# Understanding Black-box Predictions via Influence Functions

ICML 2017 Best Paper

#### Yufeng Ma

Digital Library Research Laboratory Computer Science Virginia Tech

yufengma@vt.edu

September 20, 2017

# Paper and Author Infos

#### Understanding Black-box Predictions via Influence Functions

#### Pang Wei Koh! Percy Liang

Abstract How can we explain the predictions of a blackbox model? In this paper, we use influence functions - a classic technique from robust statistics - to trace a model's prediction through the learning algorithm and back to its training data, thereby identifying training points most responsible for a given prediction. To scale up influence functions to modern machine learning settings. we develop a simple, efficient implementation that requires only oracle access to gradients and Messian vector products. We show that men on non-convex and non-differentiable models where the theory breaks down, approximations to influence functions can still provide valuable information. On linear models and convolutional neural networks, we demonstrate that influence functions are useful for multiple purposes: understanding model behavior, debuzging models, detecting dataset errors, and even creating visually-

#### 1. Introduction

A key question often asked of machine learning systems is "Why did the system make this prediction?" We want models that are not just high-performing but also explainable. By understanding why a model does what it does, we can hope to improve the model (Amershi et al., 2015), discover new science (Shrikumar et al., 2016), and provide end-users with explanations of actions that impact them (Goodman & Flaxman, 2016).

indistinguishable training-set attacks.

However, the best-performing models in many domains e.g., deep neural networks for image and speech recognition (Krizhevsky et al., 2012) - are complicated, blackbox models whose predictions seem hard to explain. Work on interpreting these black-box models has focused on undestanding how a fixed model leads to naticular profictions, e.g., by locally fitting a simpler model around the test

Starford University, Starford, CA. Correspondence to: Pane Wei Koh Crangwei@cs.stanford.edu>, Percy Liang Celiand the stanford edu >

Proceedings of the 34th International Conference on Machine Learning, Sydney, Assistalia, PMLR 70, 2017. Convenient 2017

point (Ribeiro et al., 2016) or by perturbing the test point to see how the prediction changes (Simonyan et al., 2013; Li et al., 2016b; Datta et al., 2016; Adler et al., 2016). These works explain the predictions in terms of the model, but how can we explain where the model came from?

In this paper, we tackle this question by tracing a model's predictions through its learning algorithm and back to the training data, where the model parameters ultimately derive from. To formalize the impact of a training point on a if we did not have this training point, or if the values of this training point were changed slightly?

Answering this question by perturbing the data and retraining the model can be prohibitively expensive. To overcome this problem, we use influence functions, a classic techniona from solver envirois (Cook & Weisham 1990) than tells us how the model parameters change as we upweight a training point by an infinitesimal amount. This allows us to "differentiate through the training" to estimate in closedform the effect of a variety of training perturbations.

Desnite their rich history in statistics, influence functions have not seen widespread use in machine learning; to the best of our knowledge, the work closest to ours is Woinowicz et al. (2016), which introduced a method for anproximating a quantity related to influence in generalized linear models. One obstacle to adoption is that influlations and assume model differentiability and convexity which limits their applicability in modern contexts where models are often non-differentiable, non-convex, and highdimensional. We address these challenees by showing that we can efficiently approximate influence functions using second-order optimization techniques (Pearlmatter, 1994) Martens, 2010: Assewal et al., 2016), and that they remain accurate even as the underlying assumptions of differentiability and convexity degrade.

Influence functions capture the core idea of studying models through the lens of their training data. We show that they are a versatile tool that can be applied to a wide variety of seemingly disparate tasks: understanding model behavior, debugging models, detecting dataset errors, and creatine visually-indistinguishable adversarial muisise examples that can flip neural network test predictions, the training set analogue of Goodfellow et al. (2015).

# Paper and Author Infos

#### Understanding Black-box Predictions via Influence Functions

#### Pang Wei Koh | Percy Liang

# Abstract How can we explain the predictions of a blackbox model? In this paper, we use influence funtions — a classic technique from robust statistos — to tree a model's prediction through the learning algorithm and book to its training data, thereby identifying tasking points most responsible for a given prediction. To scale up influence gractions to readom machine learning settlings,

fractions to medem machine learning settings, we develop a single, efficient implementation that requires only eracle access to gradients and Resistan-vector periods. We show that even on noe-courses and non-differentiable models where the textry breast down, approximations to influence functions and still provide valuable lafe-one functions are still provide valuable informations and approximation of the properties of the still provide valuable lafe-one functions are useful for multiple purposes: understanding model behavior, debugging models, do tecting dataset errores, and even creating visually-indictinguishable transitions extraction.

