Estimating Training Data Influence by Tracing Gradient Descent

Md Mahfuzur Rahman
Postdoctoral Research Associate
TReNDS Center
Georgia State University

TReNDS
Center for Translational Research in Neuroimaging & Data Science

Estimating Training Data Influence by Tracing Gradient Descent

Garima* Google pruthi@google.com

Frederick Liu* Google frederickliu@google.com

Satyen Kale Google satyenkale@google.com

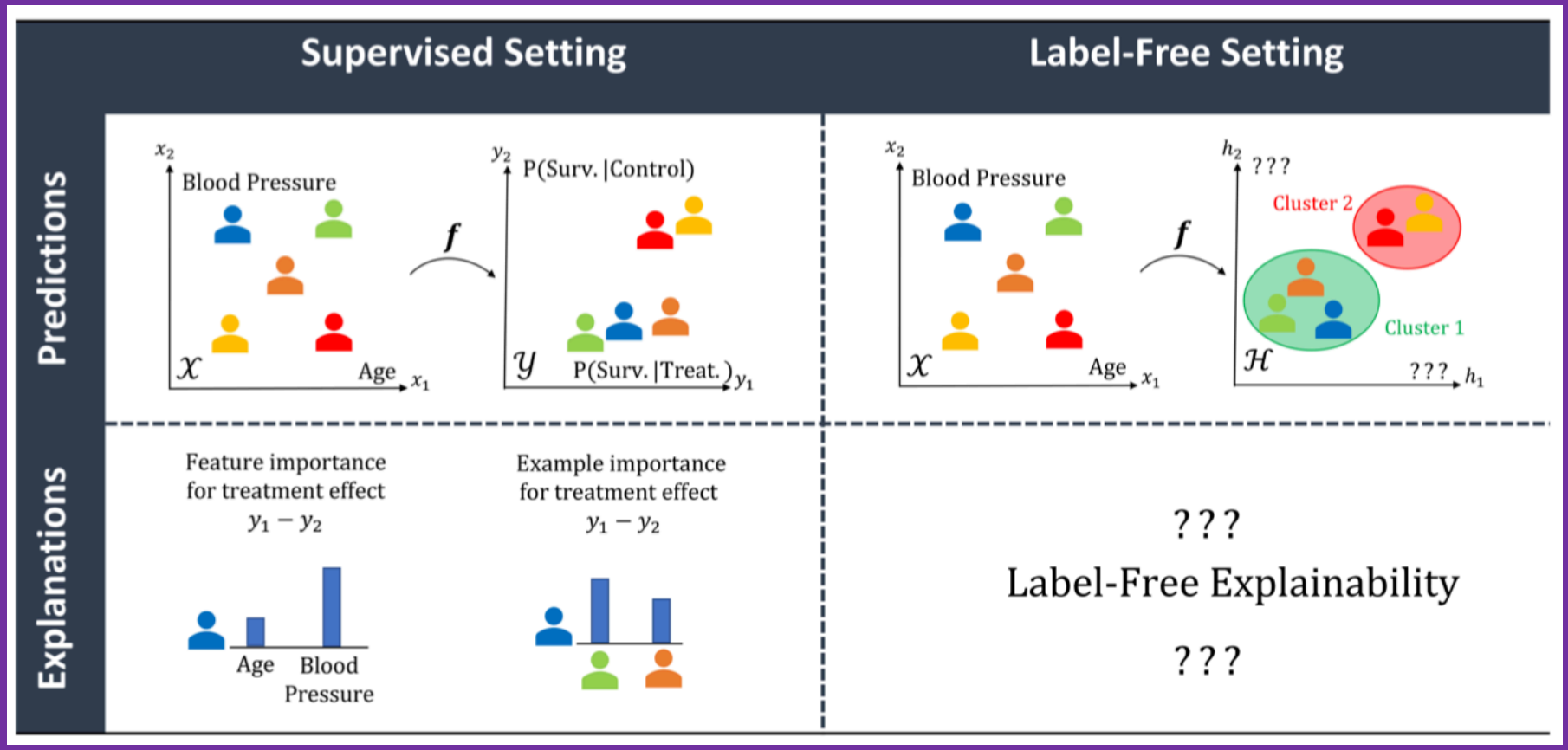
Mukund Sundararajan † Google mukunds@google.com

Venue: NeurIPS 2020

paper link

Explanation at different settings

MLBBQ



Crabbé, Jonathan, and Mihaela van der Schaar. "Label-Free Explainability for Unsupervised Models." International Conference on Machine Learning. PMLR, 2022

Sutre, Elina Thibeau, et al. "How can data augmentation improve attribution maps for disease subtype explainability." Medical Imaging 2023: Image Processing. SPIE, 2023.

