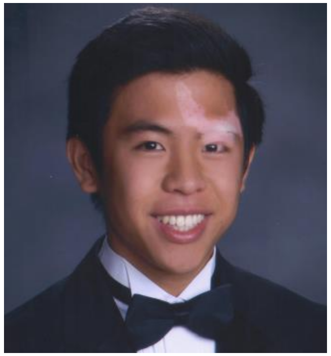
Interpreting & Manipulating Models via their Training Data

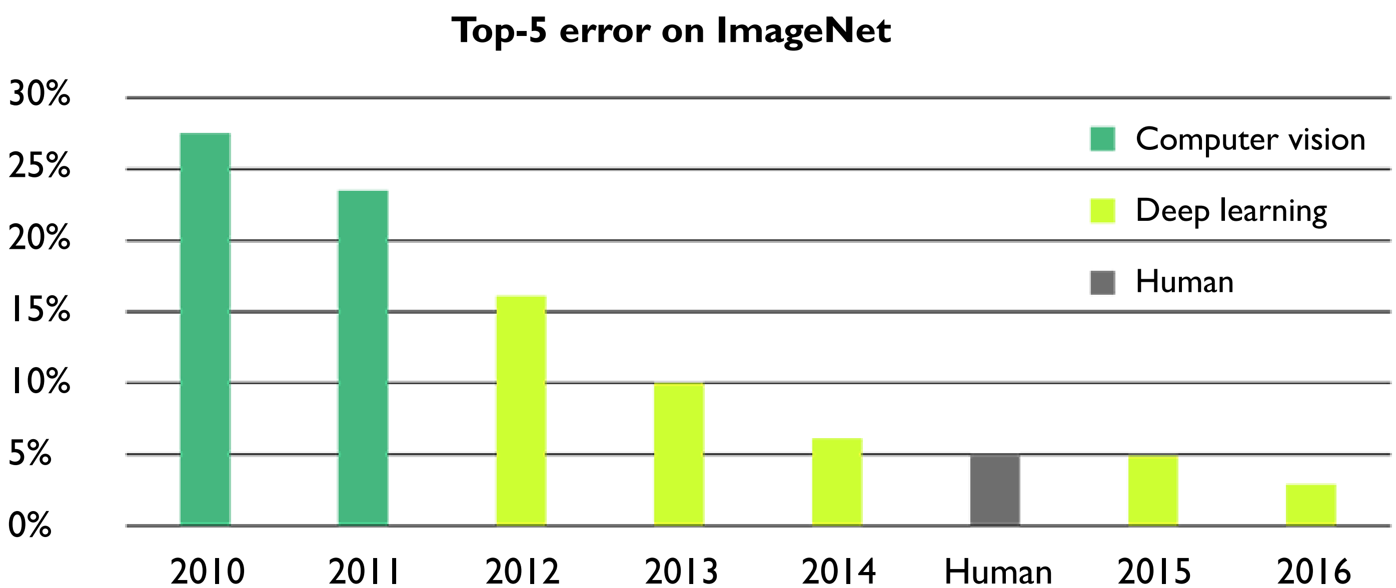
Pang Wei Koh
Stanford University


Kai-Siang Ang

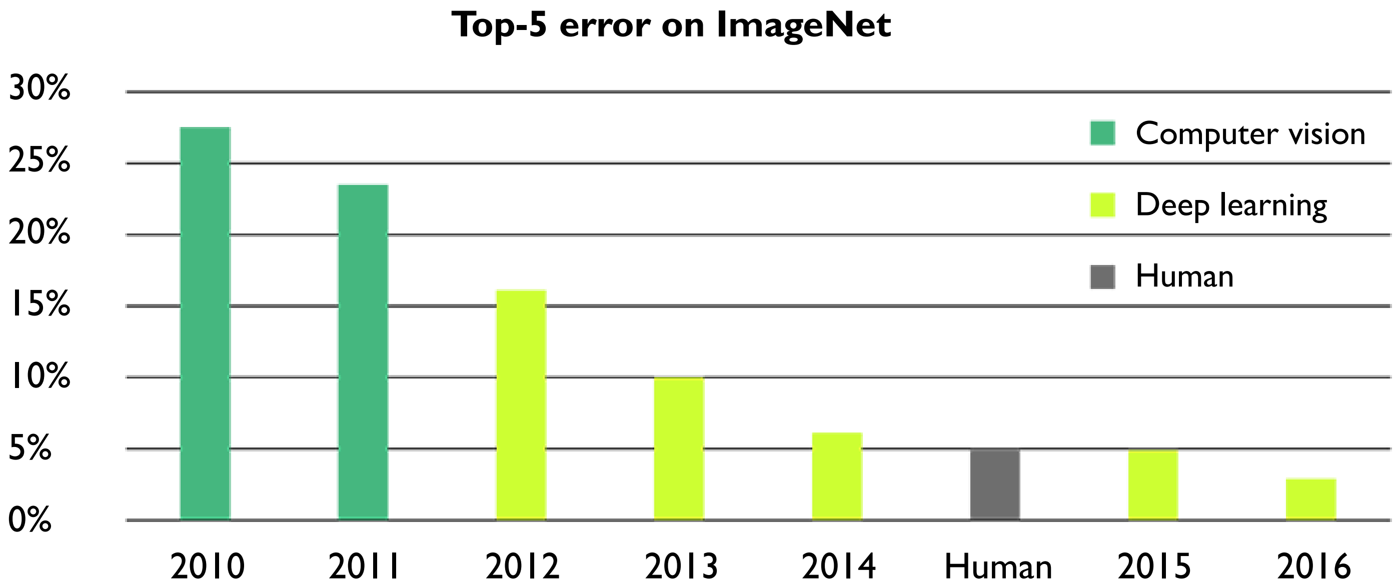

Hubert Teo

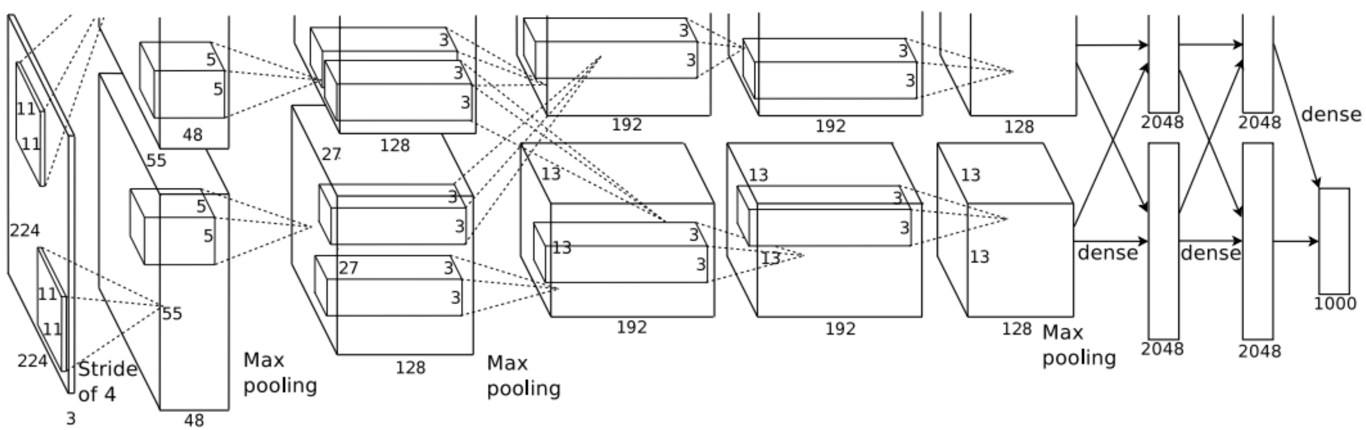

Jacob Steinhardt


Percy Liang



[1] Defense Systems Information Analysis Center





[1] Defense Systems Information Analysis Center
[2] Krizhevsky, Sutskever, and Hinton, 2012