#### Y-4---

A key question often asked of machine learning systems in "Why did the system mach this prediction." We want models that are not just high-performing but also explainable. By understanding why an nodel does what it does not once the cover now science (Shrikkuara et al., 2016), and good once were now science (Shrikkuara et al., 2016), and good ond-users with explanations of actions that impact them (Goodman & Haranna, 2016).

However, the best-performing models in many domains og, deep nearal networks for integer and speech recognition (Krizhevsky et al., 2012) — are complicated, blackbox models whose predictions seem hand to explain. Werk on interpreting these black-box models has focused on understanding how a fland model leads to particular periodtions, e.g., by locally fitting a simpler model around the test "Seafard Integers based of CA. Commenders."

Stanford University, Stanford, CA. Correspondence to: Parg Wei Koh <pargwei@cs.stanford.edu>, Percy Liang <pli>pliang@cs.stanford.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Amendia, PMLR 70, 2017. Copyright 2017 by the author(s). point (Ribeiro et al., 2016) or by perturbing the test point to see how the prediction changes (Simoeyan et al., 2013; Li et al., 2016b; Datta et al., 2016; Adler et al., 2016). These works explain the predictions in terms of the model, but how can we explain where the model came from?

In this paper, we tackle this question by tracing a model's predictions through its learning algorithm and back to the training data, where the model parameters ultimately derive from. To formalize the impact of a training point on a prediction, we said the counterfeatual: what would happen if we did not have this training point, or if the values of this training point were changed slightly!

Answoring this question by perturbing the data and retraining the model can be predictably expensive. To oversee this problem, we use infrance functions a classic technique from cohost statistics (Code & Weisbeer, 1989) that talks as how the model parameters change as we upweight a takining point by an infinite-instant amount. This allows us to "differentiate through the training" to estimate in closed-form the effect of a warriery of maning perturbations.

Despite their rish bistory in statistics, influence functions more users witespread as in markonic tearing; in the best of our browledge, the work closes to our in View best of the browledge, the work closes to our in View propositionities; a quantity serious bis influence in generalized littleer models. One obtained to subspire in their influence functions require exposure second enterior conference functions require exposure second enterior conference functions are consequently as the conference which little models conference functions of the conference function of the conference function of the conference function of the conference functions exchange of the conference, 1994. Manness, 2010. A guernel or it, 2010, and the they remain course over an interior function of effects of a second order originations exhaulting of the conference functions of the conference functions of the conference function of the conference functio

Inflamen functions capture the core idea of studying models strough the low of their training data. We show that they are a versatile tool that can be applied to a wide writery of seemingly disporate tasks: understanding model behavier, debugging models, desecting dataset curves, and crataining visually-indictinguishables adversarial motisting examples that can fijn neural network test predictions, the traintim set analouse of Goodfellows or al. (2015).

#### ICML 2017 Best Paper Award

The 34th International Conference on Machine Learning

- Pang Wei Koh
- Percy Liang

# Paper and Author Infos

#### Understanding Black-box Predictions via Influence Functions

#### Pang Wei Koh | Percy Liang

# Abstract How can we explain the predictions of a blackbox model? In this paper, we use influence functions — a classic technique from robust statistics — to trace a model's prediction through the learning algorithm and back to its training data,

thereby identifying training points most responsible for a given profession. To sake up influence functions to modern machine, difficient implementation we develop a simple, difficient implementation was developed as the profession of the profess

#### . Introduction

A key question often asked of machine learning systems in "Why did the system mach this prediction." We want models that are not just high-performing but also explainable. By understanding why an nodel does what it does not once the cover now science (Shrikkuara et al., 2016), and good once were now science (Shrikkuara et al., 2016), and good ond-users with explanations of actions that impact them (Goodman & Haranna, 2016).