Post hoc explanations – a brief introduction

Input: $\boldsymbol{x} \in \mathbb{R}^d$

Model: $F : \mathbb{R}^d \to \mathbb{R}^C$

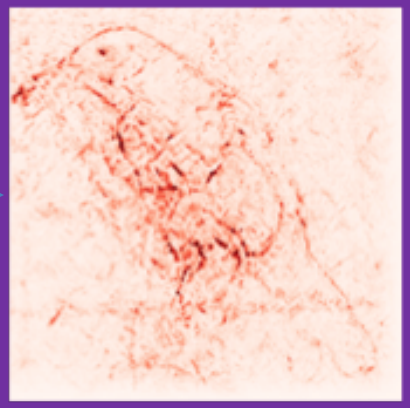
Class-specific logit: $F_c(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$

Explanation: $E : \mathbb{R}^d \to \mathbb{R}^d$

Input

Model (usually Black-box)

Explanation E



Explanation

Introduction to *TracIn*

MLBBQ

- ***TracIn*** - computes the influence of a training example on a prediction.


church


church


church


church


castle


castle


castle

test sample

proponents

opponents

The main idea:

- how does the loss on the test point change during the training process?

For scalable implementation:

a. Use first-order gradient approximation.
b. Saved checkpoints of standard training procedures
c. Cherry-picking layers of a deep neural network.

TracIn follows Fundamental theorem of calculus

MLBBQ

- FTC decomposes the difference between a function at two points
- Uses the gradients along the path.
- The integration of the gradients between two points is equal to the difference.
- Approximation to integration is available.
- TracIn decomposes the difference between the losses of the test point

Review of fundamental theorem of calculus

MLBBQ

$$A(x+h) - A(x) = f(x).h + (Excess)$$

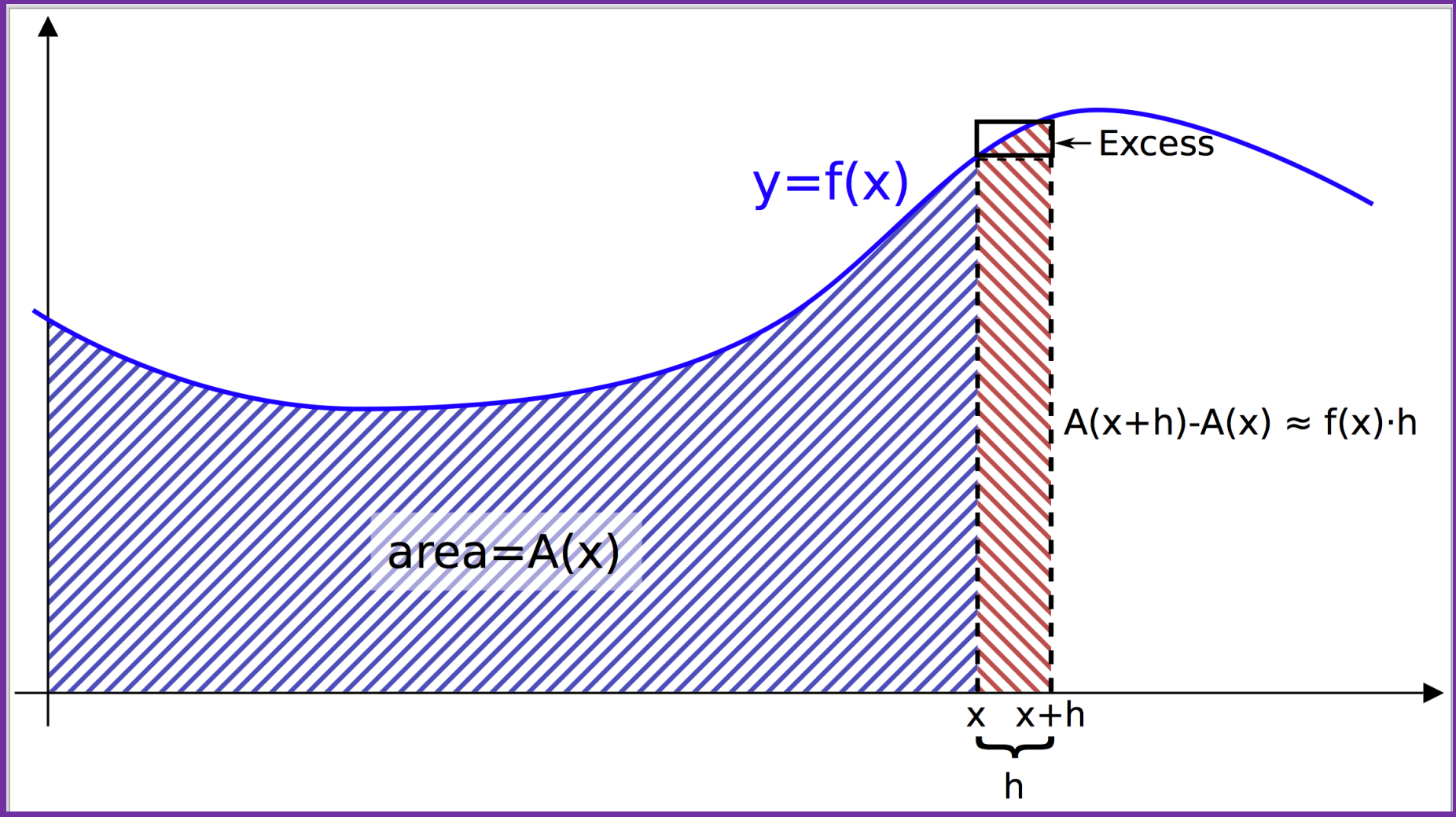


Image: wikipedia

Assume: A represents loss function, f represents gradients

Idealized Notion of Influence

Training set: $\mathbb{S} = \{z_1, z_2, \ldots, z_n \in \mathbb{Z}\}$

$$\texttt{TracInIdeal}(z, z') = \sum_{t:\, z_t = z} \ell(w_t, z') - \ell(w_{t+1}, z')$$

