Influence Functions

A Literature Study

Shreyas Havaldar and Tarun Ram Menta

Our model is making very confident predictions as “guitar” for very blatantly “cat” images. Why?

Guitar
(93%)

Guitar
(98%)

Guitar
(99%)

Overview

Influence functions are basically an analytical tool that can be used to assess the effect of removing an observation on the value of a statistic without having to re-calculate that statistic.

Whereas many methods of interpretation treat the trained model as fixed, influence functions treat it as a function of the training examples it has been fed, which gives it unique capabilities.

Their novelty lies in the fact that we can notice the effect of removing a single observation without having to re-fit the model. Because we are working on data samples it is pretty evident that influence functions are relevant to the domain of supervised learning.

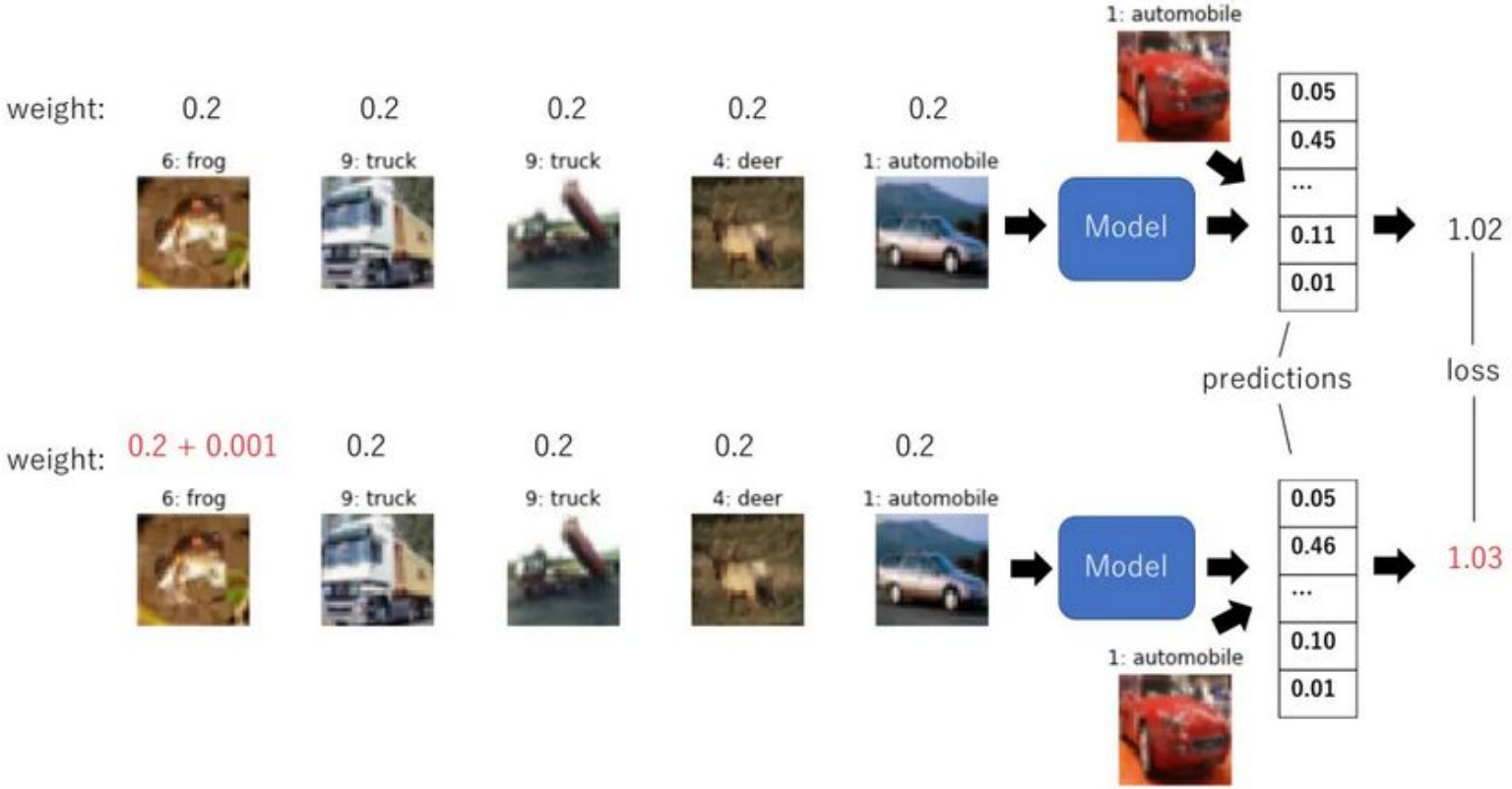
"How would the model change if we were to increase the weight/importance of some example in the training sample by a tiny bit?"

