Distributional Training Data Attribution: What Do Influence Functions Sample?

Bruno Mlodozeniec[♦12] Isaac Reid[♦1] Sam Power[3] David Krueger[4] Murat A. Erdogdu[♣56] Richard E. Turner[♣17] Roger B. Grosse[♣56]

♦Equal contributions first authors ♣Equal contributions senior authors

[1]University of Cambridge
[2]Max Planck Institute for Intelligent Systems
[3]University of Bristol
[4]MILA – Quebec AI Institute
[5]University of Toronto
[6]Vector Institute
[7]Alan Turing Institute

Training data attribution in deep learning is a **distributional problem:** how does removing each datapoint modify the distribution over possible model weights and outputs?

We show **influence functions** are **secretly distributional**, solving this problem more generally than previously thought for *non-convex* losses.

Training data attribution is distributional

Classical **T**raining **D**ata **A**ttribution (**TDA**) **asks**: *how would the* ***deterministic*** *outcome of model training differ if select examples were removed from the training dataset?*

Classical formulation does not match the deep learning setting. Here, the problem is **inherently distributional** – due to the noise in initialisation and mini-batching, training outcomes are a **random variable.**

We propose ***Distributional*** **TDA** that asks: *how would the* ***distribution*** *over outcomes of model training differ if select examples were removed from the training dataset?*

"Classical" Data Attribution	Distributional Data Attribution
Deterministic training algorithms θ^*:	*Stochastic* training algorithms θ^*:
θ^*: `datasets` → `model parameters`	θ^*: `datasets` → `distribution over model parameters`
Goal: Approximate $\theta^*(\mathcal{D}')$ (using the value of $\theta^*(\mathcal{D})$) without running θ^*.	*Goal*: Approximately sample from $\theta^*(\mathcal{D}')$ without running θ^*.

Influence functions are distributional

Influence functions (IFs) are a *classical* TDA method that use the curvature of the loss function to *efficiently* estimate how the minimum would move if perturbed as $H^{-1}\nabla\ell_k(z_k,\theta^*)$.

In the literature, **IFs are typically derived under restrictive convexity assumptions** – at odds with how well they work in deep learning.

Key insight: IFs are **implicitly distributional.** They already 'solve' the distributional TDA problem **under a much more general set of conditions** (for loss functions that can be non-convex, have multiple minima)

In what sense are influence functions distributional?

Unrolled differentiation differentiates through the training trajectory to get sensitivity of final model parameters with respect to dataset perturbation ε***:***

$$\frac{\mathrm{d}\theta_{t+1}}{\mathrm{d}\varepsilon} = \frac{\mathrm{d}\theta_t}{\mathrm{d}\varepsilon} + \frac{\partial}{\partial\varepsilon}\,\texttt{SGDUpdate}\,(\theta_t, \varepsilon, \mathcal{D})$$

Unrolled differentiation already solves the TDA task by *approximately sampling* from the model distribution trained on a perturbed dataset

We formally show — in the limit of sufficiently long training — unrolled differentiation and IFs converge.

Hence, influence functions are distributional too!

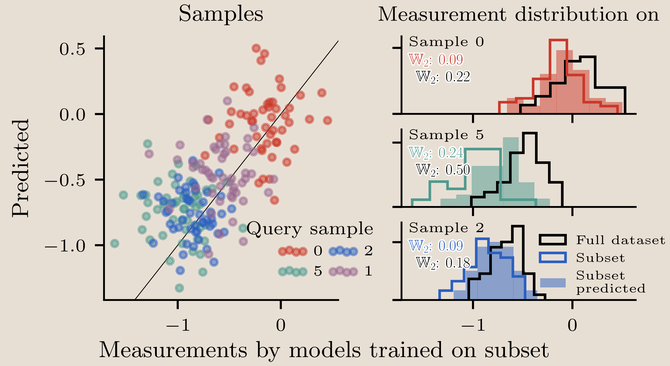


Figure 2: We empirically show existing TDA methods (unrolled differentiation, IFs) can predict *distributional* changes to training outcomes.

