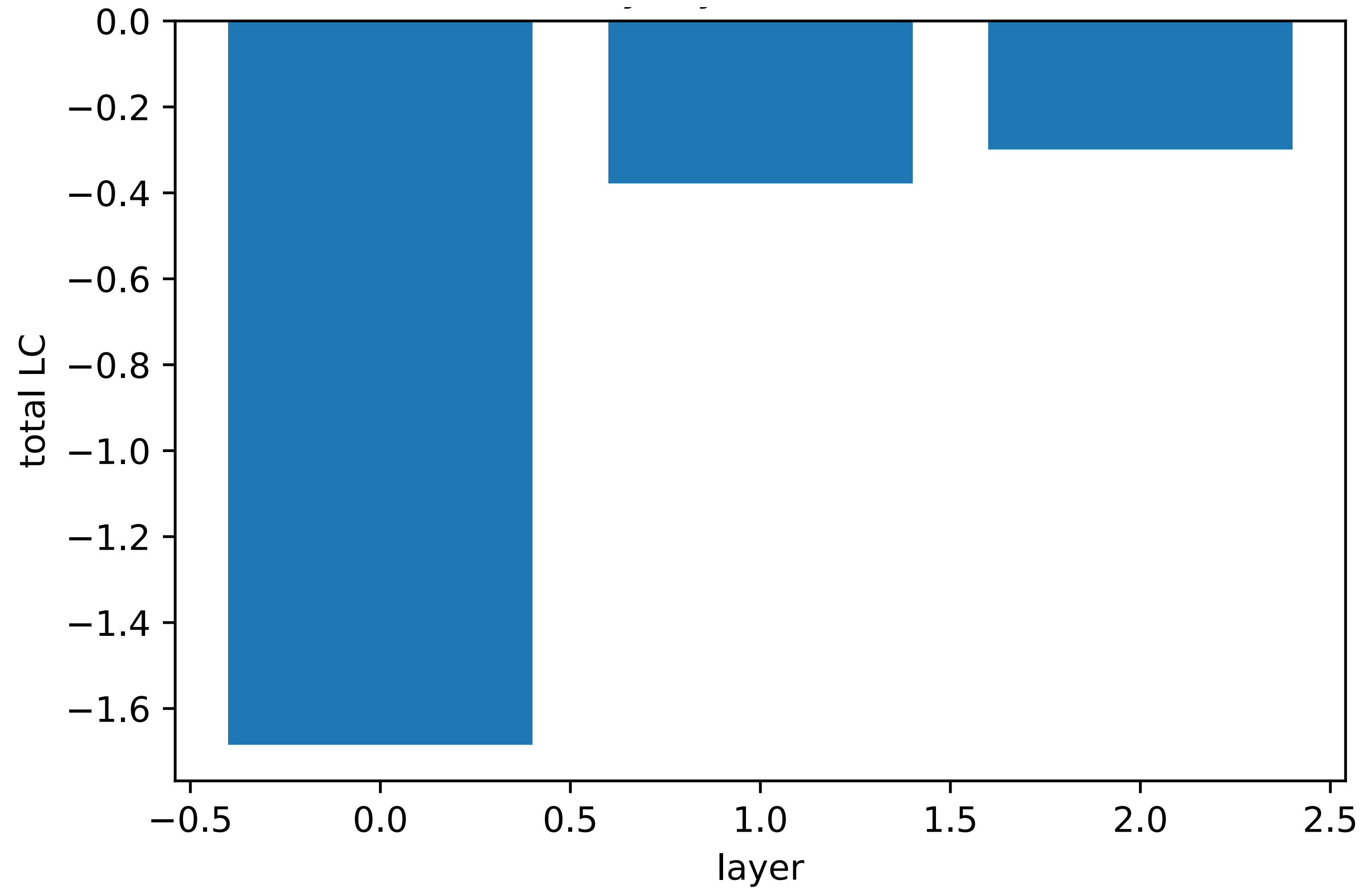




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