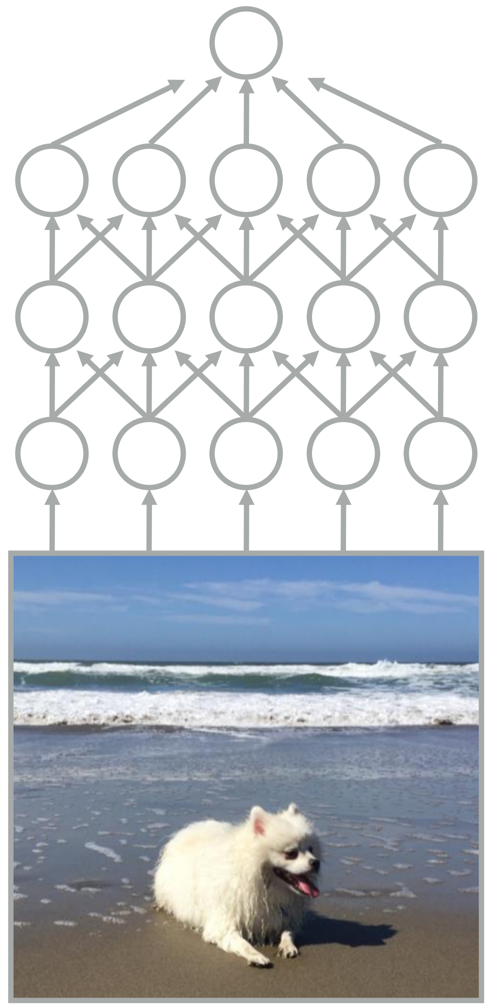
Why did the model make this prediction?

Why did the model make this prediction?

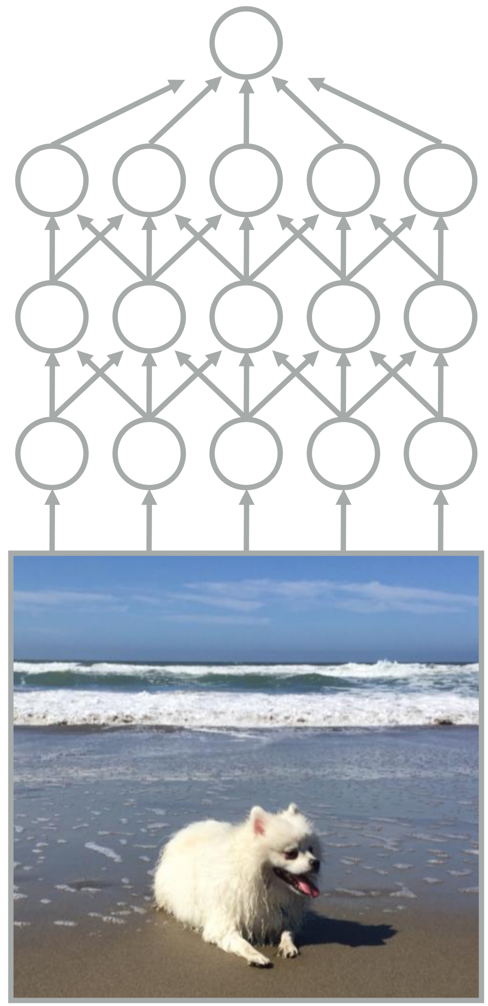
- Make better decisions [1]
- Improve the model [2]
- Discover new science [3]
- Provide end-users explanations [4]

[1] Lakkaraju, Bach, and Leskovec, 2016
[2] Amershi et al., 2015
[3] Shrikumar, Greenside, and Kundaje, 2017
[4] Goodman & Flaxman, 2016

“Dog”



"Dog"