However, the best-performing models in many domains — e.g., deep neural networks for irrange and speech recognition (Krizhevsky et al., 2012) — are complicated, black-bots models whose predicties seen hard to explain. Work on interpreting here black bots models has focused on interpreting here black bots model has focus on interpreting here black bots model amount the test focus of the black blac

ang@co.stanford.edu>

Proceedings of the 34<sup>th</sup> International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Convenient 2017

point (Ribeiro et al., 2016) or by perturbing the test point to see how the prediction changes (Simonyan et al., 2013; Li et al., 2016); Data et al., 2016; Adler et al., 2016). These works explain the predictions in terms of the model, but how can we explain where the model came front?

In this paper, we tackle this question by tracing a model's predictions through its learning algorithm and back to the training data, where the model parameters ultimately derive from. To formalize the impact of a training point on a prediction, we sait the counterbranal: what would happen if we did not have this training point, or if the values of this training point were changed slightly?

Assworing this question by perturbing the data and retraining the model can be predictable; expensive. To overage mentals problem, we use infrance functions a classic technique from cohost satisfies (Code. & Weisbeer, 1989) that tells us how the model parameters change as we expresight a tabilising point by an infinite-instal amount. This allows us to "differentiate through the training" to estimate in closed-form the effect of a warriery of maning perturbations.

Despite their rich bistory in statistics, of these frauctions me state was desperad us in machine learning; to the best of our is broaded, the work closes to our is Month or the state of the state of

Inflamen functions capture the core idea of studying medch strough the less of their mixing data. We show that they are a certain not little at an he applied to a side writery of seemingly disporter tooks: understanding model behavior, debegging models, detecting dataset errors, and cretaing visually-indicinguishable adversarial motistic geamples that can fip neural netweek test predictions, the trainties set analouse of Glosdifichow et al. (2015).

#### ICML 2017 Best Paper Award

The 34th International Conference on Machine Learning

- Pang Wei Koh
- Percy Liang

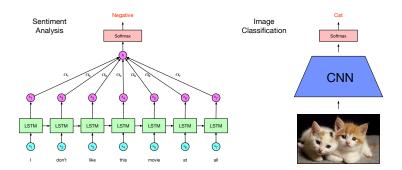
#### Affiliations:

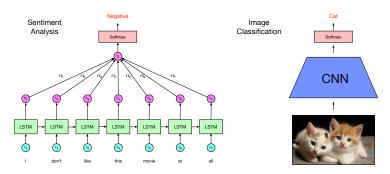
Computer Science, Stanford University

- Artificial Intelligence Lab
- Natural Language Processing Group
- Statistical Machine Learning Group

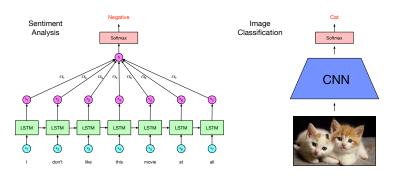
#### Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- Influence Calculation
- Validation and Extensions
- 6 Applications





- Why did neural networks make such predictions?
- Where does the model come from?
- Can we explain predictions in terms of training points?



- Why did neural networks make such predictions?
- Where does the model come from?
- Can we explain predictions in terms of training points?

#### Motivation

What's the influence of a training point to the prediction of test sample?

### **Empirical Risk Loss**

Given a prediction problem:  $\mathcal{X}$  (images)  $\rightarrow \mathcal{Y}$  (labels)

$$R(\theta) \stackrel{\mathsf{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

### Empirical Risk Loss

Given a prediction problem:  $\mathcal{X}$  (images)  $\rightarrow \mathcal{Y}$  (labels)

$$R(\theta) \stackrel{\mathsf{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

- $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y};$
- $\theta \in \Theta$ : parameter space;
- $L(z,\theta)$ : loss function, *e.g.*, cross entropy.

### Empirical Risk Loss

Given a prediction problem:  $\mathcal{X}$  (images)  $\rightarrow \mathcal{Y}$  (labels)

$$R(\theta) \stackrel{\mathsf{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

- $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y};$
- $\theta \in \Theta$ : parameter space;
- $L(z,\theta)$ : loss function, *e.g.*, cross entropy.

