How many trained neural networks are needed for influence estimation in modern deep learning?

Anonymous Author(s)
Affiliation
Address
email

Abstract

Influence estimation attempts to estimate the effect of removing a training example on downstream predictions. Prior work has shown that a first-order approximation to estimate influence does not agree with the ground-truth of re-training or fine-tuning without a training example. Recently, Feldman and Zhang [2020] created an influence estimator that provides meaningful influence estimates but requires training thousands of models on large subsets of a dataset. In this work, we explore how the method in Feldman and Zhang [2020] scales with the number of trained models. We also show empirical and analytical results in the standard influence estimation setting that provide intuitions about the role of nondeterminism in neural network training and how the accuracy of test predictions affects the number of models needed to detect an influential training example. We ultimately find that a large amount of models are needed for influence estimation, though the exact number is hard to quantify due to training nondeterminism and depends on test example difficulty, which varies between tasks.

1 Introduction

Influence functions are used to estimate the effect of removing a single training example from the training set on downstream predictions. This can be used for interpretability, analysing datasets, and detecting train-test leakage in datasets, among other applications.

While influence functions had previously been used in the field of statistics, they were re-introduced to deep learning in the seminal work by Koh and Liang [2017], where they use a Taylor approximation around a trained model to approximate the effect of removing one training example. However, as deep neural networks are highly non-linear in data and weights, this Taylor approximation tend be loose. Recent work by Basu et al. [2021] has shown that the influence function estimators introduced in Koh and Liang [2017] give erroneous estimates for modern neural networks and datasets when compared to re-training or fine-tuning without a datapoint. A different approach is taken by Feldman and Zhang [2020] who explicitly train thousands of models with/without training examples and are able to find more meaningful influence estimators with this expensive training process.

The work in Feldman and Zhang [2020] shows that meaningful influence estimators *can* be constructed, however, it requires training thousands of models. These models are trained on randomly selected subsets of the training set (of the same size). The effect of a training point on a prediction can then be computed as the difference in average predictions of models trained with/without a specific training example.

Our initial goal in this line of work was to reduce the amount of models required to perform influence estimation. We start by extending the analysis from Feldman and Zhang [2020] to analyze how the consistency of influence estimates changes with the number of trained models in their method. We also look at how the nondeterminism in neural network training affects these influence estimates.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

We also consider the standard influence estimation setting, where models are trained over the full dataset rather than a subset. In this setting, we take a closer look at nondeterminism across a variety of classification datasets and models, compare model fine-tuning with re-training and provide mathematical and empirical analysis to compute the amount of models needed to find an influential training point.

2 Leave-one-out influence background

2.1 Influence Definition

Influence estimation attempts to quantify the effect of removing a training example on the downstream prediction of a learned neural network. In this work we are concerned with *leave-one-out* influence, where a training example is explicitly removed, rather than other cases where a training example is modified (perhaps adversarially). While various estimators exist for this quantity, these methods are attempting to estimate the *true* ground truth influence function, which we define below.

Formally, let S be a training dataset consisting of n labelled examples (x_i, y_i). Also, let A be an algorithm that outputs a hypothesis function h, typically a neural network . We consider the effect of removing the ith training point $(x_i, y_i) \in S$ on the prediction of example $z = (x, y)$, with loss function L.

The (true) influence function is then defined as:

$$\text{infl}(A, S, i, z) \coloneqq L_{h \leftarrow A(S)}(h(x), y) - L_{h \leftarrow A(S \setminus x_i)}(h(x), y)$$

However, following Feldman and Zhang [2020], we constrain our analysis to influence for classification tasks, hence specify L as the 0-1 classification loss, and take into account the nondeterministic properties of A:

$$\text{infl}(A, S, i, z) \coloneqq \mathbb{P}_{h \leftarrow A(S)}[h(x) = y] - \mathbb{P}_{h \leftarrow A(S \setminus x_i)}[h(x) = y] \tag{1}$$

2.2 Model Agreement

One simple way to compare the two hypothesis functions is by comparing their output predictions over the same set of inputs. This is especially relevant in the context of influence functions, which also compute quantities at a data point level.