- Assume SGD as an optimizer
- Consider the steps where training example z was used

Lemma 3.1 *Suppose the initial parameter vector before starting the training process is w_0, and the final parameter vector is w_T. Then* $\sum_{i=1}^{n} \texttt{TracInIdeal}(z_i, z') = \ell(w_0, z') - \ell(w_T, z')$

First-order gradient approximation (FOGA)

FOGA to loss function:

$$\ell(\boldsymbol{w}_{t+1}, \boldsymbol{z}') = \ell(\boldsymbol{w}_t, \boldsymbol{z}') + \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z}') \cdot (\boldsymbol{w}_{t+1} - \boldsymbol{w}_t) + O(\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2)$$

According to gradient update rule:

$$\boldsymbol{w}_{t+1} - \boldsymbol{w}_t = -\eta_t \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z}_t)$$

Loss change now can be approximated as:

$$\ell(\boldsymbol{w}_t, \boldsymbol{z}') - \ell(\boldsymbol{w}_{t+1}, \boldsymbol{z}') \approx \eta_t \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z}') \cdot \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z}_t)$$

The influence of a training example:

$$\texttt{TracIn}(\boldsymbol{z}, \boldsymbol{z}') = \sum_{t:\, \boldsymbol{z}_t = \boldsymbol{z}} \eta_t \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z}') \cdot \nabla\ell(\boldsymbol{w}_t, \boldsymbol{z})$$

Issue: not feasible, gradient is not available per training example

Minibatch formulation

Summing up over all iterations t in which a particular training point z was chosen in B_t

$$\texttt{TracIn}(\boldsymbol{z}, \boldsymbol{z}') = \frac{1}{b} \sum_{t:\, \boldsymbol{z} \in B_t} \eta_t \, \nabla \ell(\boldsymbol{w}_t, \boldsymbol{z}') \cdot \nabla \ell(\boldsymbol{w}_t, \boldsymbol{z})$$

Practical Heuristic Influence via Checkpoints

- Access to the parameter vector at the specific iteration is not available.
- approximate it with the first checkpoint parameter vector after it.

K checkpoints:

$$\boldsymbol{w}_{t_1}, \boldsymbol{w}_{t_2}, \ldots, \boldsymbol{w}_{t_k}$$

Influence based on checkpoints:

$$\texttt{TracInCP}(\boldsymbol{z}, \boldsymbol{z}') = \sum_{i=1}^{k} \eta_t \, \nabla \ell(\boldsymbol{w}_{t_i}, \boldsymbol{z}) \cdot \nabla \ell(\boldsymbol{w}_{t_i}, \boldsymbol{z}')$$

TracIn for Different Optimizers

MLBBQ

Derivation to another experimental setup could be different. But CP version would be quite same

Remark 3.4 (Handling Variations of Training) *In our derivation of* `TracIn` *we have assumed a certain form of training. In practice, there are likely to be differences in optimizers, learning rate schedules, the handling of minibatches etc. It should be possible to redo the derivation of* `TracIn` *to handle these differences. Also, we expect the practical form of* `TracInCP` *to remain the same across these variations.*

The same formulation could be used to study the influence of a hypothetical training point

Remark 3.5 (Counterfactual Interpretation) *An alternative interpretation of Equation 1 is that it approximates the influence of a training example on a test example* had it been visited *at each of the input checkpoints. Under this counterfactual interpretation, it is valid to study the influence of a point that is not part of the training data set, keeping in mind that such an example did not impact the training process or the checkpoints that arose as a consequence of the training process.*

Evaluation Approach

Self-influence score:

the influence of a training point on its own loss, i.e., the training point z and the test point z' in TracIn are identical

- Proponents (positive value of influence) serve to reduce loss
- Opponents serve to increase loss.
- Incorrectly labelled examples are likely to be strong proponents for themselves.
- When sorted based on *self-influence,* an effective influence computation method would tend to rank mislabelled examples in the beginning.