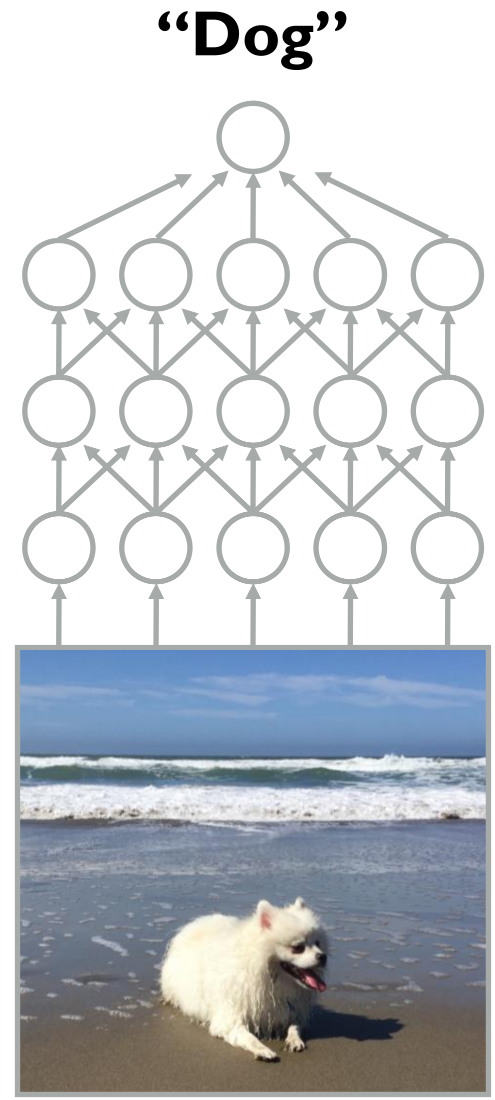


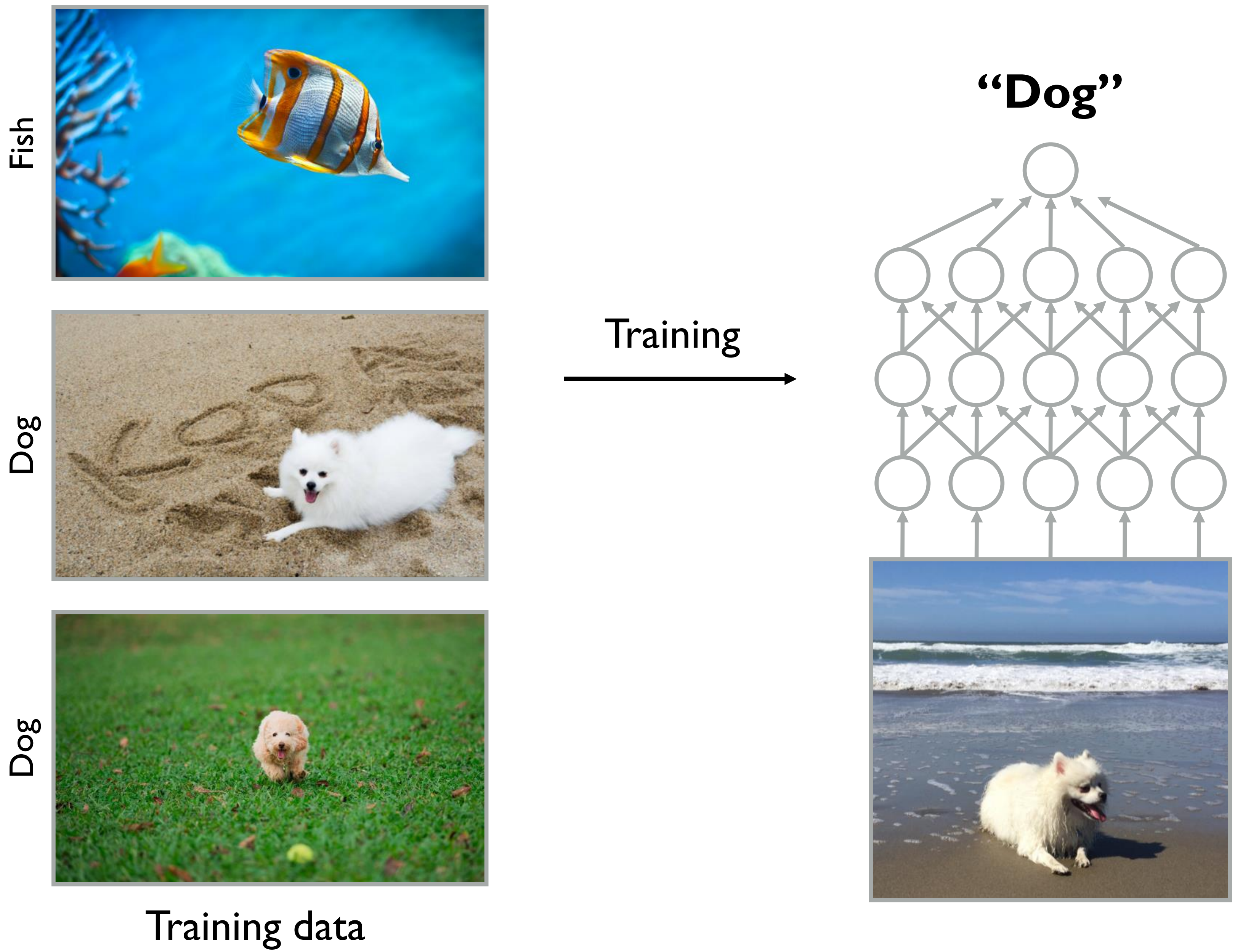
What inputs maximally activate these neurons? [1]

Can we represent this model with a simpler one? [6-7]

Which part of the input was most responsible for this prediction? [2-6]

[1] Girshick et al., 2014
[2] Zeiler and Fergus, 2013
[3] Simonyan, Vedaldi, and Zisserman, 2013
[4] Li, Monroe, and Jurafsky, 2016
[5] Shrikumar, Greenside, and Kundaje, 2017
[6] Strobel, Sliwinski , and Zick, 2018
[7] Ribeiro, Singh, and Guestrin, 2016
[8] Bastani, Kim, and Bastani, 2017


"Dog"



Fish
Dog
Dog
Training data
Training
"Dog"

Why did the model make this prediction?

Why did the model make this prediction?

Which training points were most responsible for this prediction?