But why Influence Functions?

The empirical influence function is a measure of the dependence of the estimator on the value of one of the points in the sample. It is a model-free measure in the sense that it simply relies on calculating the estimator again with a different sample.

The most effective way of quantitatively measuring the effect of a single training sample on a model's decision would be to re-train the model, excluding this sample. Since this is not feasible for large neural networks, influence functions are used.

We essentially study the influence of a training example z on the loss for a test sample z_{test} . A training sample that changes the test loss for z_{test} by a large amount has a large influence on the model predictions with regard to z_{test} , interpreting this influence whether positive or negative can be beneficial in understanding the contribution of this training sample to our prediction.


weight:
0.2
0.2
0.2
0.2
0.2
6: frog
9: truck
9: truck
4: deer
1: automobile
1: automobile
Model
0.05
0.45
...
0.11
0.01
1.02
predictions
loss
weight:
0.2 + 0.001
0.2
0.2
0.2
0.2
6: frog
9: truck
9: truck
4: deer
1: automobile
Model
1: automobile
0.05
0.46
...
0.10
0.01
1.03

Finding samples with high negative influence is an simple method to find misleading data.

Negative influence top 4

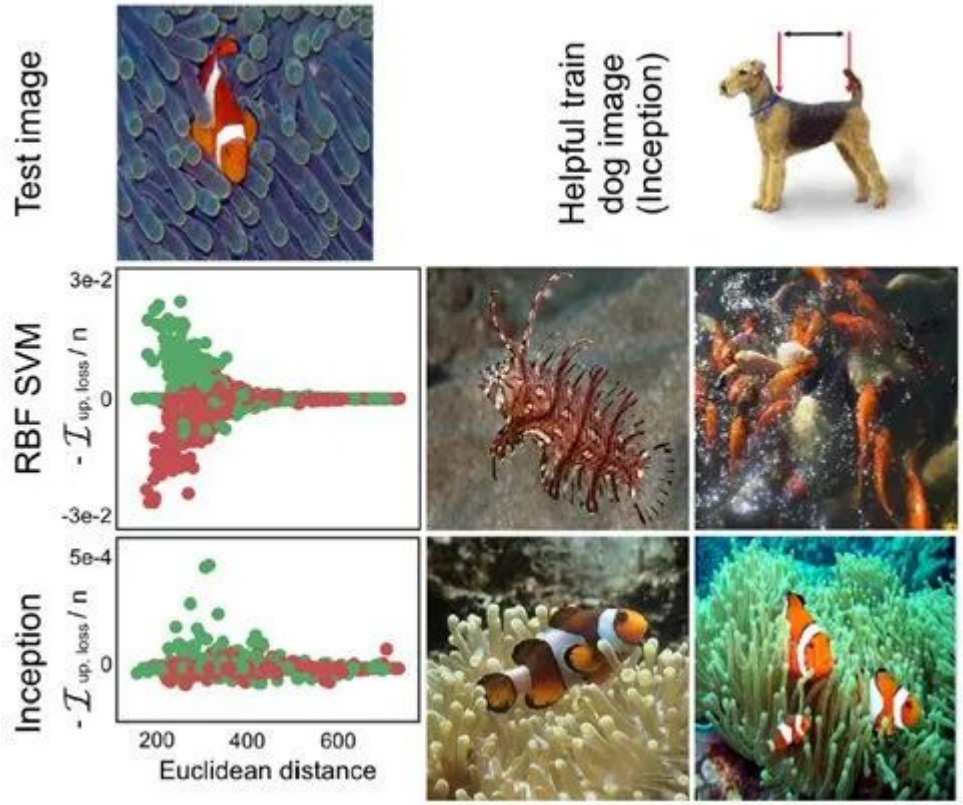
Label: Guitar