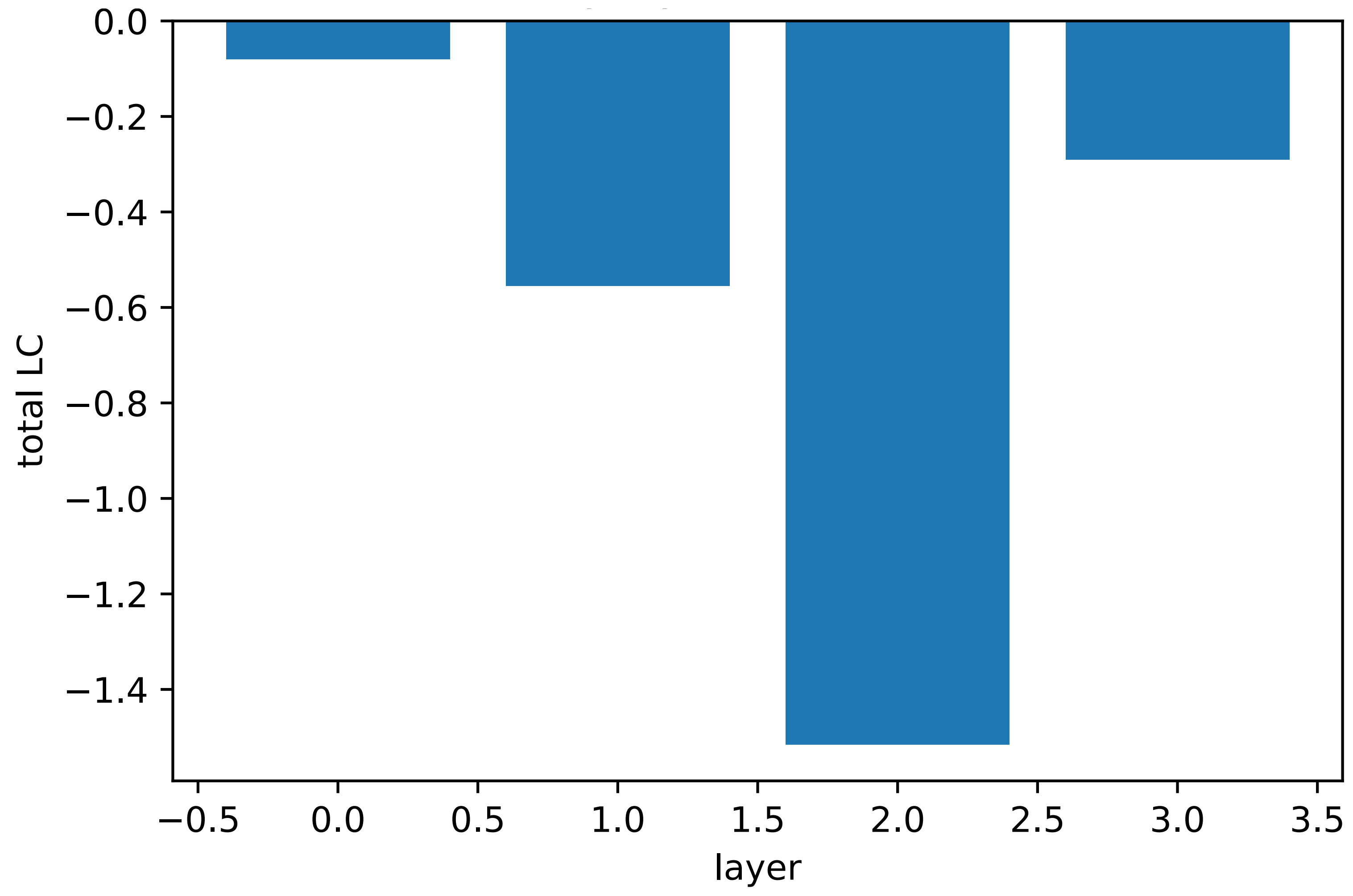
# LCA: Loss Change Allocation for Neural Network Training