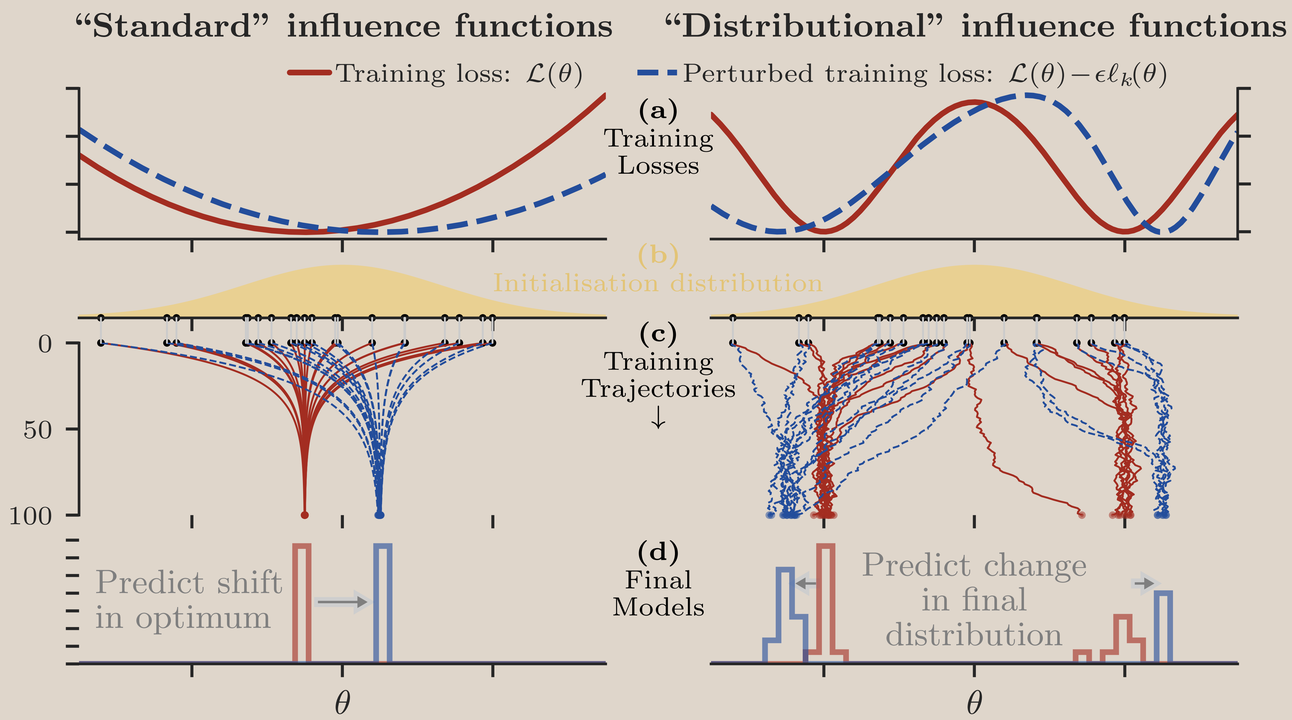
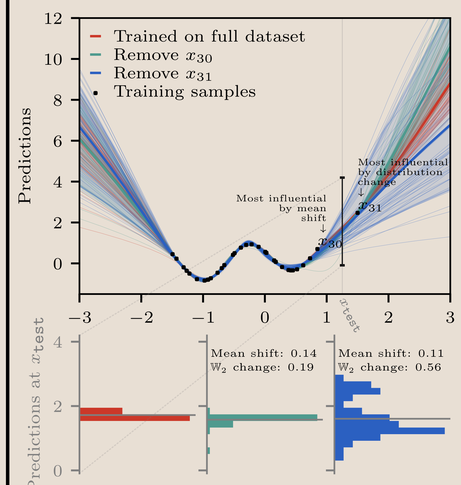


Figure 1: The goal of *classical* data attribution vs. the *distributional* data attribution. We show influence functions approximately solve the latter problem *under a more general set of conditions* (non-convex losses with multiple degenerate minima).

Distributional insights on TDA:

1 In stochastic settings, we need to be more precise about how we define influence:



Depending of how we measure discrepancy between distributions, different data might be most influential.

2 Distributional TDA gives better downstream performance. Data pruning for vision transformers with influence functions *is more effective using distributional influence,* compared to adopting "classical" influence by fixing all sources of randomness.

3 Influence functions underperform at leave-one-out because prior work didn't use enough samples. When viewed as a distributional method, Ifs can do leave-one-out.

Take-aways:

Our framework enables a better understanding of:

- How influence functions work in deep learning
- Formal connections between unrolled differentiation and influence functions
- How data attribution methods should be empirically evaluated
- What the goal of data attribution should be in deep learning