Goal: 
$$\hat{\theta} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\operatorname{arg \, min}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$



# Empirical Risk Loss

Given a prediction problem:  $\mathcal{X}$  (images)  $\rightarrow \mathcal{Y}$  (labels)

$$R(\theta) \stackrel{\mathsf{def}}{=} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

- $z_i = (x_i, y_i) \in \mathcal{X} \times \mathcal{Y};$
- $\theta \in \Theta$ : parameter space;
- $L(z,\theta)$ : loss function, *e.g.*, cross entropy.

**Goal:** 
$$\hat{\theta} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\operatorname{arg \, min}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta)$$

ullet Assumption: R( heta) is twice-differentiable and strictly convex.



#### Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- Influence Calculation
- 4 Validation and Extensions
- 6 Applications

#### Question 1:

How would predictions change if we did not have a specific training point?

#### Question 1:

How would predictions change if we did not have a specific training point?

### Influence Function Approximation

$$\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\operatorname{arg \, min}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta)$$

$$\mathcal{I}_{\text{up, params}}(z) \, \stackrel{\text{def}}{=} \, \frac{d \hat{\theta}_{\epsilon,z}}{d_{\epsilon}} \Bigg|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z,\hat{\theta})$$

#### Question 1:

How would predictions change if we did not have a specific training point?

### Influence Function Approximation

$$\hat{\theta}_{\epsilon,z} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\operatorname{arg \, min}} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta)$$

$$\mathcal{I}_{\text{up, params}}(z) \stackrel{\text{def}}{=} \left. \frac{d\hat{\theta}_{\epsilon,z}}{d_{\epsilon}} \right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z,\hat{\theta})$$

# Influence of training point z to $z_{\text{test}}$

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) \, \stackrel{\text{def}}{=} \, \left. \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon, z})}{d\epsilon} \right|_{\epsilon = 0} = \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^{\top} \left. \frac{d\hat{\theta}_{\epsilon, z}}{d\epsilon} \right|_{\epsilon = 0}$$

#### Question 2:

How would the predictions change if a specific training point was modified?

#### Question 2:

How would the predictions change if a specific training point was modified?

Influence to 
$$\theta$$
 with  $z=(x,y)$  replaced by  $z_{\delta}=(x+\delta,y)$ 

#### Question 2:

How would the predictions change if a specific training point was modified?

Influence to 
$$\theta$$
 with  $z=(x,y)$  replaced by  $z_{\delta}=(x+\delta,y)$ 

$$\hat{\theta}_{\epsilon,z_{\delta},-z} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\arg \min} \frac{1}{n} \sum_{i=1}^{n} L(z_{i},\theta) + \epsilon L(z_{\delta},\theta) - \epsilon L(z,\theta)$$

$$\frac{d\hat{\theta}_{\epsilon,z_{\delta},-z}}{d\epsilon} \bigg|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} (\nabla_{\theta} L(z_{\delta},\hat{\theta}) - \nabla_{\theta} L(z,\hat{\theta}))$$

$$\approx -H_{\hat{\theta}}^{-1} [\nabla_{x} \nabla_{\theta} L(z,\hat{\theta})] \delta$$

#### Question 2:

How would the predictions change if a specific training point was modified?

Influence to 
$$\theta$$
 with  $z=(x,y)$  replaced by  $z_{\delta}=(x+\delta,y)$ 

$$\hat{\theta}_{\epsilon,z_{\delta},-z} \stackrel{\text{def}}{=} \underset{\theta \in \Theta}{\arg \min} \frac{1}{n} \sum_{i=1}^{n} L(z_{i},\theta) + \epsilon L(z_{\delta},\theta) - \epsilon L(z,\theta)$$

$$\frac{d\hat{\theta}_{\epsilon,z_{\delta},-z}}{d\epsilon} \bigg|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} (\nabla_{\theta} L(z_{\delta},\hat{\theta}) - \nabla_{\theta} L(z,\hat{\theta}))$$

$$\approx -H_{\hat{\theta}}^{-1} [\nabla_{x} \nabla_{\theta} L(z,\hat{\theta})] \delta$$