Outline

I. The influence of individual training points

Outline

I. The influence of individual training points

II. The influence of groups of training points

The influence of individual training points

Koh & Liang, Understanding Black-box Predictions via Influence Functions, *ICML* 2017

Fish

Dog

Dog

Training data $z_1, z_2, \ldots, z_n$

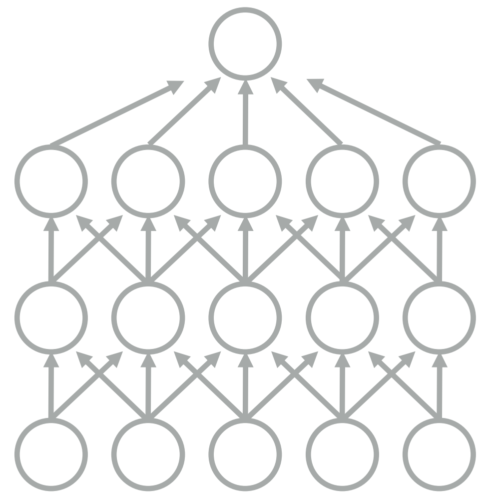
Fish

Dog

Dog

Training data $z_1, z_2, \ldots, z_n$

“Dog”



$\hat{\theta}$

Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$

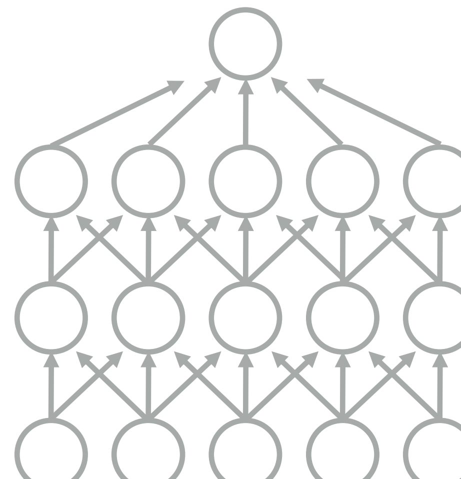
Fish

Dog

Dog

Training data $z_1, z_2, \ldots, z_n$

"Dog"



$\hat{\theta}$

Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$

Fish

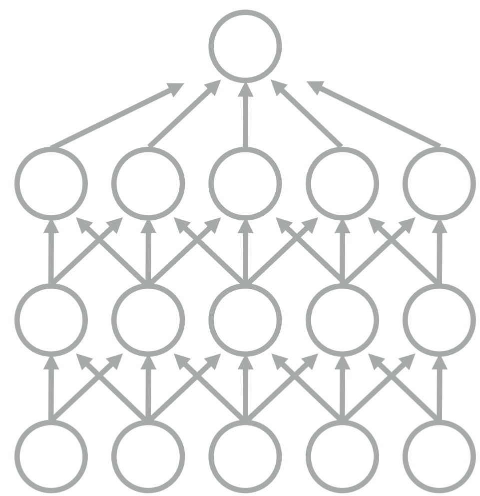
Dog

z_{train}

Dog

Training data $z_1, z_2, \ldots, z_n$

"Dog"



$\hat{\theta}$

Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$

Fish

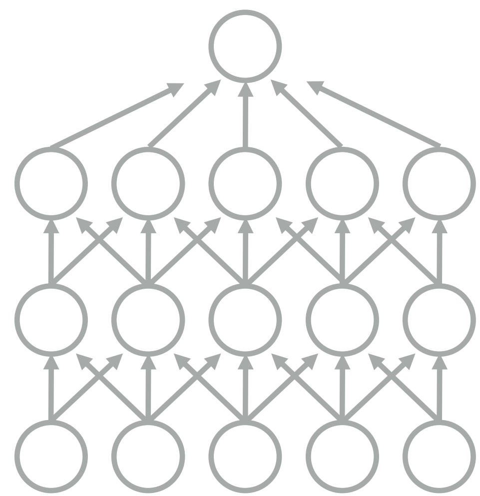
Dog

z_{train}

Dog

Training data $z_1, z_2, \dots, z_n$

“Dog”



$\hat{\theta}$

Pick $\hat{\theta}$ to minimize $\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta)$

