TrackIn FAQ

April 2020

Contact: mukunds@cs.stanford.edu
Authors: Garima, Frederick Liu, Satyen Kale, Mukund Sundararajan (Google)

Abstract: *This doc discusses how to apply TrackIn, a technique to identify the influence of training data points on a specific prediction. It is a simple, versatile technique that can be used in various ways to analyze and improve the quality of training data. The technique is still in its infancy and we expect the best practices to evolve and welcome feedback.*

What is TrackIn?

TrackIn identifies the influence of a training data point on a prediction point, i.e., it attributes the score of a prediction point back to the training data points. This is in contrast with attribution techniques like Shapley value or Integrated Gradients that attribute the score back to the features.

A **prediction point/example** is just an example on which you run prediction. It can be a point from the test set, the dev set, or the training set; just about any point on which you can run prediction. The prediction point does not need to be known at training time.

A **training data point/example** is just a point from the training data set. Crucially, we need to be able to compute the loss-gradient for this point, so we will need to know its true label.

There are two types of **scores** of the prediction point that we can explain. The first is the **prediction score**, which could be the softmax or the logit. The second is the loss of the prediction point. For loss, you need to know the true label of the test point, therefore it is not possible to explain loss for unlabelled examples. In this document, we will use loss as a running example.

The **input** to TrackIn is a prediction point, a training point, and a model. The **output** of TrackIn is an attribution/influence score that is proportional to the training point's effect on the prediction or the loss of the prediction point.

More specifically, the idealized version of TrackIn (discussed in Section 3.1 of the paper) produces scores for training points, that when aggregated across the training points, sum up to difference between the loss on prediction point at the *beginning* of training (high) minus the loss on the prediction point at the *end* of training (hopefully lower). The practical implementation of

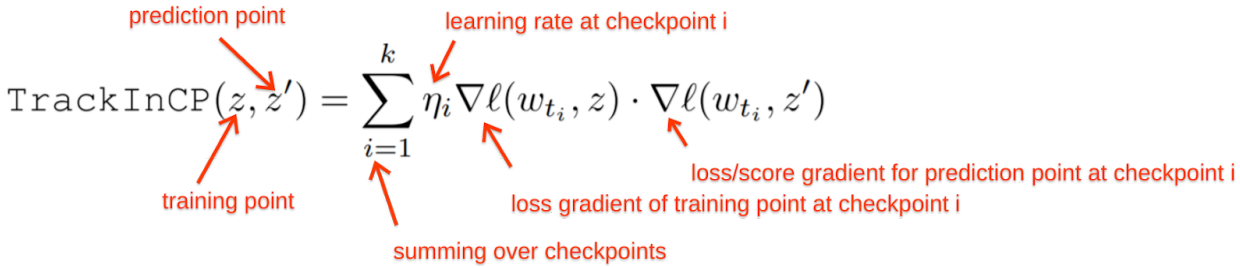
TrackIn (discussed below) is an approximation and does not satisfy this property; however, we expect the approximation to still reflect the relative importance of a training point.

We will establish two terms that make it easy to discuss the types of the influence. **Proponents** have *positive* influence/attribution scores proportional to how much they *reduce* loss or *increase* the prediction score. **Opponents** have *negative* influence scores proportional to how much they *increase* loss or *decrease* the prediction score. Proponents support the prediction. Opponents oppose the prediction.

How does TrackIn work?

TrackIn is simple to understand if you know how Stochastic Gradient Descent works. Every time a training point is visited during training, it changes the model's parameters. This change in the model's parameters in turn changes the prediction score or the loss of the prediction point. The basic idea is to attribute this change in the score to the training point.