#### Influence to $z_{\mathrm{test}}$ with z replaced by $z_{\delta}$

$$\mathcal{I}_{\mathsf{pert},\mathsf{loss}}(z,z_{\mathsf{test}}) \, \stackrel{\mathsf{def}}{=} \, -\nabla_{\theta} L(z_{\mathsf{test}},\hat{\theta})^{\top} H_{\hat{\theta}}^{-1} \nabla_x \nabla_{\theta} L(z,\hat{\theta})$$

#### Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- Influence Calculation
- 4 Validation and Extensions
- 6 Applications

# Challenges

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^{\top} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

## Challenges

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^{\top} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

- $H_{\hat{\theta}}=rac{1}{n}\sum_{i=1}^{n}\nabla_{\theta}^{2}L(z_{i},\hat{\theta}), \theta\in R^{p}$  : inversion takes  $O(np^{2}+p^{3})$ ;
- $\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$  for all training points  $z_i$ ;

# Challenges

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^{\top} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

- $H_{\hat{\theta}}=rac{1}{n}\sum_{i=1}^{n}\nabla_{\theta}^{2}L(z_{i},\hat{\theta}), \theta\in R^{p}$  : inversion takes  $O(np^{2}+p^{3})$ ;
- $\mathcal{I}_{\text{up,loss}}(z_i, z_{\text{test}})$  for all training points  $z_i$ ;

# Hessian-Vector Products (HVPs)

$$s_{\text{test}} \stackrel{\text{def}}{=} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})$$

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_{\theta} L(z_i, \hat{\theta})$$

# Challenges

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_{\theta} L(z_{\text{test}}, \hat{\theta})^{\top} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z, \hat{\theta})$$

- $H_{\hat{\theta}}=rac{1}{n}\sum_{i=1}^n 
  abla_{\theta}^2 L(z_i,\hat{\theta}), \theta \in R^p$  : inversion takes  $O(np^2+p^3)$ ;
- $\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$  for all training points  $z_i$ ;

# Hessian-Vector Products (HVPs)

$$s_{\text{test}} \stackrel{\mathsf{def}}{=} H_{\hat{\theta}}^{-1} \nabla_{\theta} L(z_{\text{test}}, \hat{\theta})$$

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -s_{\text{test}} \cdot \nabla_{\theta} L(z_{i}, \hat{\theta})$$

#### Solving $s_{ m test}$

- Conjugate Gradients (CG):  $\arg\min_t \{\frac{1}{2}t^\top H_{\hat{\theta}}t s_{\text{test}}^\top t\};$
- Stochastic estimation: speed up CG;

#### Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- 3 Influence Calculation
- 4 Validation and Extensions
- 6 Applications

### Validity of Influence Function

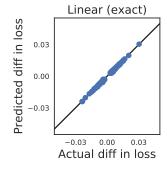
$$-\frac{1}{n}\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}})$$
 vs.  $L(z_{\text{test}}, \hat{\theta}_{-z}) - L(z_{\text{test}}, \hat{\theta})$ 

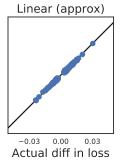
Influence function vs. leave-one-out retraining

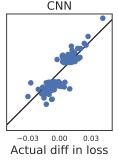
#### Validity of Influence Function

$$-\frac{1}{n}\mathcal{I}_{\mathrm{up,loss}}(z, z_{\mathrm{test}})$$
 vs.  $L(z_{\mathrm{test}}, \hat{\theta}_{-z}) - L(z_{\mathrm{test}}, \hat{\theta})$ 

• Influence function vs. leave-one-out retraining







## Validation and Extensions

#### Extensions

- Non-convexity and non-convergence: quadratic approximation
  - $\bullet \ \widetilde{L}(z,\theta) = L(z,\widetilde{\theta}) + \nabla L(z,\widetilde{\theta})^\top (\theta \widetilde{\theta}) + \frac{1}{2} (\theta \widetilde{\theta})^\top (H_{\widetilde{\theta}} + \lambda I) (\theta \widetilde{\theta})$
- Non-differentiable loss: replaced with smoothing version
  - Hinge(s) =  $\max(0, 1 s) \to \text{SmoothHinge}(s, t) = t \log(1 + \exp(\frac{1 s}{t}))$

