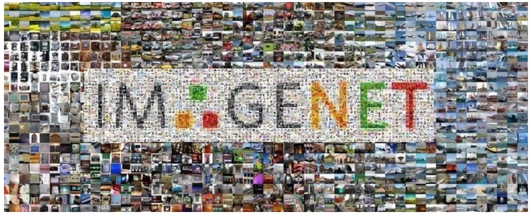
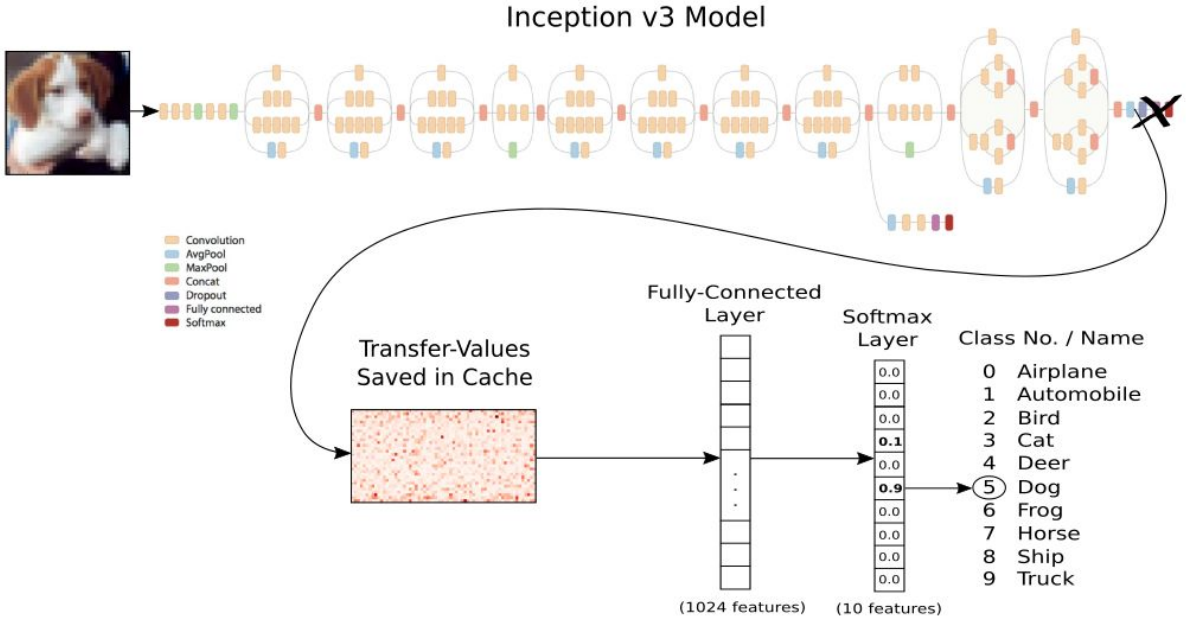
Understanding Black-box Predictions via Influence Functions

Pang Wei Koh, Percy Liang

As narrated by: Dany Haddad, Alan Gee

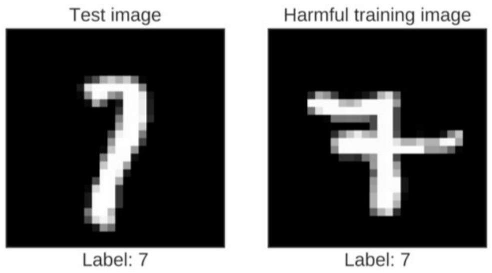
Big Picture





How can we explain where results of the model came from?

Influence Functions



Main Idea: quantify the influence of a training sample on the loss at a test sample. Not actually a black-box method.