Label: Guitar

Label: Guitar

Label: Guitar

Points that are closer will have a larger influence which seems reasonable given that the model predictions for a certain test example will be most influenced by examples that are "similar" from the perspective of the model. Note that this "similarity" can be completely different from simple Euclidean distance as seen here for a Dog vs Fish classification task. Fishes are green points and Dogs are red points.



This means the influence function is a much more general method of discovering influential training examples. Versatility is the x-factor we aim to exploit when using influence functions.

And thus influence functions are a really potent tool we can apply in a wide variety of domains.

Influence functions mathematically.

The new optimal parameters when a training sample z is up-weighted by a small value ϵ is defined as

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

Using influence functions, the effect of the upweighting of the particular training sample can be approximated as

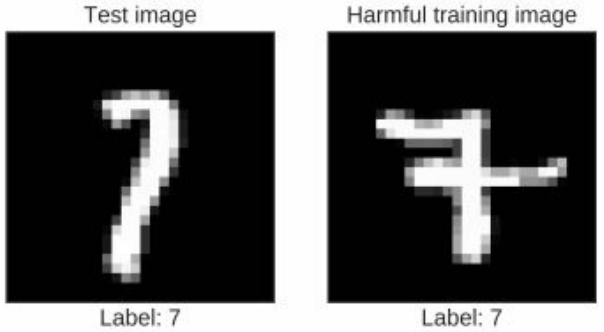
$$\mathcal{I}_{\text{up,params}}(z) \stackrel{\text{def}}{=} \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1}\,\nabla_\theta L(z,\hat{\theta}),$$

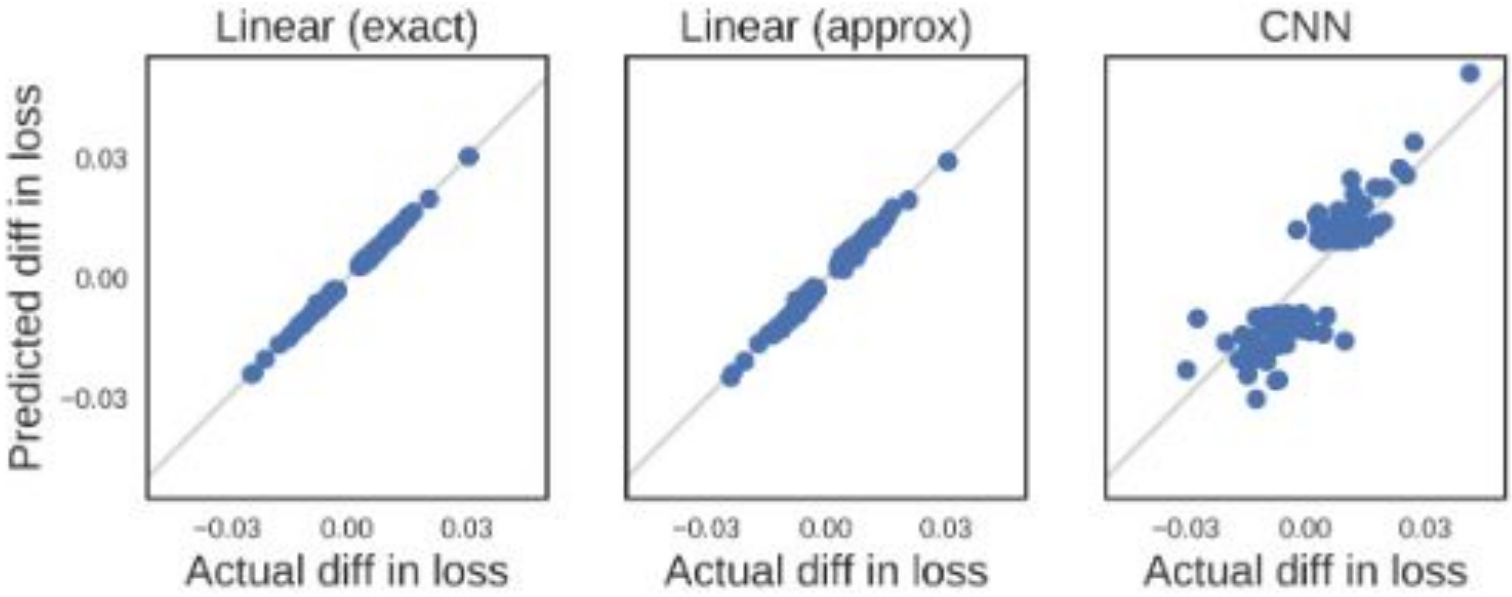
$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta^2 L(z_i,\hat{\theta})$$