1 Training is noisy  2 Some layers go backwards

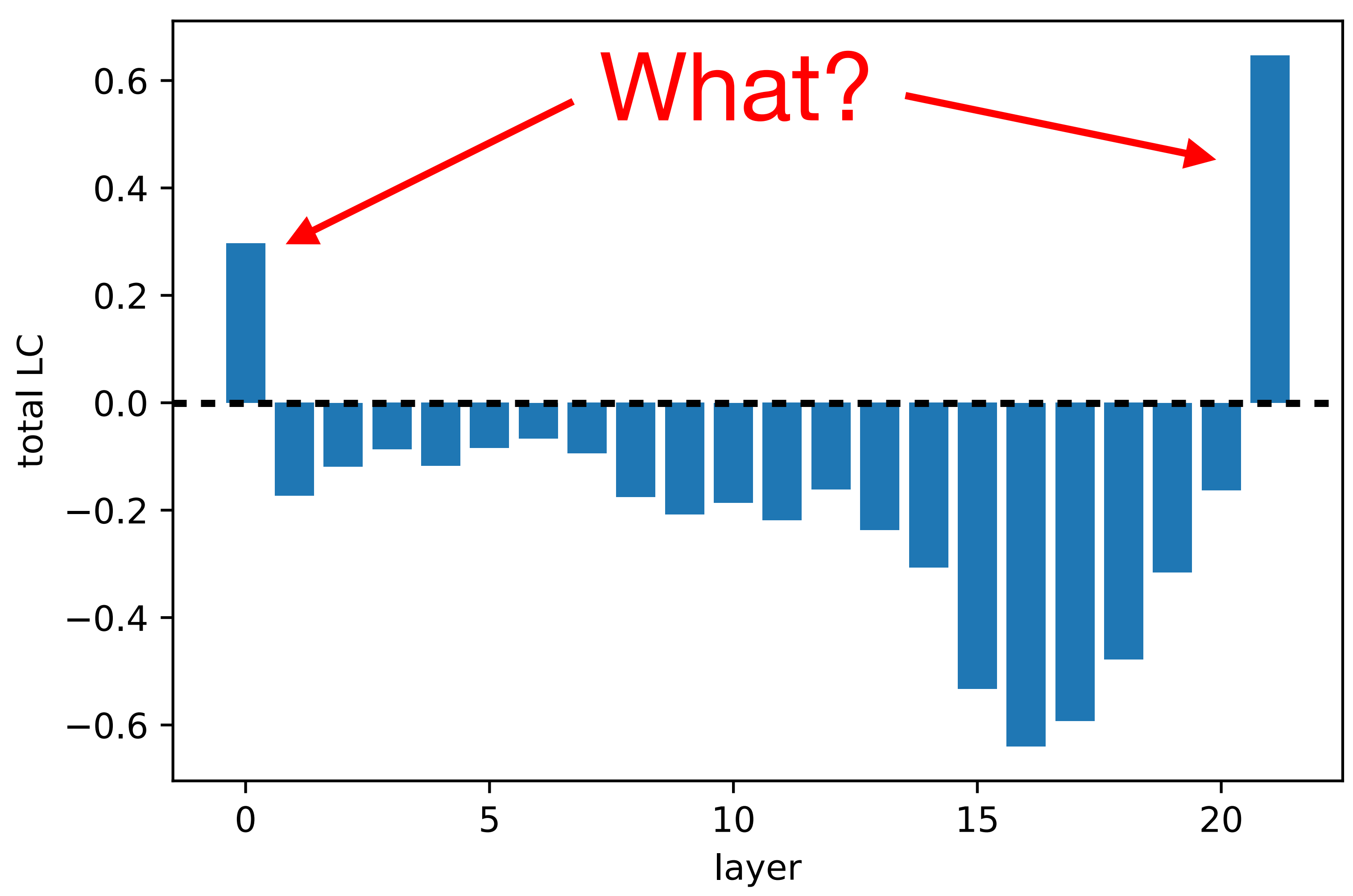
MNIST-FC

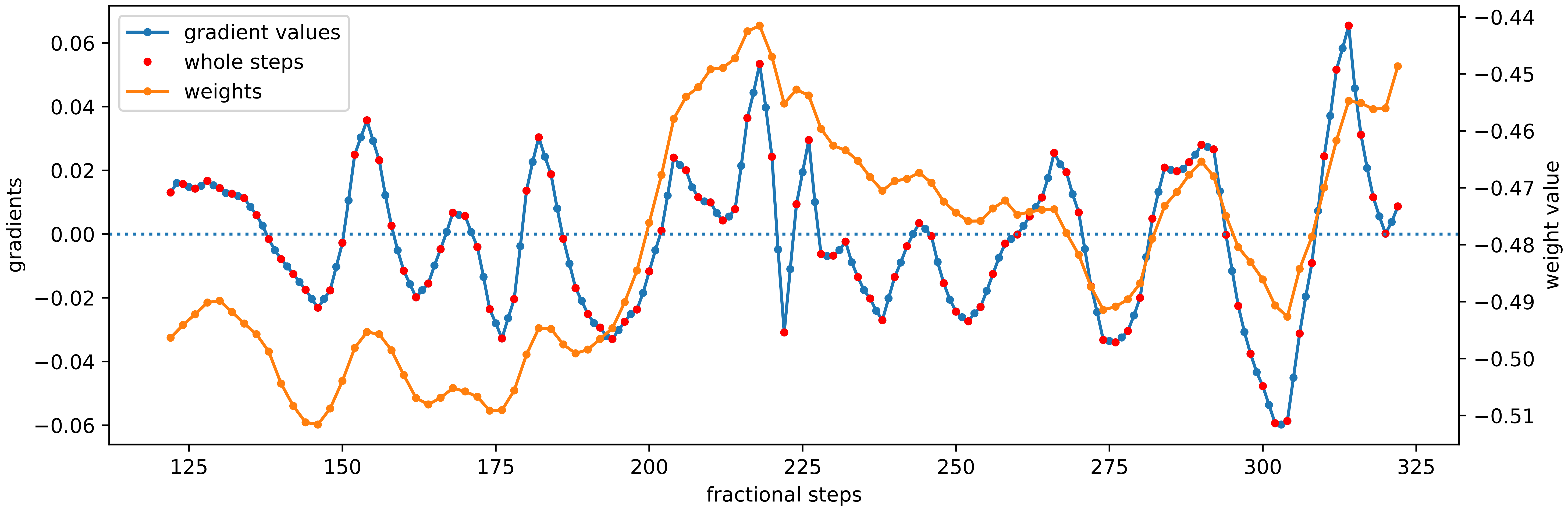



MNIST-LeNet




CIFAR-ResNet



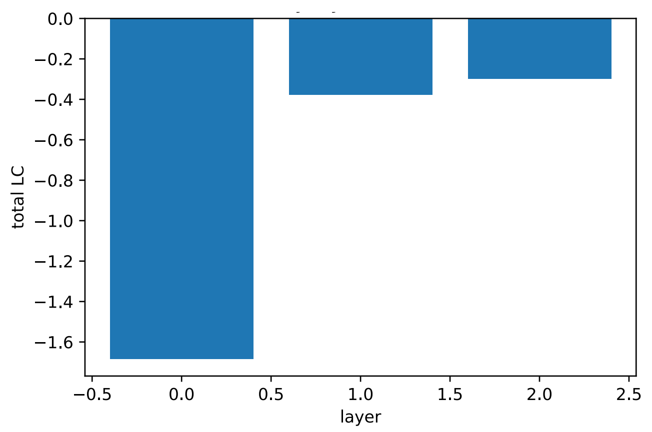

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

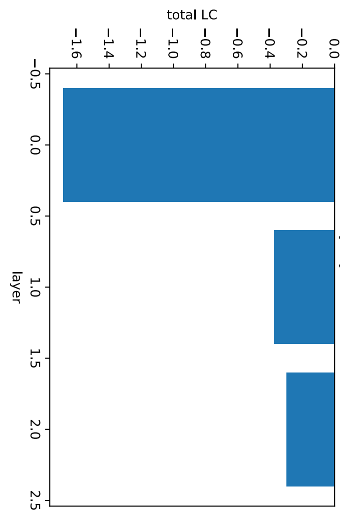
# LCA: Loss Change Allocation for Neural Network Training

1 Training is noisy