Applications:

- Determine which training examples are most beneficial or harmful to test set performance
- Prioritizes which training set points to check for errors
- Generating adversarial *training* examples

Efficiency:

- No need to retrain the entire model

Assumptions

- Strictly convex and twice differentiable objective function
- What if these assumptions don’t hold?
    - Can form a local strictly convex quadratic approximation

$$L(z,\theta) \approx L(z,\tilde{\theta}) + \nabla L(z,\theta)^T(\theta - \tilde{\theta}) + \frac{1}{2}(\theta - \tilde{\theta})^T(H_{\tilde{\theta}} + \lambda I)(\theta - \tilde{\theta})$$

    - For the Hinge loss case they propose to compute the Influence function through a smooth hinge loss term

$$\text{SmoothHinge}(s,t) = t\log(1+\exp(\tfrac{1-s}{t}))$$

Influence Functions: Mathematical Formulation

$$z_i = (x_i, y_i) \qquad \hat{\theta} = argmin_\theta \frac{1}{n}\sum_i^n L(z_i, \theta)$$

Upweight the influence of z_j, by some ε. The new model parameters are:

$$\hat{\theta}_{\epsilon,z} = argmin_\theta \frac{1}{n}\sum_i^n L(z_i, \theta) + \epsilon L(z_j, \theta)$$

This has an influence on the parameters given by [Cook & Weisberg, 1982]:

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

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \tfrac{1}{n}\textstyle\sum_{i=1}^n \nabla_\theta^2 L(z_i, \hat{\theta})$$

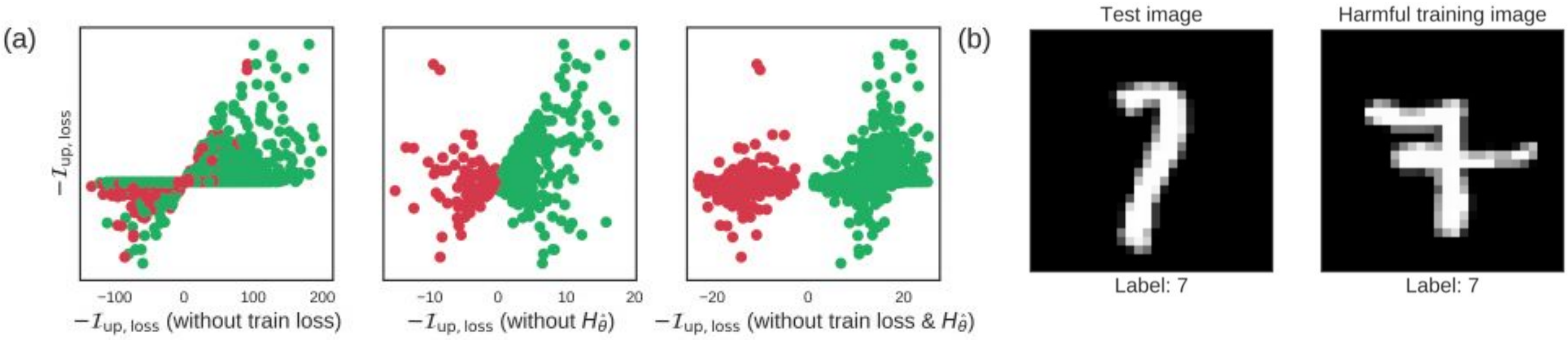
Influence on Test Loss

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

H^{-1}_θ tells us that is if ∇_θ $L(z,\theta)$ points in a direction of little curvature, its influence on the loss at z_{test} will be higher

Influence Functions vs nearest neighbors

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) \;=\; -y_{\text{test}}\, y \cdot \sigma(-y_{\text{test}}\theta^\top x_{\text{test}}) \cdot \sigma(-y\theta^\top x) \cdot x_{\text{test}}^\top H_{\hat{\theta}}^{-1} x.$$



Efficiently Computing Influence

Main Idea: Can compute $s_{\text{test}} \stackrel{\text{def}}{=} H_{\hat{\theta}}^{-1} \nabla_\theta L(z_{\text{test}}, \hat{\theta})$ without explicitly inverting H_θ

Conjugate gradient method: Compute s_{test} by framing it as a solution to a system of equations