Pick $\hat{\theta}_{-z_{train}}$ to minimize

$$\frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

Fish

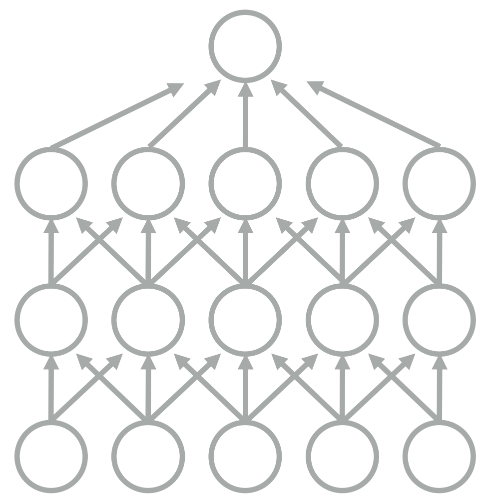
Dog

z_{train}

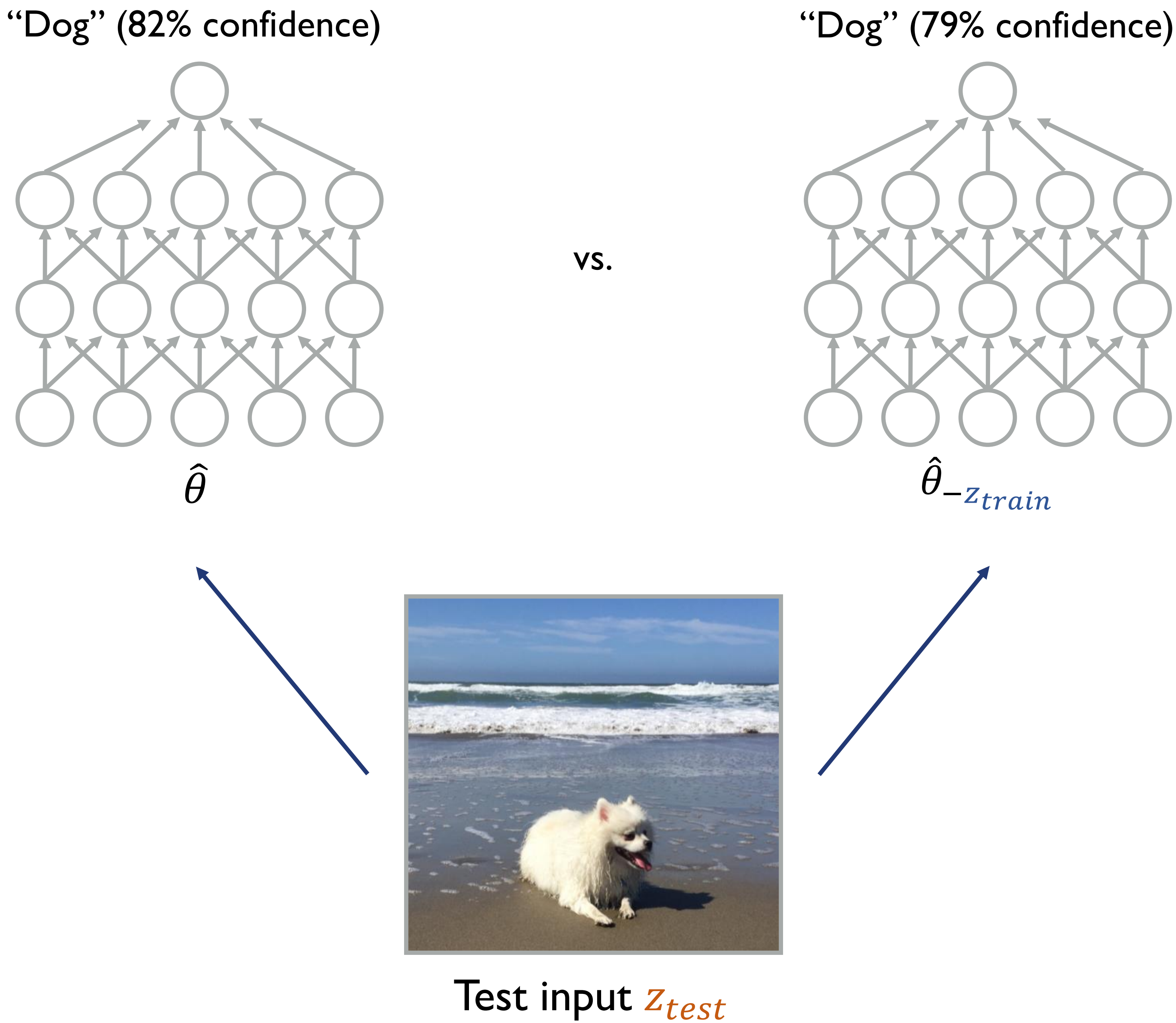
Dog

Training data $z_1, z_2, \ldots, z_n$

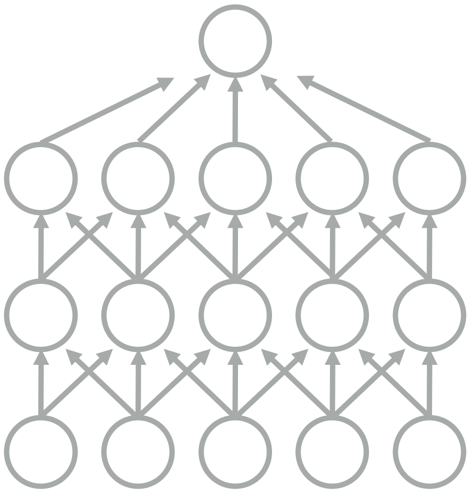
"Dog"



$\hat{\theta}_{-z_{train}}$


"Dog" (82% confidence)
"Dog" (79% confidence)
vs.
$\hat{\theta}$
$\hat{\theta}_{-z_{train}}$
Test input z_{test}

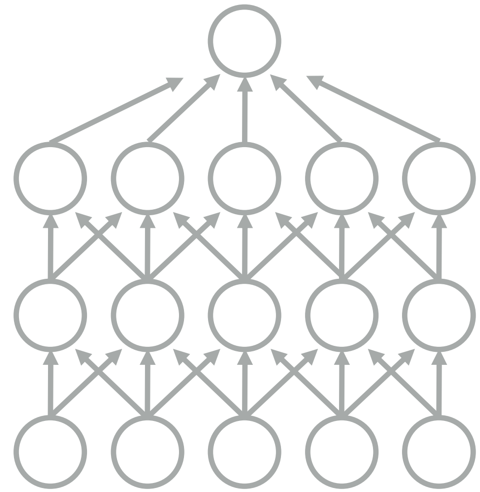
“Dog” (82% confidence)



$\hat{\theta}$

vs.

“Dog” (79% confidence)



$\hat{\theta}_{-z_{train}}$

What is $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right)$?

Why did the model make this prediction?

Which training points were most responsible for this prediction?

How would the prediction change if we removed a training point?

Problem

Repeatedly removing a training point and retraining the model is too slow

Problem

Repeatedly removing a training point and retraining the model is too slow

Solution

Approximation via influence functions (a classical technique from the 1970s)

Influence functions

- Goal: Compute $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right)$

$$\hat{\theta}_{-z_{train}} \stackrel{\text{def}}{=} \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

Influence functions

- Goal: Compute $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right)$

$$\hat{\theta}_{-z_{train}} \stackrel{\text{def}}{=} \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

- Equivalent to removing $\frac{1}{n}$ weight from z_{train} in the empirical distribution, then renormalizing

Influence functions

- Goal: Compute $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right)$

$$\hat{\theta}_{-z_{train}} \stackrel{\text{def}}{=} \operatorname{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

- Idea:
  - Assume $\frac{1}{n}$ is small
  - Use calculus to compute effect of removing ϵ weight from z_{train}
  - Linearly extrapolate to removing $\frac{1}{n}$ weight

Influence functions

- Goal: Compute $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right)$

$$\hat{\theta}_{-z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) - \frac{1}{n} L(z_{train}, \theta)$$

- Specifically, compute gradient of

$$\hat{\theta}_{\epsilon, z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z_{train}, \theta)$$

w.r.t. ϵ.