2 Some layers go backwards

MNIST-FC




Layer 1

Layer 2

Layer 3

0

25

Training iterations

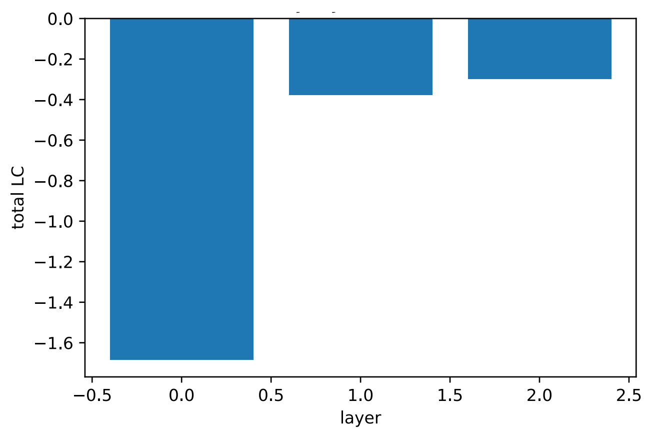
Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

# LCA: Loss Change Allocation for Neural Network Training

1 Training is noisy

2 Some layers go backwards

MNIST-FC





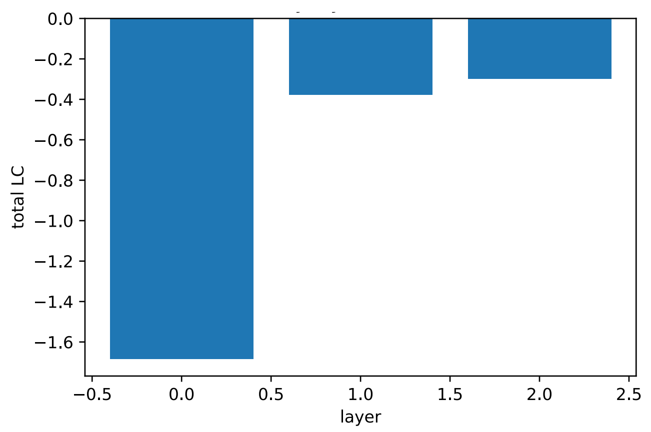
Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