Experiments

Dataset and models:

ResNet-56 trained on the CIFAR-10.
The model on the original dataset has 93.4% test accuracy

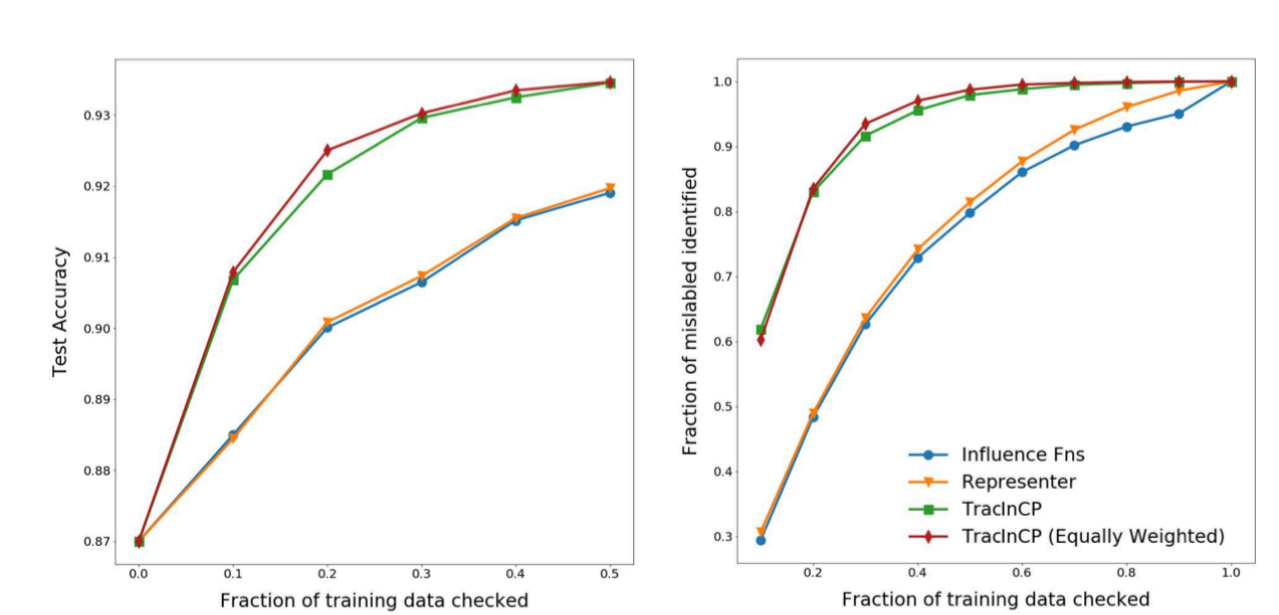
Setup for Mislabelled Example Identification:

- first train a model on correct data.
- Then, for 10% of the training data, labels were changed to the highest scoring incorrect labels.
- Test accuracy reduces from 93.4% to 87.0% (train accuracy is 99.6%).)
- Identify mislabelled examples

Metric:
the fraction of mislabelled data recovered for different prefixes of the rank order

Identifying Mislabelled Examples – CIFAR 10

- TracInCP with only the last layer.
- sample every 30 checkpoints starting from the 30th checkpoint
- Consider both “weighted” and “equally weighted” checkpoints.



(a) CIFAR-10

Right: TracIn recovers more than 80% of the mislabelled data in 20% of the ranking, the other methods recover less than 50% at the same point

Left: fixing the mislabelled data found within a certain fraction of the training data, results in a larger improvement of test accuracy