Hence, The effect of removing the training sample z on the optimal parameters can be approximated as

$$\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} I_{up,params}(z)$$

$$\hat{\theta}_{-z} \stackrel{\text{def}}{=} \arg\min_{\theta\in\Theta} \sum_{z_i\neq z} L(z_i,\theta).$$




Linear (exact)
Linear (approx)
CNN
Predicted diff in loss
0.03
0.00
−0.03
−0.03
0.00
0.03
Actual diff in loss
Actual diff in loss
Actual diff in loss

Generating Adversarial Training Images

Influence functions are used to find perturbations in particular training samples which will result in the maximum increase in test loss

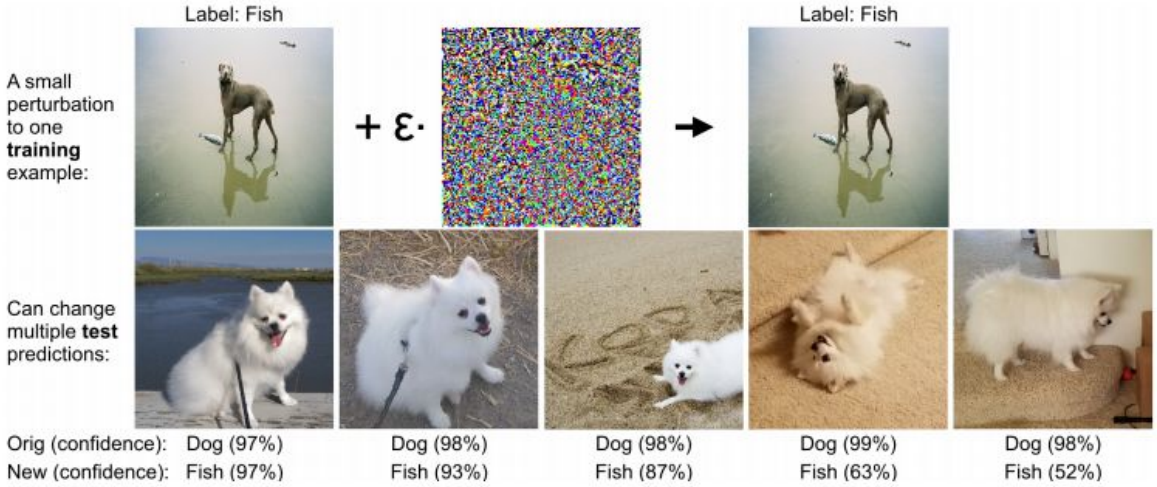


Figure 5. **Training-set attacks.** We targeted a set of 30 test images featuring the first author's dog in a variety of poses and backgrounds. By maximizing the average loss over these 30 images, we created a visually-imperceptible change to the particular training image (shown on top) that flipped predictions on 16 test images.