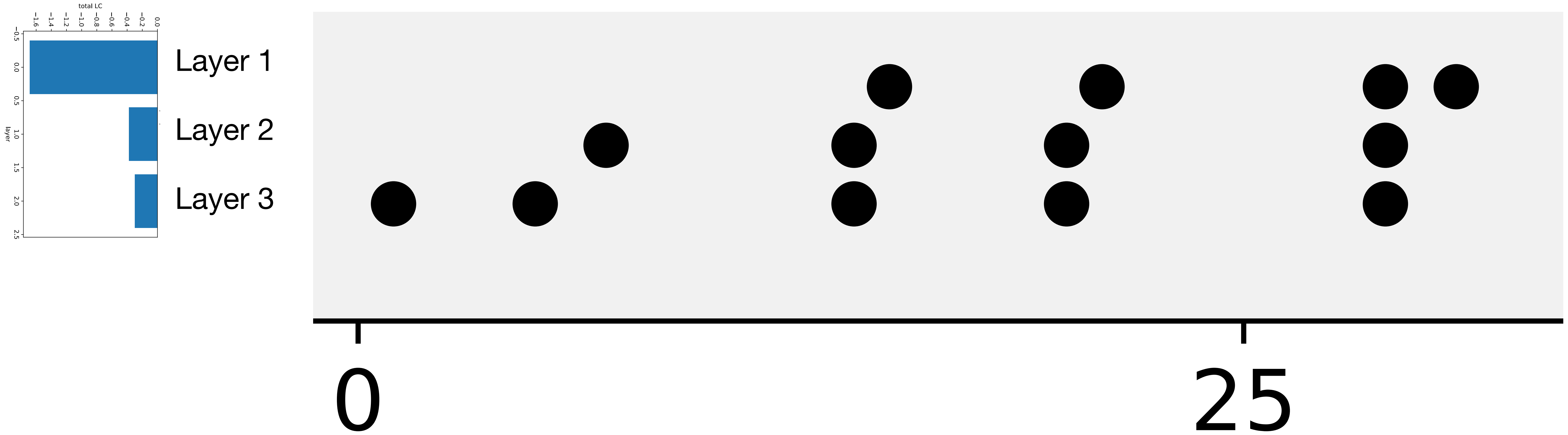
# LCA: Loss Change Allocation for Neural Network Training

1 Training is noisy

2 Some layers go backwards

MNIST-FC




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

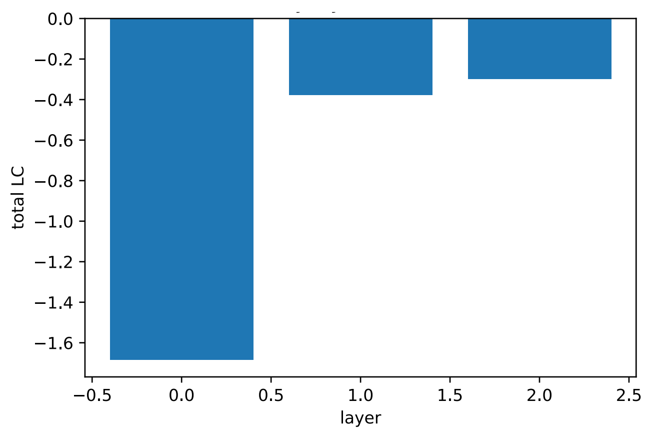
# LCA: Loss Change Allocation for Neural Network Training