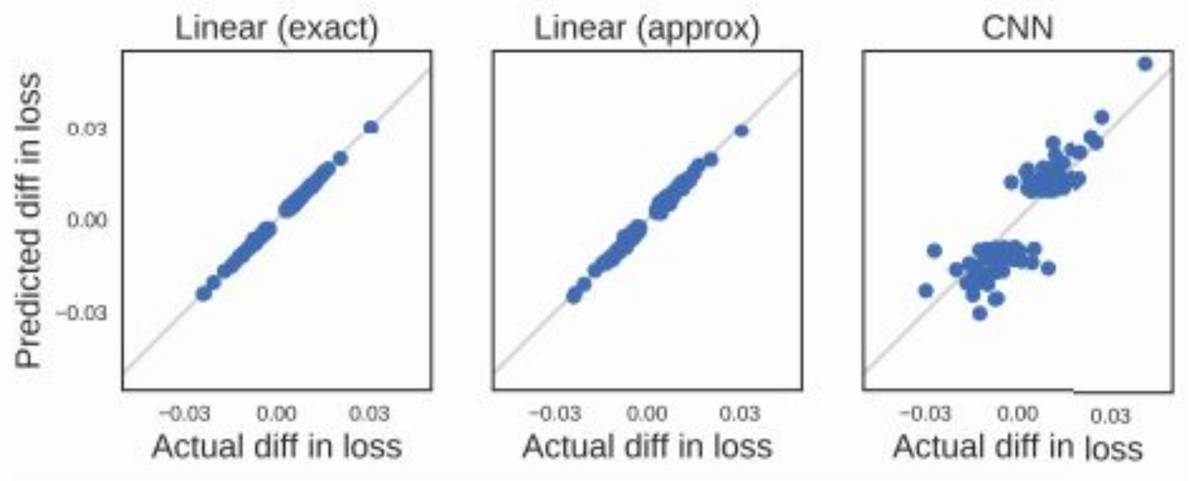
$$H_{\hat{\theta}} s_{test} = \nabla_\theta L(z_{test}, \hat{\theta})$$

Stochastic estimation [Agarwal et al. (2016)]: Sample *t* training points uniformly and recursively compute an unbiased estimate of s_{test} Empirically show this is faster than CG

$$\hat{s}_{test,j} = \nabla_\theta L(z_{test}, \hat{\theta}) + (I - \nabla_\theta^2 L(z_j, \hat{\theta}))\hat{s}_{test,j-1}$$