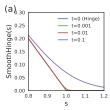
## Validation and Extensions

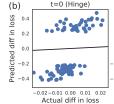
#### Extensions

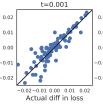
Non-convexity and non-convergence: quadratic approximation

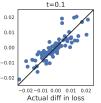
$$\bullet \ \widetilde{L}(z,\theta) = L(z,\widetilde{\theta}) + \nabla L(z,\widetilde{\theta})^\top (\theta - \widetilde{\theta}) + \frac{1}{2} (\theta - \widetilde{\theta})^\top (H_{\widetilde{\theta}} + \lambda I) (\theta - \widetilde{\theta})$$

- Non-differentiable loss: replaced with smoothing version
  - Hinge(s) =  $\max(0, 1 s) \rightarrow \text{SmoothHinge}(s, t) = t \log(1 + \exp(\frac{1 s}{t}))$









## Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- Influence Calculation
- 4 Validation and Extensions
- 6 Applications

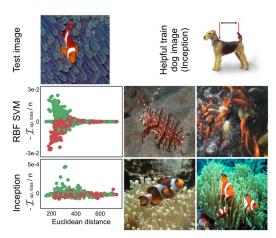
• Model Behavior Understanding:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$ 

• Model Behavior Understanding:  $-\mathcal{I}_{\text{up,loss}}(z_i, z_{\text{test}})$ 

RBF SVM vs. Inception for Fish/Dog Classification

ullet Model Behavior Understanding:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$ 

RBF SVM vs. Inception for Fish/Dog Classification



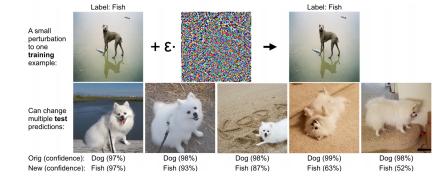
ullet Adversarial training examples:  $\mathcal{I}_{\mathrm{pert,loss}}(z, z_{\mathrm{test}})$ 

ullet Adversarial training examples:  $\mathcal{I}_{\mathrm{pert,loss}}(z, z_{\mathrm{test}})$ 

$$\widetilde{z}_i := \prod (\widetilde{z}_i + \alpha \operatorname{sign}(\mathcal{I}_{\operatorname{pert}, \operatorname{loss}}(\widetilde{z}_i, z_{\operatorname{test}})))$$

ullet Adversarial training examples:  $\mathcal{I}_{\mathrm{pert,loss}}(z,z_{\mathrm{test}})$ 

$$\widetilde{z}_i := \prod (\widetilde{z}_i + \alpha \operatorname{sign}(\mathcal{I}_{\operatorname{pert}, \operatorname{loss}}(\widetilde{z}_i, z_{\operatorname{test}})))$$



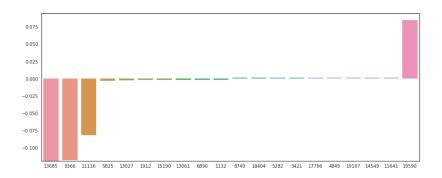
• Debugging domain mismatch:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$ 

• Debugging domain mismatch:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i, z_{\mathrm{test}})$ 

Patients re-admission classification children re-admitted: 3/24 changed to 3/4

ullet Debugging domain mismatch:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i,z_{\mathrm{test}})$ 

Patients re-admission classification children re-admitted: 3/24 changed to 3/4



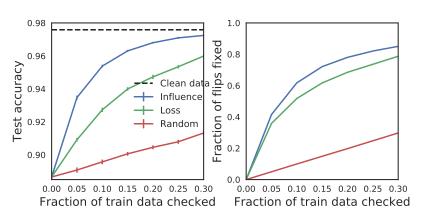
ullet Fixing mislabeled examples:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i,z_i)$ 

ullet Fixing mislabeled examples:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i,z_i)$ 

Email spam classification: 10% labels flipped

ullet Fixing mislabeled examples:  $-\mathcal{I}_{\mathrm{up,loss}}(z_i,z_i)$ 

Email spam classification: 10% labels flipped



## Overview

- Introduction
- 2 Approach
  - Upweighting a training point
  - Perturbing a training point
- Influence Calculation
- 4 Validation and Extensions
- 6 Applications

# Questions

# Questions?