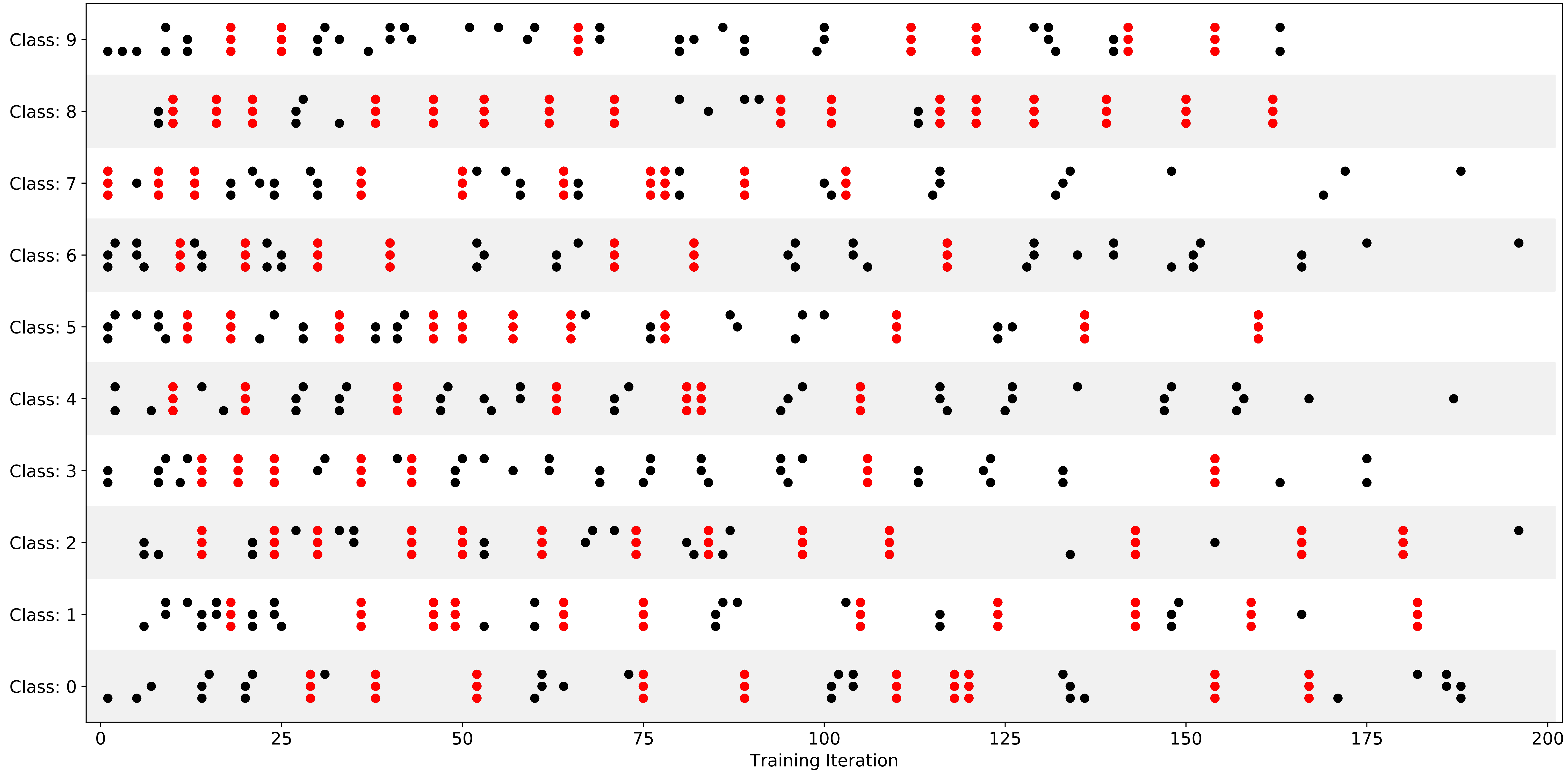
1 Training is noisy

2 Some layers go backwards

3 Some micro-learning is synchronized

MNIST-FC




Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

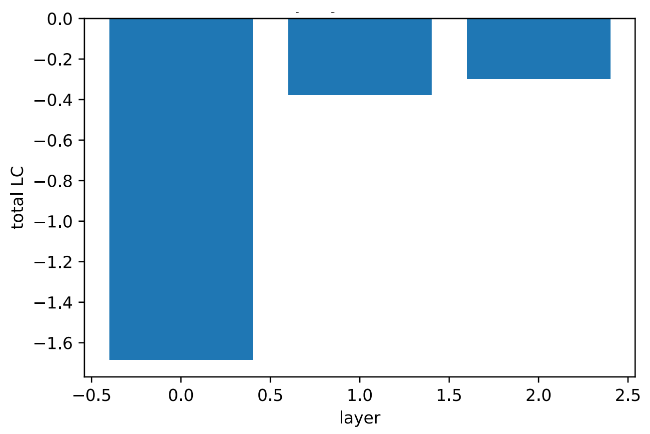
# LCA: Loss Change Allocation for Neural Network Training