Influence functions

- $\hat{\theta}_{\epsilon,z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_{train},\theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left.\frac{dL\left(z_{test}, \hat{\theta}_{\epsilon,z_{train}}\right)}{d\epsilon}\right|_{\epsilon=0}$$

Influence functions

- $\hat{\theta}_{\epsilon, z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z_{train}, \theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left. \frac{dL\left(z_{test}, \hat{\theta}_{\epsilon, z_{train}}\right)}{d\epsilon} \right|_{\epsilon=0}$$

$$= -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{train}, \hat{\theta}\right)^\top$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L\left(z_i, \hat{\theta}\right)$.

Influence functions

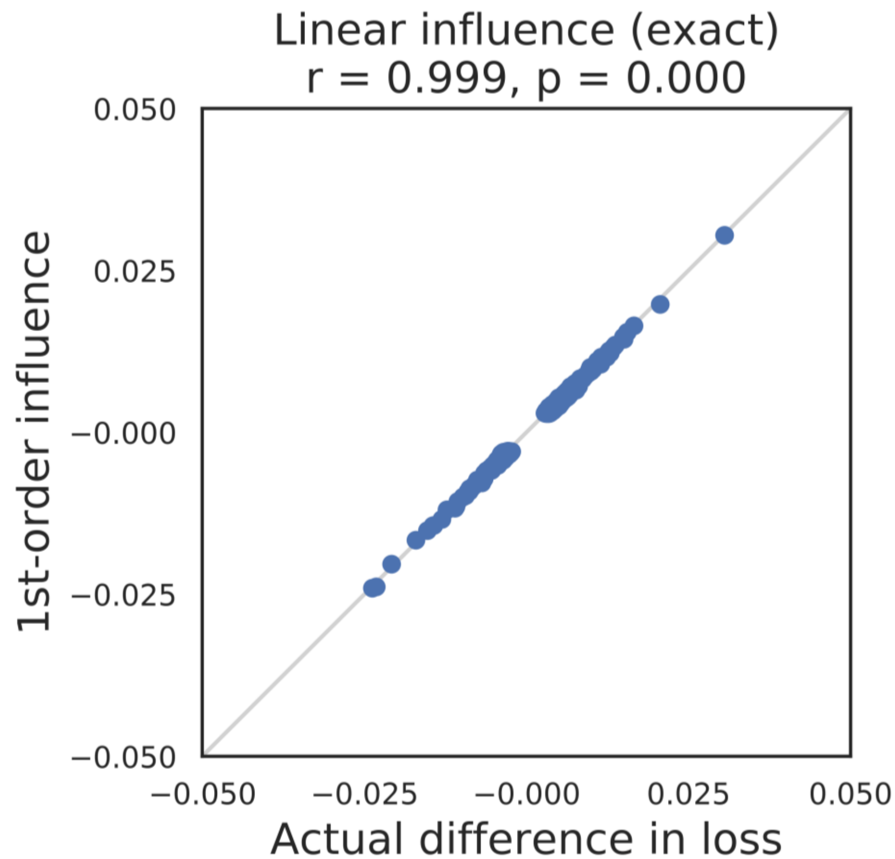
- $\hat{\theta}_{\epsilon,z_{train}} \stackrel{\text{def}}{=} \text{argmin}_{\theta\in\Theta} \frac{1}{n}\sum_{i=1}^{n} L(z_i,\theta) + \epsilon L(z_{train},\theta)$
- Under smoothness assumptions,

$$I_{up,loss}(z_{train}, z_{test}) \stackrel{\text{def}}{=} \left.\frac{dL\left(z_{test}, \hat{\theta}_{\epsilon,z_{train}}\right)}{d\epsilon}\right|_{\epsilon=0}$$

$$= -\nabla_\theta L\left(z_{test}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z_{train}, \hat{\theta}\right)^\top$$

where $H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L\left(z_i, \hat{\theta}\right)$.

- $L\left(z_{test}, \hat{\theta}_{-z_{train}}\right) - L\left(z_{test}, \hat{\theta}\right) = -\frac{1}{n} I_{up,loss}(z_{train}, z_{test})$



Linear influence (exact)
r = 0.999, p = 0.000
1st-order influence
Actual difference in loss
0.050
0.025
−0.000
−0.025
−0.050
−0.050
−0.025
−0.000
0.025
0.050

RBF SVM (raw pixels)

Logistic regression (Inception features)

Test image

Test image

RBF SVM (raw pixels)

Logistic regression (Inception features)

Other applications

- Debugging model errors
- Fixing training data

Influence Functions in Deep Learning are Fragile

Samyadeep Basu, Phillip Pope, Soheil Feizi

Department of Computer Science
University of Maryland, College Park

ICLR 2021

Outline

- Intro to Influence Functions
- Influence with Exact Hessian
- Influence for Shallow Architectures
- Influence for Deep Architectures
- Influence for ImageNet

Influence Functions

- First introduced in **robust statistics** by Jaeckel (1972), Cook(1977)
- From the context of ML, influence functions address the following question:
    - How do the **parameters** of a machine learning model **change** when the empirical weight distribution of the training samples are perturbed **infinitesimally**?

Prior Works on Influence Functions

Interpretability

- Koh et al. (2017) : Finding **influential samples.**
- Koh et al. (2019): Finding **influential group** of samples.
- Chen et al. (2019): Identification of **influential pre-training samples.**
- Brunet et al. (2018): Interpretation of **word-embeddings** with influence functions.

Prior Works on Influence Functions

Interpretability

- Koh et al. (2017) : Finding **influential samples.**
- Koh et al. (2019): Finding **influential group** of samples.
- Chen et al. (2019): Identification of **influential pre-training samples.**
- Brunet et al. (2018): Interpretation of **word-embeddings** with influence functions.

Uncertainty Estimation

- Schulam et al. (2019) : **Confidence intervals** for test-predictions using influence functions.
- Giordano et al. (2019): Infinitesimal jackknife for uncertainty estimation.
- Madras et al. (2019): **Extrapolation detection** using influence functions.