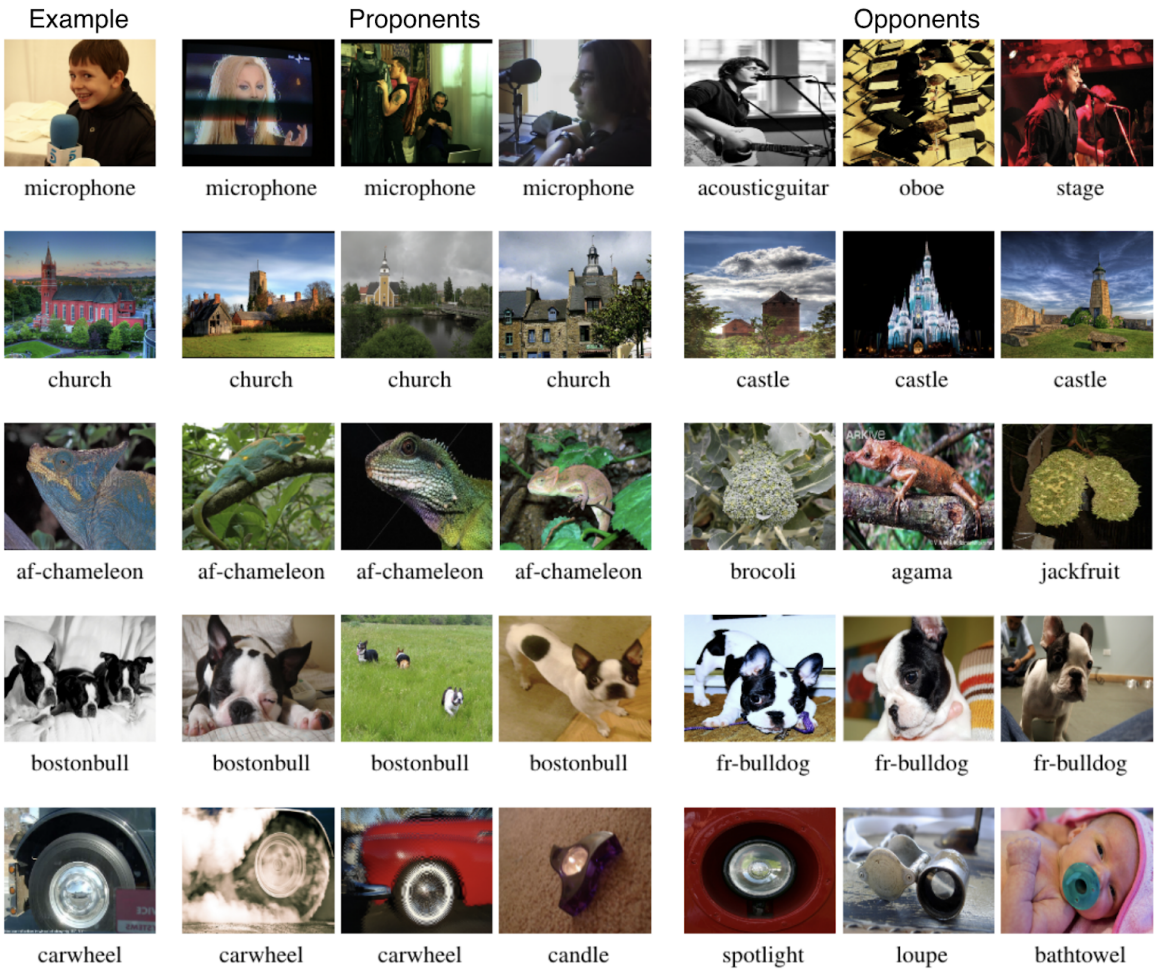
To track these changes exactly, we would have to know the prediction point during training. This is impractical. Therefore we use checkpoints and gradients to estimate/approximate this influence. Roughly, we use the gradient of the loss of the training point to estimate how it would change the model's parameters, and the gradient of the loss/prediction score of the test point to estimate how the change in the parameters would affect the score. These gradients are chained together to compute the score for the checkpoint, and this is accumulated across checkpoints:



What does TrackIn look like in action?

Image Example

We use the 30, 60, 90th checkpoint of a ResNet-50 model trained on imagenet trained for 90 epochs over the Imagenet data set. We show the top 3 proponents and top 3 opponents for 5 test examples.



Notice the visual similarity of the opponents and the proponents to the test image (the first column). However as we would expect for opponents, they have a different label from that of the test example.

Text Example

We study a task that predicts the occupation of an individual from a blurb of text. This is a task from the DBPedia Ontology dataset. There are fourteen different occupations. We use 10, 20, 30, 40, 50, 60th checkpoint of a Simple Word-Embedding model trained for 60 epochs. We show the top opponents and proponents for one example that is classified as a ‘OfficeHolder’