Effect of Checkpoints on TracIn Scores

MLBBQ

Different checkpoints had focus on different classes

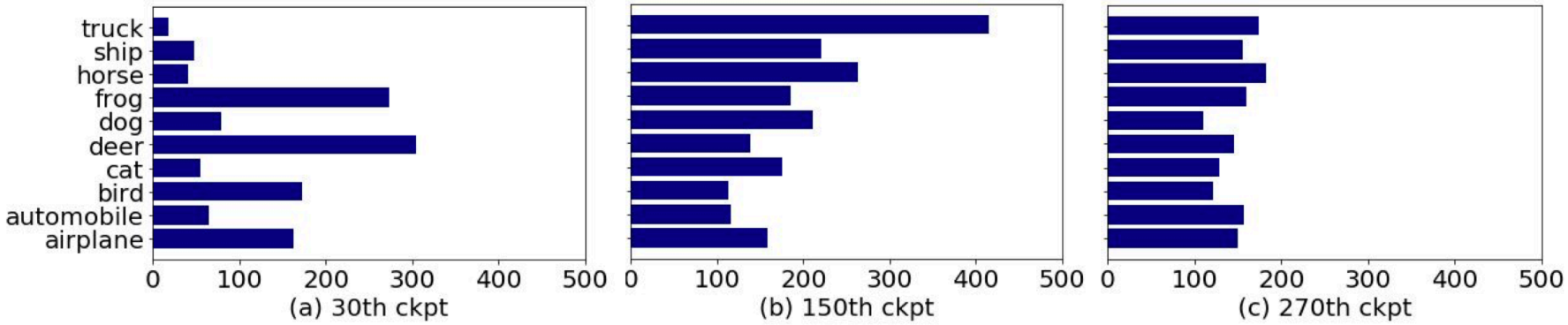
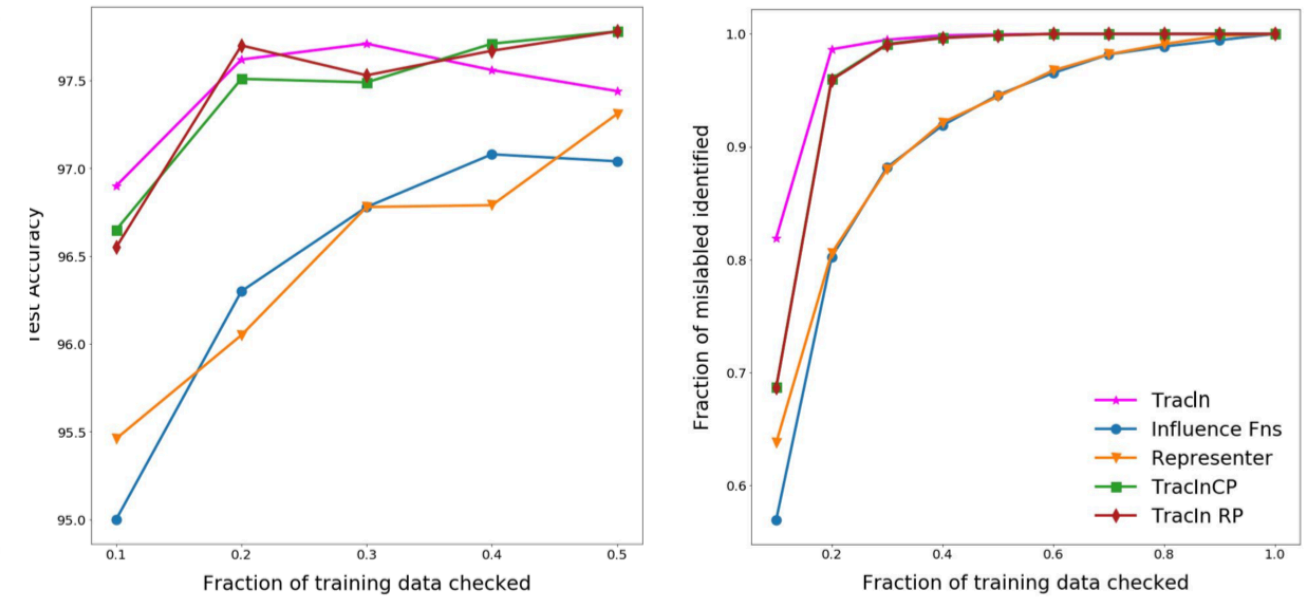


Figure 8: Number of identified mislabelled examples by class for three checkpoints within the top 10% of ranking by self-influence. Different checkpoints highlight different labels—see Section 4.2.

MNIST Experiments

- Model is small - 3 hidden layers and 240K parameters.
- 99.30% train, 97.55% test accuracy.
- 89.94% mislabelled train, 89.95% on test set
- Use each training step, and not just checkpoints.
- The whole model used, not just the last layer.