Prior Works on Influence Functions

Interpretability

- Koh et al. (2017) : Finding **influential samples.**
- Koh et al. (2019): Finding **influential group** of samples.
- Chen et al. (2019): Identification of **influential pre-training samples.**
- Brunet et al. (2018): Interpretation of **word-embeddings** with influence functions.

Uncertainty Estimation

- Schulam et al. (2019) : **Confidence intervals** for test-predictions using influence functions.
- Giordano et al. (2019): Infinitesimal jackknife for uncertainty estimation.
- Madras et al. (2019): **Extrapolation detection** using influence functions.

Other Applications

- Koh et al. (2019): Data **poisoning** attacks using influence functions.
- Cohen et al. (2020) : Detecting **adversarial examples** using influence functions.
- Wang et al. (2019): Influence functions for **improving model fairness.**

Influence Functions

$$\theta^* = \arg\min_{\theta \in \theta} \quad L_{\emptyset}(\theta) := \frac{1}{|\mathcal{S}|} \sum_{z \in \mathcal{S}} \ell(h_\theta(z))$$ → ERM

$$\theta^{\epsilon}_{\{z\}} = \arg\min_{\theta \in \theta} \quad \frac{1}{|\mathcal{S}|} \sum_{z \in \mathcal{S}} \ell(h_\theta(z_i)) + \epsilon \ell(h_\theta(z))$$ → ERM with **one** sample up-weighted

How to quantify $\Delta\theta = \theta^{\epsilon}_{\{z\}} - \theta^*$?

Influence Functions

Assumption: If the loss function is twice differentiable and **strictly convex** around θ^*

$$\theta^{\epsilon}_{\{z\}} \approx \theta^* - \epsilon H_{\theta^*}^{-1} \nabla_{\theta} \ell(h_{\theta^*}(z))$$

By **First-Order** Taylor's Approximation

$$\mathcal{I}(z) = \frac{d\theta^{\epsilon}_{\{z\}}}{d\epsilon}\Big|_{\epsilon=0} = -H_{\theta^*}^{-1} \nabla_{\theta} \ell\left(h_{\theta^*}(z)\right)$$

Change in model parameters when a training example is up-weighted by an **infinitesimal** amount

First-Order Influence Function

where $H_{\theta^*} = \nabla^2_{\theta} L_{\emptyset}(\theta)$

Influence Functions

How to quantify the **change in loss** for a **test-sample** z_t ?

$$\mathcal{I}(z, z_t) = -\nabla \ell(h_{\theta^*}(z_t))^T H_{\theta^*}^{-1} \nabla \ell(h_{\theta^*}(z))$$

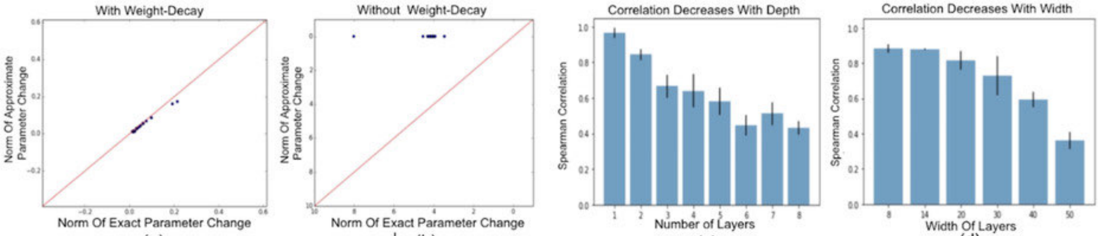
Using **Chain-Rule**

How do influence functions fare for **non-convex** loss functions ?

Issues for Influence Functions in Deep Learning

- Loss function is highly non-convex
- Different initialisations of the perturbed model can lead to different model parameters (with similar loss values)
- High values of loss curvature can widen the Taylor's gap leading to poor influence estimates
- Inverse Hessian-vector product (e.g. Stochastic Estimation) can be erroneous
- Ground-truth influence can be noisy for large models and datasets (e.g. ImageNet)

Influence Estimates with Exact Hessian



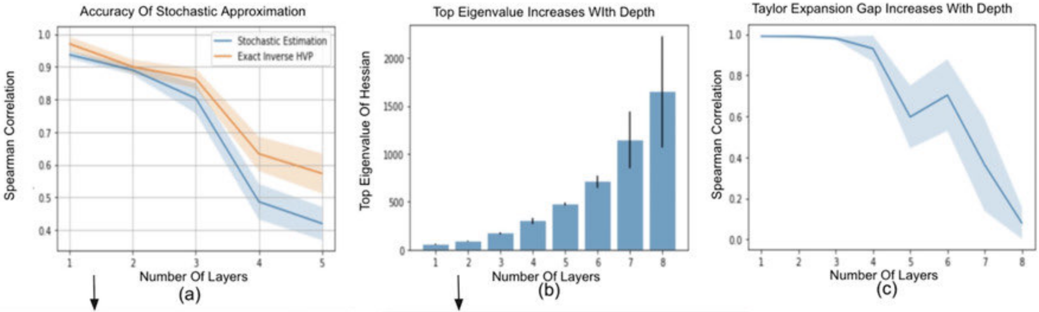
(a) (b) (c) (d)