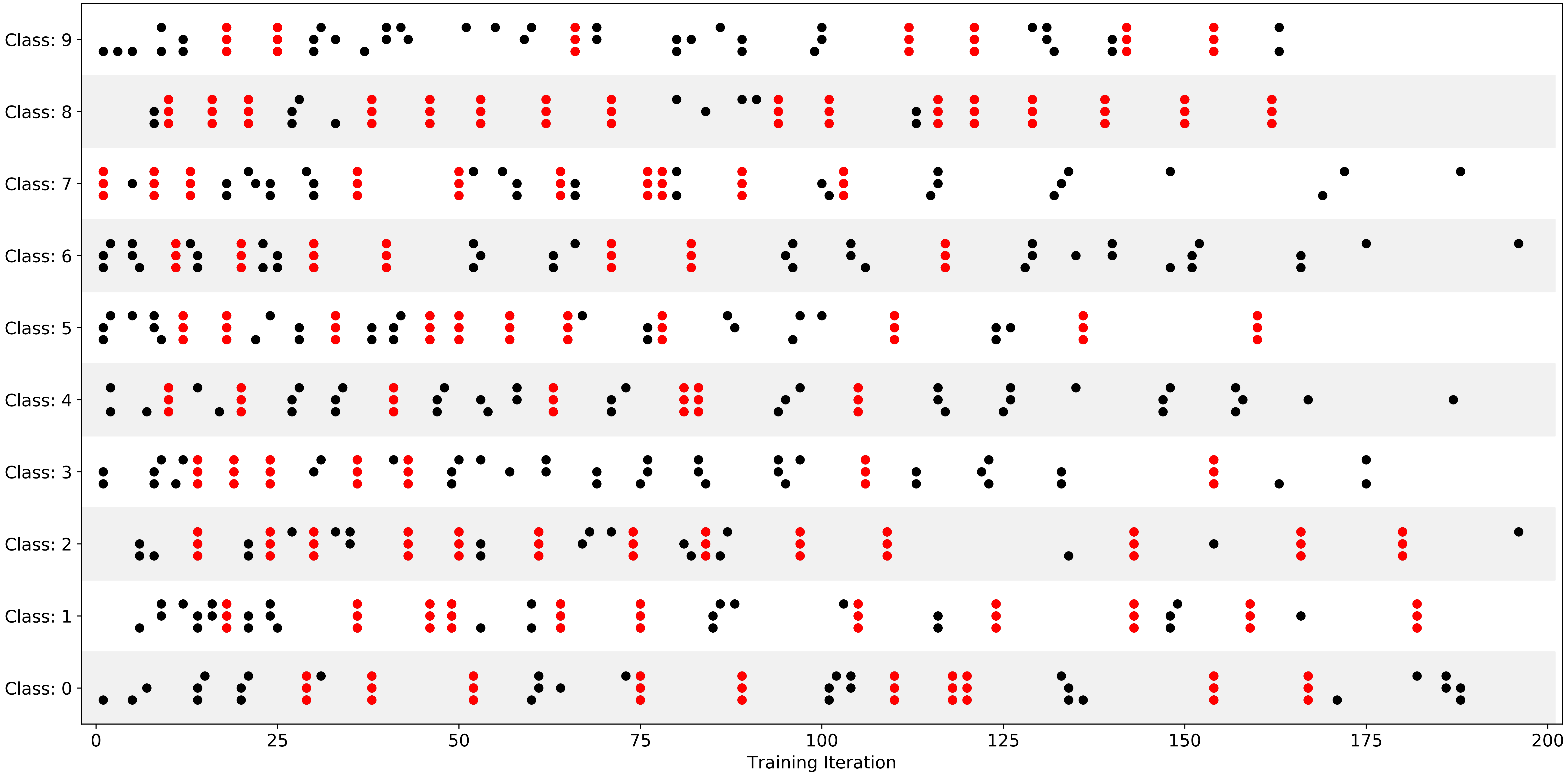
1 Training is noisy

2 Some layers go backwards

3 Some micro-learning is synchronized

MNIST-FC



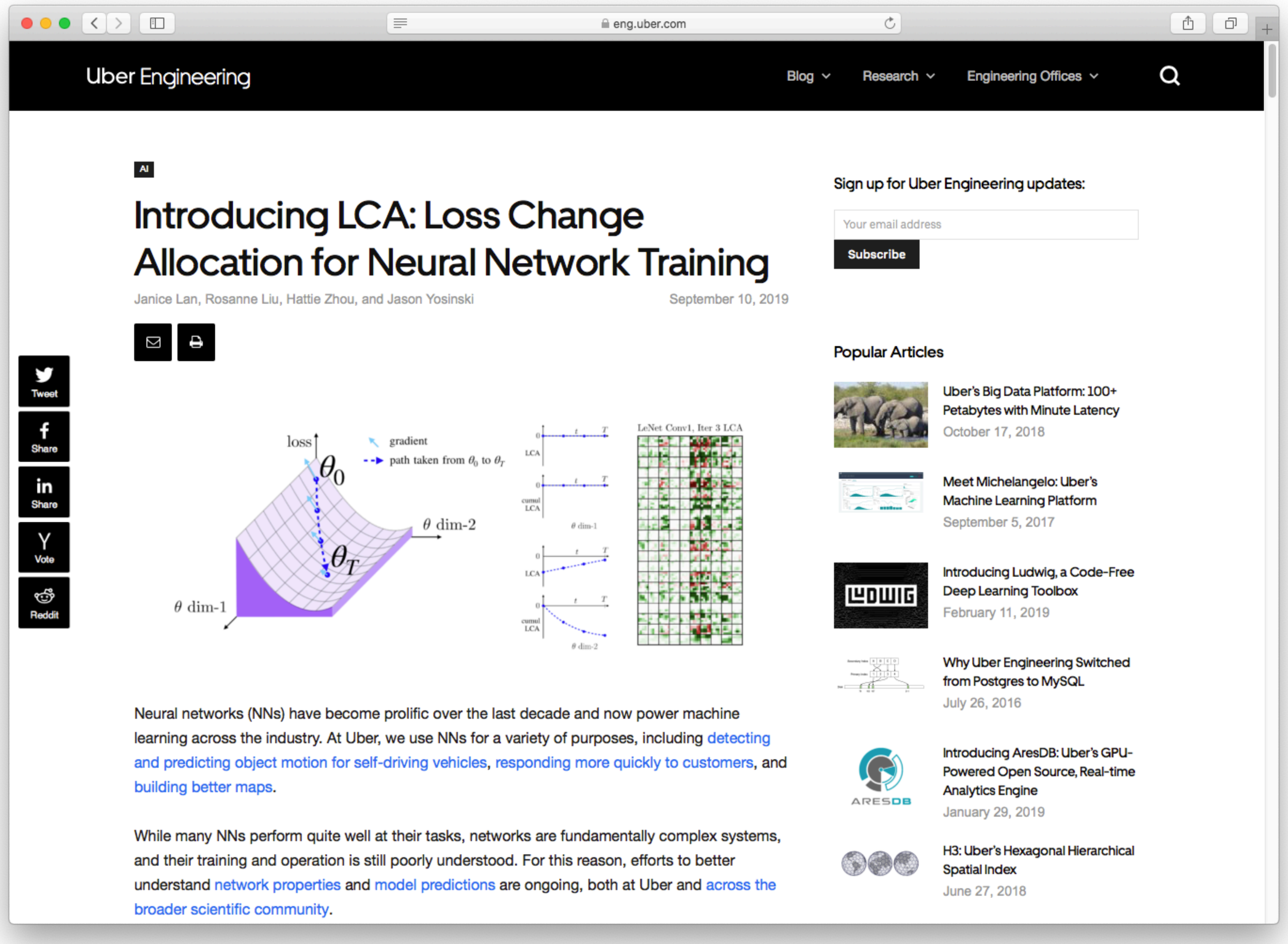



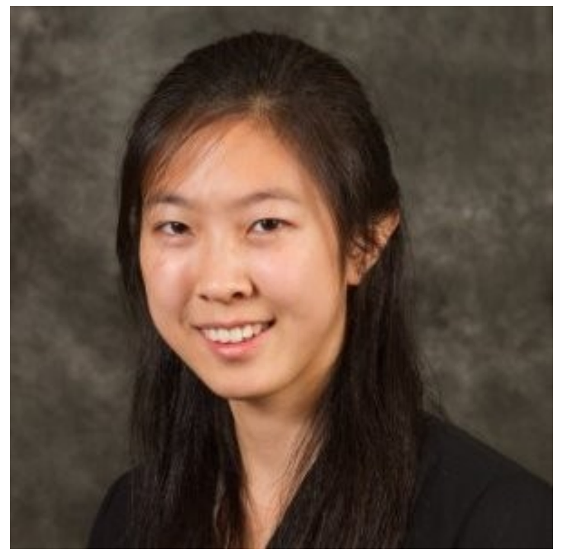