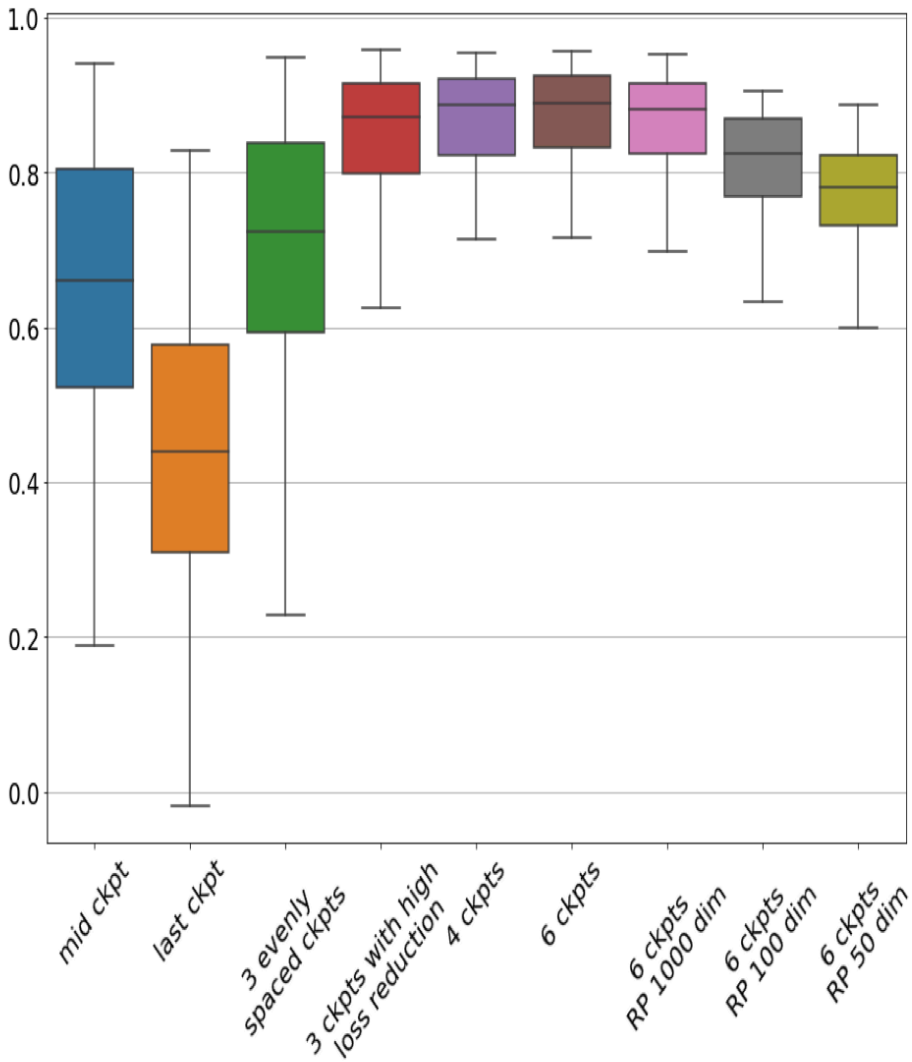


(b) MNIST

Note: approximate TracIn is able to recover mislabelled examples faster than TracInCP

Relationship between TracInCP vs. TracIn

MLBBQ

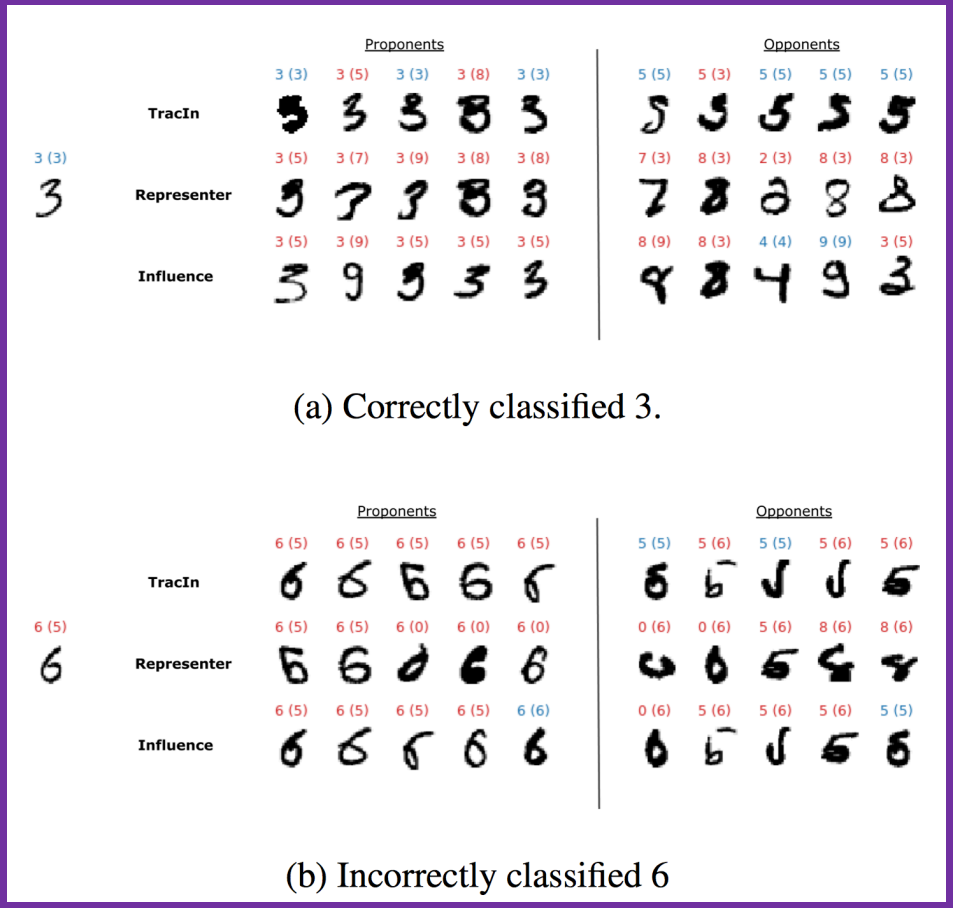


- correlation of the influence scores
- 100 test points
- TracInCP (different CPs) vs. approximation TracIn.
- Mid CP is more informative than last CP
- CPs with high loss reduction, are more informational
- Random Projection (RP) version saves computations.

RP formulation:

$$(\eta_t G \nabla \ell(\boldsymbol{w}_t, \boldsymbol{z}_t)) \cdot (G \nabla \ell(\boldsymbol{w}_t, \boldsymbol{z}'))$$

MNIST Results



(Predicted class in brackets)

- Proponents are consistently similar for TracInCP and Representer methods.
- Not necessarily examples from the same class are proponents always.
- TracInCP seems to pick pixel-wise similar opponents