Fixing mis-labelled training examples

- It is impossible in many applications to manually review all of the training data

- Using influence functions to prioritize the training points to inspect for possible mis-labelling

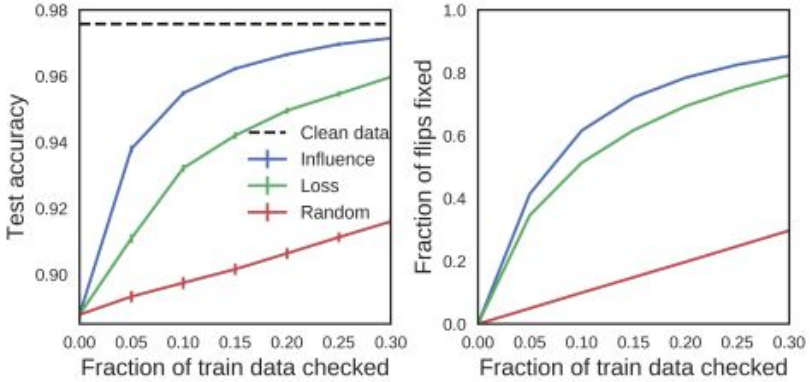


Figure 6. **Fixing mislabeled examples.** Plots of how test accuracy (left) and the fraction of flipped data detected (right) change with the fraction of train data checked, using different algorithms for picking points to check. Error bars show the std. dev. across 40 repeats of this experiment, with a different subset of labels flipped in each; error bars on the right are too small to be seen. These results are on the Enron1 spam dataset (Metsis et al., 2006), with 4,147 training and 1,035 test examples; we trained logistic regression on a bag-of-words representation of the emails.

Different Approach to Calculating influence for GLMs

Wojnowicz et al.

$$c_i \propto (\widehat{\boldsymbol{\beta}}_{(i)} - \widehat{\boldsymbol{\beta}})^T \, Cov(\widehat{\boldsymbol{\beta}})^{-1} \, (\widehat{\boldsymbol{\beta}}_{(i)} - \widehat{\boldsymbol{\beta}}) \qquad (7)$$

Algorithm 1 (Influence Sketching): Calculating sample influence for large scale regressions

Data A dataset $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, a random projection matrix $\boldsymbol{\Omega} \in \mathbb{R}^{p \times k}$, and a regression model chosen from the family of generalized linear models.
Result A vector $\boldsymbol{c} \in \mathbb{R}^n$ quantifying the influence of each sample on the model fit.

1: From the regression model, obtain the fitted observations $\widehat{\mathbb{E}}[\boldsymbol{y}|\boldsymbol{X}]$ and the converged IRLS (iteratively reweighted least squares) diagonal weight matrix, $\boldsymbol{V} \in \mathbb{R}^{n \times n}$, where the latter can be constructed using a lookup table or the relationship in Equation 21.
2: Compute the re-weighted residuals $\widehat{\boldsymbol{r}^*} = \boldsymbol{V}^{-1/2}(\boldsymbol{y} - \widehat{\mathbb{E}}[\boldsymbol{y}|\boldsymbol{X}])$
3: Form re-weighted randomly projected data: $\boldsymbol{Z} \in \mathbb{R}^{n \times k}$: $\boldsymbol{Z} = \boldsymbol{V}^{1/2}\boldsymbol{Y}$, where $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\Omega}$ is the original dataset randomly projected to k dimensions.
4: Form the inverse-covariance matrix of the re-weighted randomly projected data : $\boldsymbol{W} \in \mathbb{R}^{k \times k} : \boldsymbol{W} = (\boldsymbol{Z}^T\boldsymbol{Z})^{-1}$
5: Get the leverage values as the diagonal elements of the generalized hat matrix, $h_i^* \in \mathbb{R} : h_i^* = \boldsymbol{z}_i^T \boldsymbol{W} \boldsymbol{z}_i$
6: Compute the approximate influence (or generalized Cook's distance) scores as $c_i = (\widehat{r^*}_i)^2 \dfrac{h_i^*}{(1 - h_i^*)}$