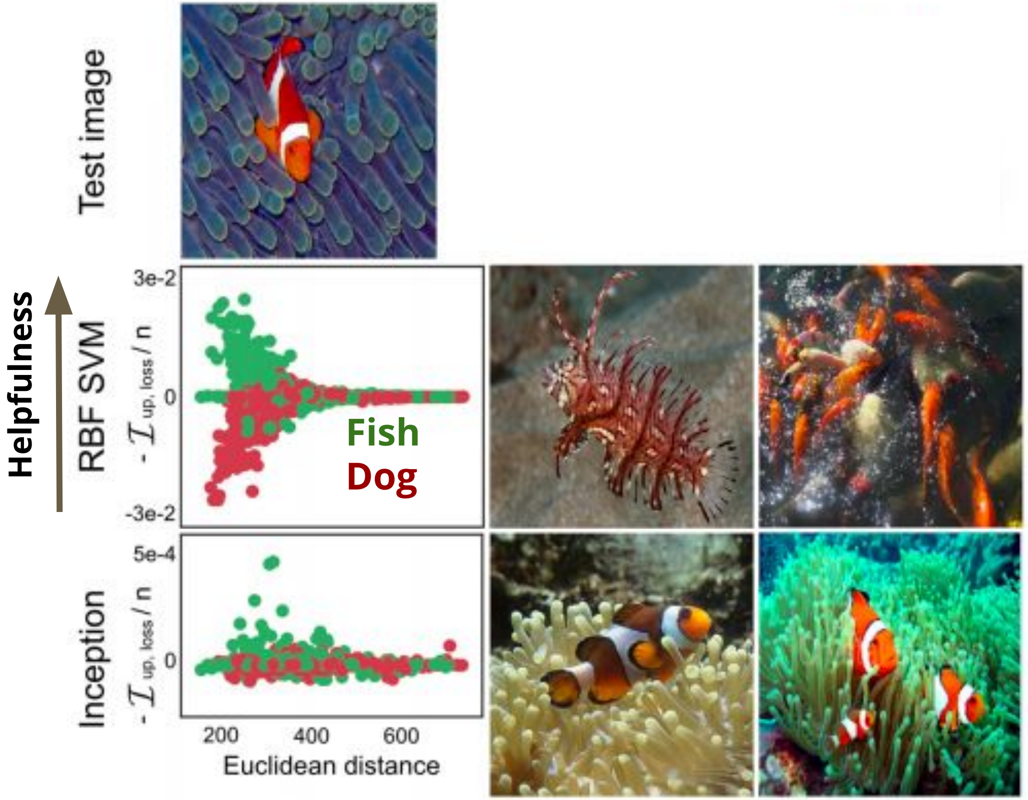
Comparison to Leave-one-out Training

Well correlated with actual difference in loss even for more complex models (such as a CNN)



Applications

Understanding Model Behavior

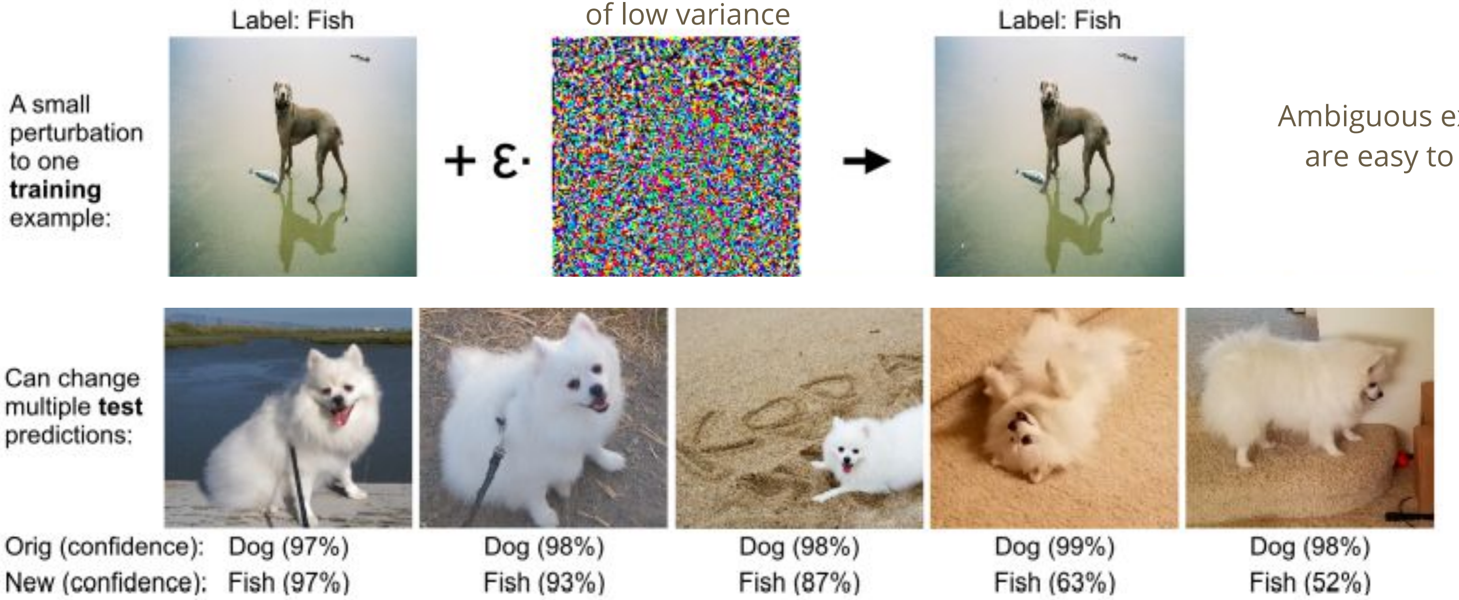


Influence functions can show which training data can help make a correct prediction on a test image

For example: the dog (pictured) was most helpful in determining that the test image was a fish