Example	OfficeHolder	**Manuel Azaña** Manuel Azaña Díaz (Alcalá de Henares January 10 1880 – Montauban November 3 1940) was the first Prime Minister of the Second Spanish Republic (1931–1933) and later served again as Prime Minister (1936) and then as the second and last President of the Republic (1936–1939). The Spanish Civil War broke out while he was President. With the defeat of the Republic in 1939 he fled to France resigned his office and died in exile shortly afterwards.
Proponents	OfficeHolder	**Annemarie Huber-Hotz** Annemarie Huber-Hotz (born 16 August 1948 in Baar Zug) was Federal Chancellor of Switzerland between 2000 and 2007. She was nominated by the FDP for the office and elected on 15 December 1999 after four rounds of voting. The activity is comparable to an office for Minister. The Federal Chancellery with about 180 workers performs administrative functions relating to the co-ordination of the Swiss Federal government and the work of the Swiss Federal Council.
Proponents	OfficeHolder	**José Manuel Restrepo Vélez** José Manuel Restrepo Vélez (30 December 1781 – 1 April 1863) was an investigator of Colombian flora political figure and historian. The Orchid genus Restrepia was named in his honor.Restrepo was born in the town of Envigado Antioquia in the Colombian Mid-west. He graduated as a lawyer from the Colegio de San Bartolomé in the city of Santa Fe de Bogotá. He later worked as Secretary for Juan del Corral and Governor Dionisio Tejada during their dictatorial government over Antioquia.
Proponents	OfficeHolder	**K. C. Chan** Professor Ceajer Ka-keung Chan (Traditional Chinese: 陳家强) SBS JP (born 1957) also referred as KC Chan is the Secretary for Financial Services and the Treasury in the Government of Hong Kong. He is also the ex officio chairman of the Kowloon-Canton Railway Corporation and an ex officio member of the Hong Kong International Theme Parks Board of Directors.
Opponents	Artist	**Mikołaj Rej** Mikołaj Rej or Mikołaj Rey of Nagłowice (February 4 1505 – between September 8 and October 5 1569) was a Polish poet and prose writer of the emerging Renaissance in Poland as it succeeded the Middle Ages as well as a politician and musician. He was the first Polish author to write exclusively in the Polish language and is considered (with Biernat of Lublin and Jan Kochanowski) to be one of the founders of Polish literary language and literature.
Opponents	Artist	**Justin Jeffre** Justin Paul Jeffre (born on February 25 1973) is an American pop singer and politician. A long-time resident and vocal supporter of Cincinnati Jeffre is probably best known as a member of the multi-platinum selling boy band 98 Degrees.Before shooting to super stardom Jeffre was a student at the School for Creative and Performing Arts in Cincinnati. It was there that he first became friends with Nick Lachey. The two would later team up with Drew Lachey and Jeff Timmons to form 98 Degrees.
Opponents	Artist	**David Kitt** David Kitt (born 1975 in Dublin Ireland) is an Irish musician. He is the son of Irish politician Tom Kitt.He has released six studio albums to date: Small Moments The Big Romance Square 1 The Black and Red Notebook Not Fade Away and The Nightsaver.

The first two opponents are artists and politicians, but labelled as artists. The last opponent is the son of a politician.

How can I use TrackIn to identify mislabelled examples?

There are two different ways to use TrackIn to identify mislabelled examples:

1. **Ranking by Self-influence**:
    - **a.** One method is to rank the training points in decreasing order of self-influence, and have the human inspect and relabel the top of this list. To compute self-influence, the training data point doubles as the prediction point in the Equation.
    - **b.** The premise is that other correctly labelled training points will oppose this point's incorrect label. However, the point will act as a proponent of its own label. Because it is unique in pulling in this direction, it will be a strong proponent. There are two **caveats**:
        i. The premise above fails if the mislabeling rate is very high (>30%) or if the mislabeling is correlated in a way that all the points in a neighborhood are mislabelled.

    ii. Furthermore, correctly labelled points that are very different from other points in the feature space, and therefore prone to memorization, will also have high-self influence.
  c. Due to the caveats above, human judgement is necessary to verify and correct mislabelling. (See Sections 4.3.1 and 4.4.2 from the paper for more details.)
2. **Aggregating Opponents over several test examples**:
  a. Here, we compute the top opponents for a representative set of test examples, and have a human inspect and relabel the list.
  b. The premise is that mislabelled training examples will oppose predictions of correctly labelled test points in their vicinity. We therefore expect mislabelled examples to recur in the top opponent lists of several test examples, and aggregating the opponents across a test set will tend to identify incorrectly labelled examples. There are caveats:
    i. The first caveat is that if many of the test points are themselves mislabelled, mislabelled training points will support the prediction instead of opposing it.
    ii. The second caveat is a recall issue. Mislabelled training examples that are dissimilar to any test example will probably not be caught by this analysis. If the test set is representative, fixing such recall errors may not improve model quality, because it is hard to generalize from training points that are dissimilar to all other training points.
    iii. The third caveat is that correctly labelled examples that are near decision boundaries may also be caught by such an analysis. For instance, a '7' in MNIST that looks like a '1'.
  c. Due to the first and third caveats, human judgement is again necessary to verify and fix mislabelling.

Notice that the two methods make different types of errors, and the right choice may depend on the application.

What is the difference between influence and feature similarity?

For a deep learning model, one can define a feature similarity measure in terms of the distance between embedding vectors for some layer of the network. Can we use a similarity measure to compute influence? Why do we need the concept of influence?