More examples from CIFAR 10

MLBBQ

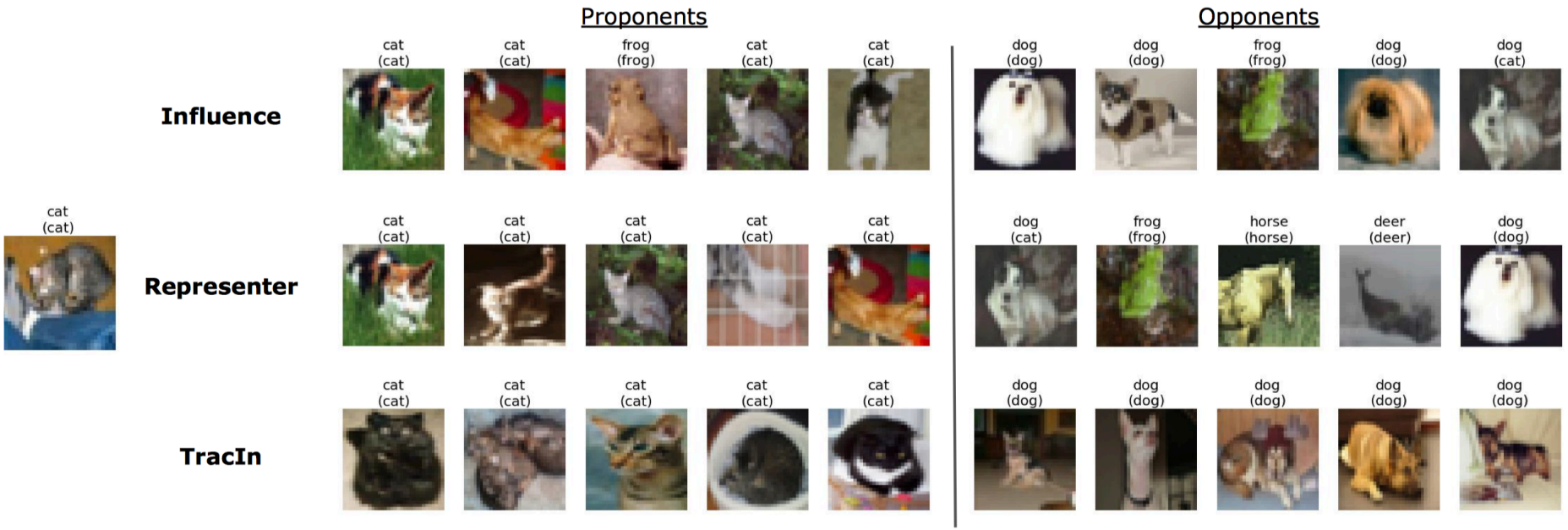


Figure 5: CIFAR-10 results: Proponents and opponents examples of a correctly classified cat for influence functions, representer point, and `TracIn`. (Predicted class in brackets)

More examples from CIFAR 10 (contd.)

MLBBQ

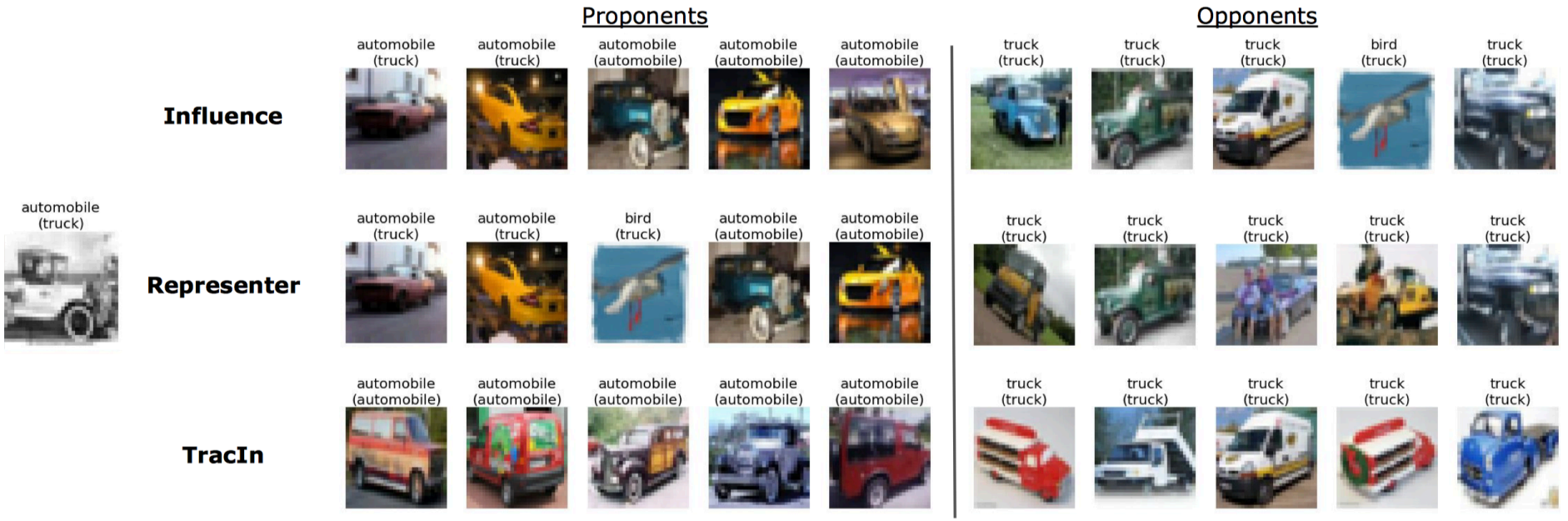


Figure 6: CIFAR-10 results: Proponents and opponents examples of an incorrectly classified automobile for influence functions, representer point, and `TracIn`. (Predicted class in brackets)