Janice Lan, Hattie Zhou, Rosanne Liu, Jason Yosinski. NeurIPS 2019.

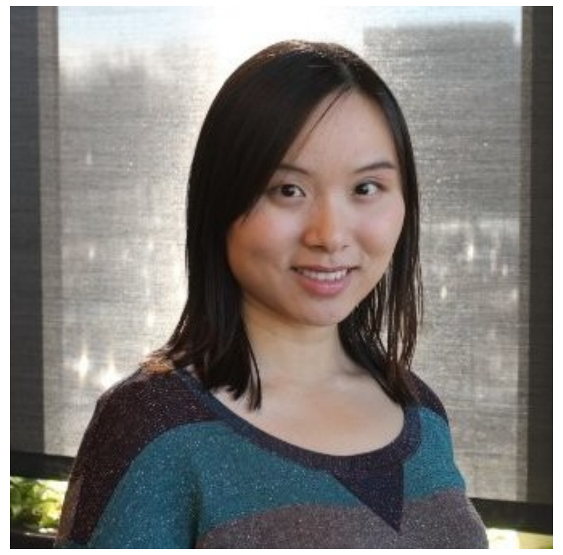
# LCA: Loss Change Allocation for Neural Network Training

NeurIPS 2019.

Blog: https://eng.uber.com/loss-change-allocation/





Janice Lan


Rosanne Liu


Hattie Zhou


Jason Yosinski