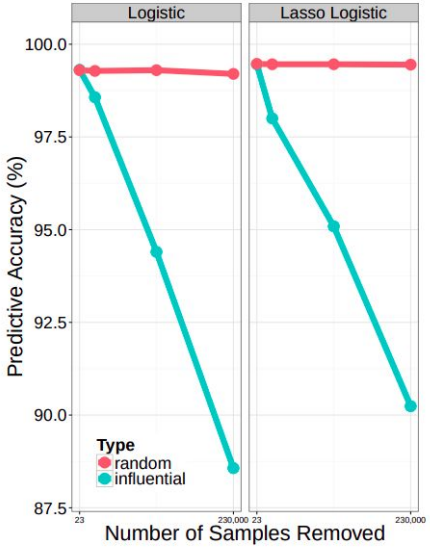


Fig. 5: *Influential samples, as determined by influence sketching, disproportionately drive model performance.* The plot shows the predictive performance on a hold-out test set after removing up to ≈10% of the training set of ≈2.3 million samples, where removed samples are either influential or random. The model on the left is a standard (unregularized) logistic regression, whereas the model on the right is a L1-regularized ("lasso") logistic regression.

Better Data Augmentation via Influence Functi
Lee et al.

- Use influence functions to learn an optimal image-wise transformation
- Attempt to find the per-image transformation that maximizes the improvement in validation loss

$$-\frac{1}{N}\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}}) \simeq \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}})) - \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \tilde{z}_i \backslash z_i)),$$

$$\mathcal{I}_{\text{aug, loss}}(z_i, \tilde{z}_i, \mathbf{z}^{\text{val}}) \triangleq \frac{d\mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon\tilde{z}_i \backslash \epsilon z_i))}{d\epsilon}\Big|_{\epsilon=0} \tag{16}$$

$$= \nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top \frac{d\hat{\theta}(\mathbf{z}^{\text{tr}} \cup \epsilon\tilde{z}_i \backslash \epsilon z_i)}{d\epsilon}\Big|_{\epsilon=0} \tag{17}$$

$$= -\nabla_\theta \mathcal{L}(\mathbf{z}^{\text{val}}, \hat{\theta}(\mathbf{z}^{\text{tr}}))^\top H(\hat{\theta}(\mathbf{z}^{\text{tr}}))^{-1} \left(\nabla_\theta l(\tilde{z}_i, \hat{\theta}(\mathbf{z}^{\text{tr}})) - \nabla_\theta l(z_i, \hat{\theta}(\mathbf{z}^{\text{tr}}))\right) \tag{18}$$

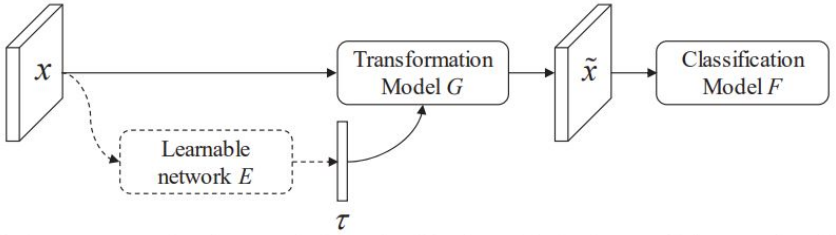


Figure 1: A generic data augmentation framework for a classification task. An input training sample x is transformed to $\tilde{x}$ by a transformation model, which is parameterized by τ. The conventional method usually defines G as a composition of predefined transformations based on randomly sampled range τ (solid line path). In this paper, G is a differentiable network. A learnable network E estimates τ given x to obtain the transformed sample $\tilde{x}$, where $\tilde{x}$ maximizes the generalization performance of the classification model (dashed line path).