TracIn is scalable to larger dataset/model

TracIn was applied on the fully connected layer of ResNet-50 trained on ImageNet

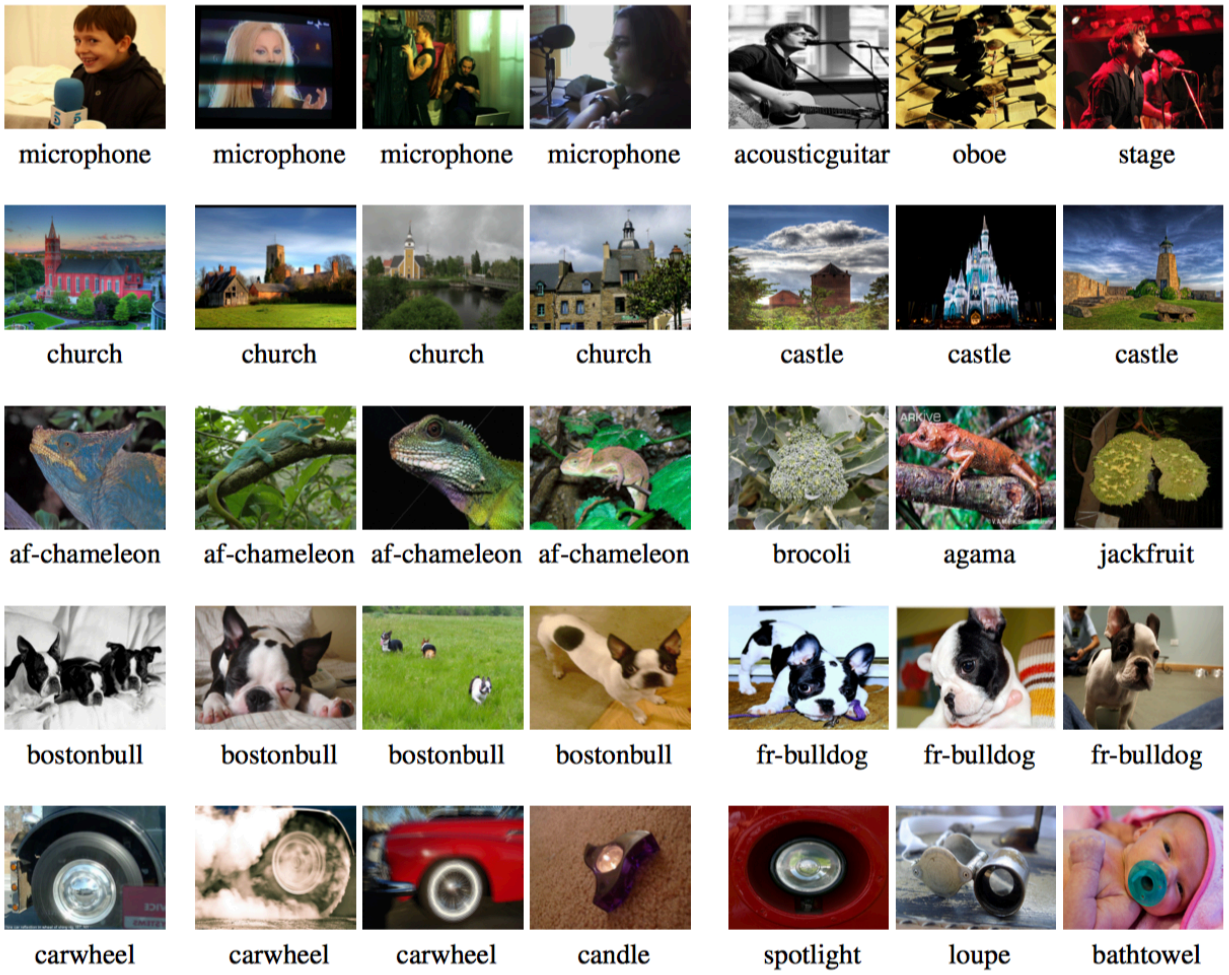


Figure 4: `TracIn` applied on Imagenet. Each row starts with the test example followed by three proponents and three opponents. The test image in the first row is classfied as band-aid and is the only misclassified example. (af-chameleon: african-chameleon, fr-bulldog: french-bulldog)

Conclusion

- The method is simple
  - only requires concepts like gradients, checkpoints and loss functions.
- The method is general
  - applicable to any ML model using SGD or its variants,
  - agnostic of architecture, domain and task.
- The method is versatile
  - explain a single prediction
  - identify misslabelled examples, etc.

Questions?

Contact: mrahman21@gsu.edu

Georgia State University

TReNDS

Center for Translational Research in Neuroimaging & Data Science

Summary

Influence functions:

what happens to the loss of a test point when you drop an individual training point and retrain?

TracIn:

how much change in loss of a test point happened between the start of training and the end of training, for a training example?

Koh, P. W., & Liang, P. (2017). Understanding black-box predictions via influence functions. In *ICML*. PMLR.