Generating examples for Adversarial Attacks

Small, and in direction of low variance



Ambiguous examples are easy to attack

Attacking one image flipped 57% of the test predictions, 2 images flipped 77%

Finding training points that have most influence

Domain Mismatch

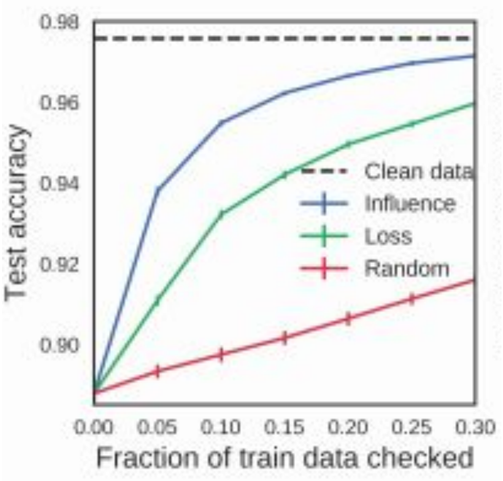
Training data distribution doesn't match test data distribution

Example: data from different hospitals serve different populations

- Use I.F. to track training data that drive incorrect classifications
- Example: identifying offenders in an Imbalanced data set (e.g. children are re-admitted with high % so classifier predicts most re-admits are children)

Mislabled Examples

Focusing the attention of experts on training data that may be "corrupt": E.g. can we find the randomly flipped labels (~10%) in training data for spam filtering



Code and Data Repository

Dockerfile (full tensorflow environment): https://hub.docker.com/r/pangwei/tf1.1/

- Numpy/Scipy/Scikit-learn/Pandas
- Tensorflow (tested on v1.1.0)
- Keras (tested on v2.0.4)
- Spacy (tested on v1.8.2)
- h5py (tested on v2.7.0)

Datasets

We use the MNIST, Dogfish, Enron spam, and UCI diabetes hospital readmission datasets. For attribution, please see the paper.

uuid[0:8]	name	summary	data_size	state	description
0xb2d85c	mnist	[uploaded]	11.1m	ready	MNIST
0x550cd3	dogfish	[uploaded]	741m	ready	Dogfish
0x19a9fd	spam	[uploaded]	7.2m	ready	Spam (enron1) data
0xfabc11	hospital	[uploaded]	18.3m	ready	Hospital readmission data

GitHub: http://bit.ly/gt-influence
Codalab: http://bit.ly/cl-influence

Discussion

<u>Main contribution:</u> Use of Influence functions to perturb model parameters and training data to realize the impact on the test data

Influence functions give us explainability but not interpretability

- Helps with model and dataset debugging after model parameters have been generated

Influence functions measure the effect of local changes: what happens when we upweight a point with small perturbations?

- Lack of global influence

Can be applied to models with non-differentiable loss functions by replacing with smooth loss functions

As developed by the authors, which of the following terms is NOT a component that effects I_up,loss, the influence of upweighting z on the loss at a test point z_test?
a) training loss
b) weighted covariance matrix
c) scaled Euclidean distance
d) nearest neighbor in Euclidean space

Calculating the influence function can be computationally challenging. Which method did the authors contribute to to improve computational efficiency for large data sets?
a) conjugate gradients
b) inversion of the Hessian of the empirical risk function
c) second-order stochastic estimation
d) leave-one-out retraining

What was NOT a model used by the authors to demonstrate the significance of influence functions?
a) Random forest
b) Logistic regression
c) Inception v3 neural network
d) SVM with RBF kernels

What was NOT a use-case offered by the authors of the application of influence functions?
a) Identifying models that heavily rely on few training examples
b) Removing training points to improve classification predictions
c) Identifying training examples responsible for misclassification errors
d) Fixing mislabeled examples

Identifying data points for Adversarial Attacks

Use influence functions to generate small perturbations on training set that change prediction outcomes on test images

$z = (x, y) \mapsto z_\delta = (x + \delta, y)$

$\mathcal{I}_{\text{pert,loss}}(z, z_{\text{test}})^\top$: shows how we can perturb a training point z by δ to have the maximal increase in the loss on z_{test}

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