Model	Baseline [6]	Baseline (ours)	AutoAug. [6]	Proposed
ResNet-50 [15]	76.3 / 93.1	76.1 / 93.0	77.6 / 93.8	77.1 / 93.4
ResNet-200 [15]	78.5 / 94.2	78.1 / 94.0	80.0 / 95.0	79.0 / 94.6

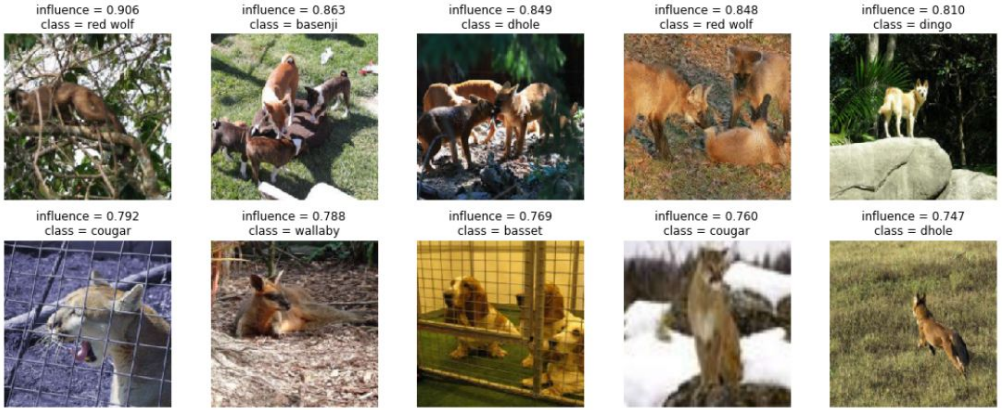
Table 3: Validation set Top-1 / Top-5 accuracy (%) on ImageNet dataset. The experiments are conducted under the same setting as [6]. All results are obtained using 1-crop testing.

Dataset	AutoAug. [6]	Proposed
CIFAR-10	5,000	8
ImageNet	15,000	40

Table 4: GPU hours comparison of AutoAugment and the proposed method. Ours are estimated with Titan-X Pascal.

Criticisms

- The influence functions currently in use are found to be erroneous for deeper networks
- The stochastic estimation methods for the Hessian are also erroneous as the network depth increases.
- In certain architectures, weight decay regularization is very important in obtaining high quality influence estimates



Unrelated images are found to have high influence on the test class(Kit Fox).

Advantages of Influence Functions

- Interpretability, understanding the particular weaknesses or strengths of a model.
- The approaches of deletion diagnostics and influence functions are very different from the mostly feature-perturbation based approaches. A look at influential instances emphasizes the role of training data in the learning process. Efficient approach to fix and modify training data to remove unwanted noise.
- Data Augmentation at a very inexpensive computing costs as compared to SOTA methods. The influence function is used to predict the effects of a particular augmented training sample on generalization performance. The differentiable augmentation network is learned to generate augmented samples that maximize the influence function, thereby minimizing the validation loss.
- Finding possible domain mismatch situations for the training and test sets. We can easily see if the test samples are being predicted wrong due to specific test samples in our input thus helping us find if the training distribution does not match the test distribution.