There are at least two reasons:

- First, feature similarity does not take the importance of the feature into account.
- Second, even if weighted by the importance of the feature, the measure would not take the training process into account correctly. It is possible that a very dissimilar example is nevertheless influential. For instance, suppose we had a regression task to predict

house prices from features such as number of bedrooms, restrooms, locality, number of residents, their income etc. Suppose that you use a linear model to do this task. Then, the coefficient of the #bedrooms feature can be affected by houses that have a large number of bedrooms, and this in turn can influence the prediction of a house with very few bedrooms.

For a practical example of the difference between similarity and influence, consider the first test example in the Figure, i.e., the kid holding the microphone. Suppose that we use embeddings from the last layer of the network, i.e., the one preceding the logits to compute similarity. Here are the results (contrast them with the influential proponents and opponents in the Figure):

We don’t use plain SGD but a different optimizer like, say, Adam. Will TrackIn work for us?

Even though our paper derives the idealized form of TrackIn for SGD, it is simple to repeat the derivation for other SGD-like optimizers, and the approximation one arrives at is identical to Equation.

How do I pick checkpoints?

The key to good quality results is to pick checkpoints around steps where the loss reduces by a large magnitude. In contrast, If you happened to pick, say, just the final checkpoint, the influence scores are likely to explain the change in loss/prediction in the vicinity of this checkpoint, and all the loss reduction activity of the model may have preceded this. And if you compute proponents and opponents with this checkpoint, you may get noisy results. It is perhaps also worth avoiding

the very initial stages of the training process, before the model has begun to make progress on reducing loss. One approach is to sample checkpoints only after the model is doing better than random guessing.

I do not have easy access to the learning rates. What should I do?

Ignore them. As long as you pick checkpoints reasonably, you should get reasonable results without weighting by learning rates (this is the η_i parameter in Equation.)

I have a lot of training data points. How can I identify top opponents and top proponents quickly?

You can precompute and store the loss gradients in Equation for the training points across different checkpoints. You can then load these vectors (one vector per training example, a concatenation of loss gradients across checkpoints) into a nearest neighbor library like Annoy. For retrieving the most influential examples, we compute a similar vector for the prediction point and perform a nearest neighbor look-up. (This nearest neighbor lookup performs a dot product that implements Equation.)

I have a model with a huge number of training parameters. What can I do to speed-up the implementation?

Instead of taking the gradients with respect to all the model parameters just pick a layer to work with. Pick a layer that models the input at sufficiently high level. This is usually a layer close to the output of the network, but not synonymous with the output. (For instance, use the weights that connect the layer before the logit layer to the logit layer.)

One can also use random projections, as described in Section 3.4 of the paper. This will not speed-up pre-processing, but will speed up identification of influential points at serving time and save on storage.

Is there an implementation of TrackIn that I can re-use? Is there a library?

TrackIn is just a few lines of code. Most of the work is in constructing the right gradient calls, by identifying the loss head, and cherry-picking checkpoints and layers. This work is model-specific and we do not have a library.

I implemented TrackIn. I am seeing strange results. What could it be?

It is possible you have a bug. But if you are seeing some strange, seemingly mislabelled training points show up as influential for several test examples, perhaps you are encountering this. Try to filter these out and re-examine your results.

How did you evaluate the technique?

We compared to two previously proposed techniques called Influence functions and Representer Point method. See Section 4 in the paper. On the tasks we evaluated on, our method seems to do better. But more importantly, it is far simpler to implement and use.

When is TrackIn not useful or hard to apply?

- TrackIn is **hard** to apply when you have a very large number of training examples for computational reasons. Models that use incremental learning for large products often have this issue. In practice, ‘very large’ is probably defined by the limits of the nearest neighbor library used for retrieval, because the rest of the computation can be done in batch.
- TrackIn is **less useful** if the influence distribution is heavy-tailed. By heavy-tailed, we mean that a large fraction of the training points are important to the prediction. This does not mean that TrackIn is only useful when the model does memorization. We mean that it is more useful if 10s of training points are influential for a prediction, because, then, deleting these points or changing their labels has influence. Whether the influence distribution is heavy tailed is an empirical issue.

What are some ways to use TrackIn within a larger analysis/system?

TrackIn is a simple, versatile technique; the influence scores can be aggregated in several ways. We list some examples of use-cases. If other use cases come to mind, feel free to send us a mail, and we will add a note with credits.

- **Finding mislabelled examples**: see here
- **Debugging an individual prediction**: see here, and Sections 4.4.1, 4.3.2 and 5 in the paper.
- **Data valuation**:
    - One framing of the data valuation problem is to identify how much a training data point contributes to reducing test loss.
    - To do this, simply aggregate the LHS of Equation over several test examples.
    - One can then aggregate the results over various categorizations of the test examples.