Large Taylors gap

Overparameterization leads to erroneous influence estimates

Dataset: Iris

Influence Estimates with Exact Hessian

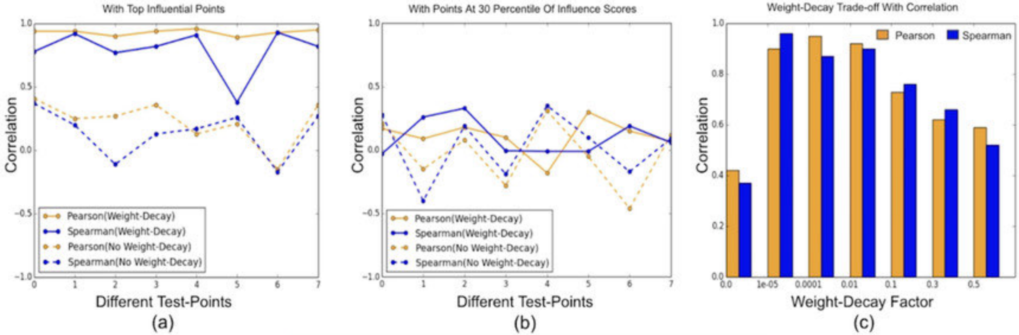


Stochastic Estimation is erroneous for large depths

High loss curvature degrades influence estimates

Dataset: Iris

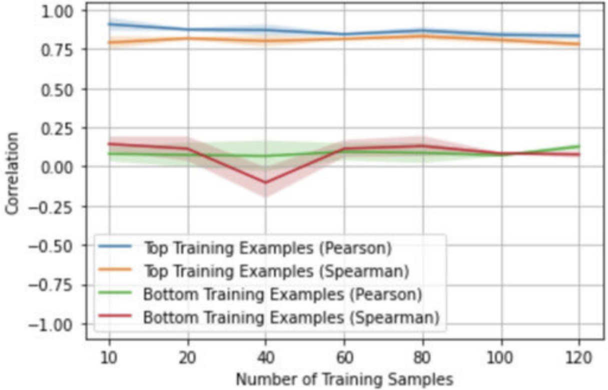
Influence Estimates for Shallow Architectures



Weight-decay is important for shallow architectures

Dataset: MNIST

Influence Estimates for Shallow Architectures



Influence function is accurate only when evaluated with the top influential samples

Dataset: MNIST

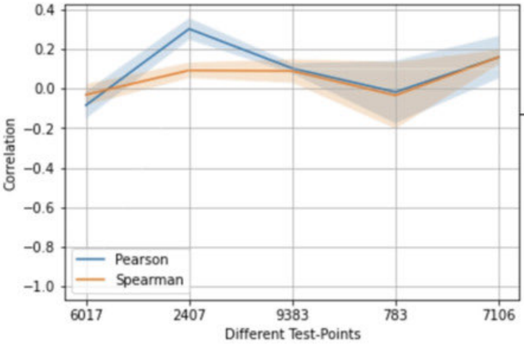
Influence Estimates for Deep Architectures

Dataset	MNIST						CIFAR-10					
	A (With Decay)		**B (With Decay)**		**A (Without Decay)**		**A (With Decay)**		**B (With Decay)**		**A (Without Decay)**	
Architecture	P	S	P	S	P	S	P	S	P	S	P	S
Small CNN	**0.95**	**0.87**	**0.92**	**0.82**	**0.41**	0.35	-	-	-	-	-	-
LeNet	**0.83**	**0.51**	0.28	0.29	0.18	0.12	**0.81**	**0.69**	0.45	0.46	0.19	0.09
VGG13	0.34	**0.44**	0.29	0.18	0.38	0.31	**0.67**	**0.63**	**0.66**	**0.63**	**0.79**	**0.73**
VGG14	0.32	0.26	0.28	0.22	0.21	0.11	**0.61**	**0.59**	**0.49**	0.41	**0.75**	**0.64**
ResNet18	**0.49**	0.26	0.39	0.35	0.14	0.11	**0.64**	**0.42**	0.25	0.26	**0.72**	**0.69**
ResNet50	0.24	0.22	0.29	0.19	0.08	0.13	**0.46**	0.36	0.24	0.09	0.32	0.14

Table 1: Correlation estimates on MNIST And CIFAR-10 ; A=Test-point with highest loss; B=Test-point at the 50^{th} percentile of test-loss spectrum; P=Pearson correlation; S=Spearman correlation

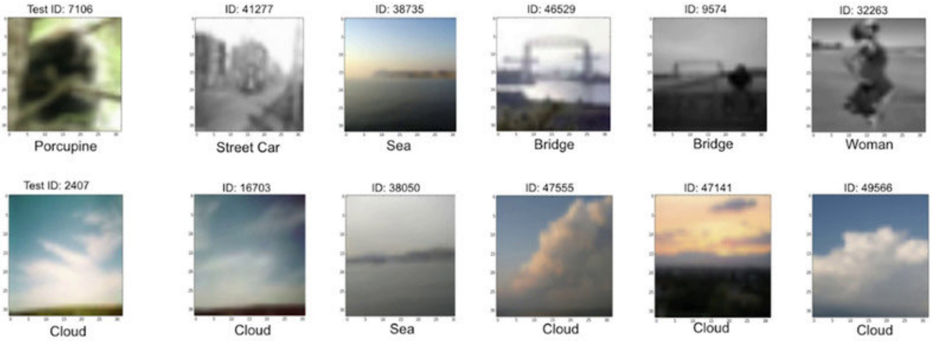
Influence estimates are erroneous for deeper networks such as ResNet-18/50

Influence Estimates for Deep Architectures



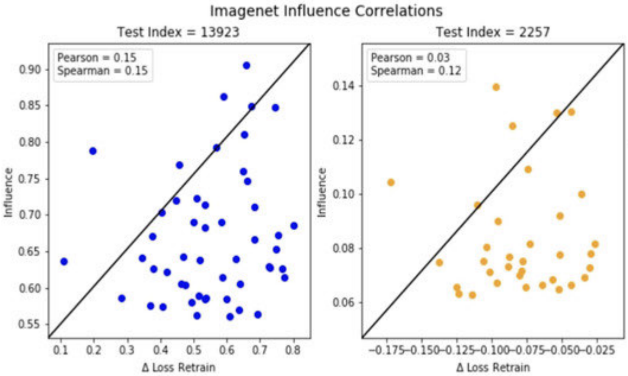
ResNet-18 + CIFAR-100 : Poor Influence Estimates

Influence Estimates for Deep Architectures



CIFAR-100: Inconsistent top influential points across different test-points

Influence Estimates for ImageNet



For ImageNet, ground-truth influence is also noisy

Future Directions

- Improvement in influence estimates by regularizing the loss curvature
- Theory for our empirical observations
- Make influence functions work for ImageNet and beyond !