Drawbacks and Solutions

- Influence functions require the computation of a hessian gradient product but we do possess multiple tricks for computing efficiently. So computation costs can be significantly shrunk.
- Oracle access to gradients and Hessian-vector products.
- Assume that the loss function is convex wrt the model parameters. But via approximations can work surprisingly well for non-convex models as well. Can be applied to deep neural networks via a smoothing constant to make the hessian positive-definite.
- No clear cutoff of the influence measure at which we call an instance influential or non-influential.
- Assumption that the loss is twice differentiable with regards to the model parameters. We simply approximate the loss function with a twice differentiable loss and move ahead!

$$\max(0, 1-x) \longrightarrow t\log(1+\exp(\tfrac{1-x}{t}))$$

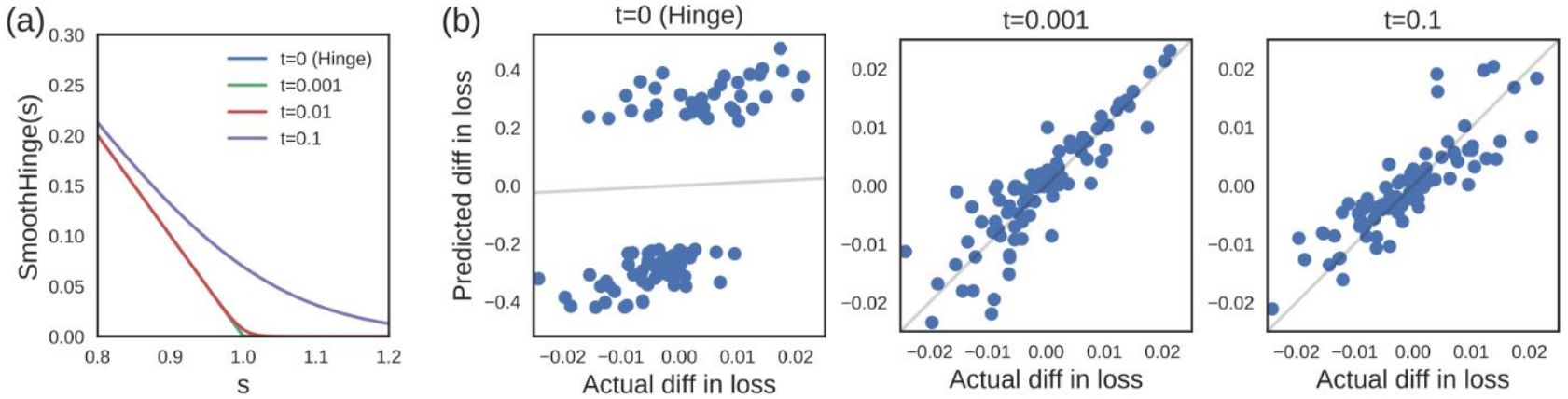


Figure 3. **Smooth approximations to the hinge loss. (a)** By varying t, we can approximate the hinge loss with arbitrary accuracy: the green and blue lines are overlaid on top of each other. **(b)** Using a random, wrongly-classified test point, we compared the predicted vs. actual differences in loss after leave-one-out retraining on the 100 most influential training points. A similar trend held for other test points. The SVM objective is to minimize $0.005 \|w\|_2^2 + \frac{1}{n} \sum_i \text{Hinge}(y_i w^\top x_i)$. **Left:** Influence functions were unable to accurately predict the change, overestimating its magnitude considerably. **Mid:** Using $\text{SmoothHinge}(\cdot, 0.001)$ let us accurately predict the change in the hinge loss after retraining. **Right:** Correlation remained high over a wide range of t, though it degrades when t is too large. When

Possible Directions

- Different Influence Functions
  - Capture the idea of similarity better
  - Functions which are simpler to compute
  - More universal and robust influence functions

- Adversarial Robustness
  - Finding and generating misleading examples

- Data Augmentation
  - Using influence functions as a means to generate most influential data sample to achieve best generalization performance

References

- Understanding Black-box Predictions via Influence Functions
- Influence Functions in Deep Learning Are Fragile
- Learning Augmentation Network via Influence Functions
- "Influence Sketching": Finding Influential Samples In Large-Scale Regressions
- Influence of single observations on the choice of the penalty parameter in ridge regression

Thank You!