Formally, if we have two hypothesis functions, h_1 and h_2, their agreement on a set of points s (where $(x_i, y_i) \in s$) is defined as:

$$\text{Agreement}(h_1, h_2, s) \coloneqq \frac{1}{|s|} \sum_{i=1}^{|s|} \mathbb{1}[h_1(x_i) = h_2(x_i)]$$

2.3 Overview of Feldman and Zhang [2020]

Rather than computing the influence in Eqn. 1 with respect to the training set S (of size n), Feldman and Zhang [2020] compute the influence for a random subset of m training examples sampled from S (where $m < n$). Let $[n]$ denote the set of indices of all training examples, $I \sim \mathbb{P}([n] \setminus i, m-1)$ is a set of $m-1$ samples drawn uniformly (without replacement) from $[n] \setminus i$. They define this influence estimator as follows:

$$\text{infl}_{\text{m}}(A, S, i, z) \coloneqq \mathbb{E}_{I \sim \mathbb{P}([n] \setminus i, m-1)}[\text{infl}(A, S_{I \cup i}, i, z)] \tag{2}$$

The practical algorithm is as follows (implemented differently than in Eqn. 2 but shown to be equivalent in Feldman and Zhang [2020]):

1. From a dataset of length n, sample a subset J of length m, picking examples uniformly.
2. Repeat this sampling procedure t times, and train a model per subset J
3. Now, from the t models, select the models trained with example i, compute their average (binary) prediction on a test point z, and select models trained without example i (again computing the average prediction on z) and return the difference of the averaged terms

2.3.1 IOU Comparisons of Influence Scores

The authors also provide a useful statistic to compare pairs of influence estimates (also known as influence scores), $\widetilde{\text{infl}}_{\text{m}}(A, S, i, z)$ and $\widetilde{\text{infl}'}_{\text{m}}(A, S, i, z)$. A set of influential examples is defined as all pairs of training examples i and test examples j in a dataset above a threshold θ, or formally: $I(\theta) = \{(i, j) : \widetilde{\text{infl}}_{\text{m}}(A, S, i, z_j) \geq \theta\}$ (and $I'(\theta) = \{(i, j) : \widetilde{\text{infl}'}_{\text{m}}(A, S, i, z_j) \geq \theta\}$). Then two pairs of influence estimates can be compared by their corresponding influential example sets with their *intersection over union* (IOU) above an influence threshold: $\text{IOU}(I, I', \theta) = \frac{|I(\theta) \cap I'(\theta)|}{|I(\theta) \cup I'(\theta)|}$.

3 How does the method in Feldman and Zhang [2020] scale with the number of trained models?

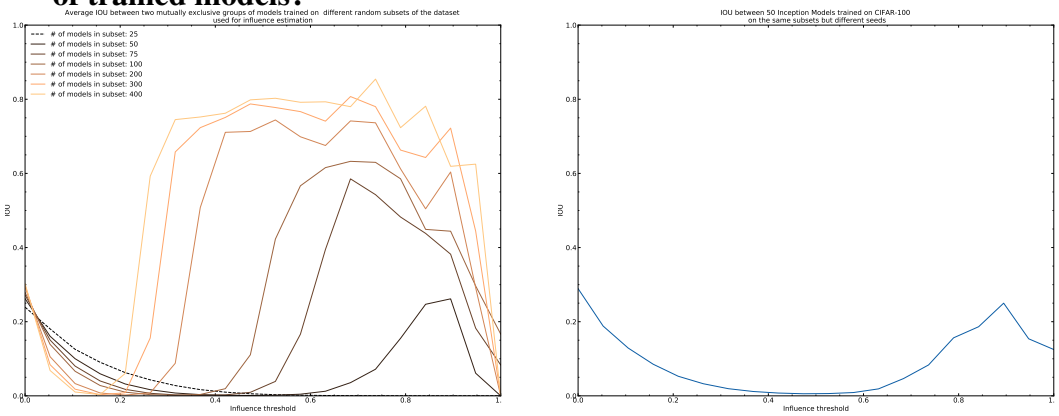


Figure 1: Left: Comparing influence score consistency between t models trained on different sampled subsets I *(higher IOU is better)* Right: Comparing influence scores using t models compared to the full 2000 models trained in Feldman and Zhang [2020]

We first perform an empirical study to better understand how many models are needed to estimate influence scores in Feldman and Zhang [2020]. Note that in their approach, each model is trained on a different randomly selected subset J of size m, in-practice $m/n = 0.7$. We use the 2000 publicly released Inception models from Feldman and Zhang [2020] trained on CIFAR-100 for this analysis.

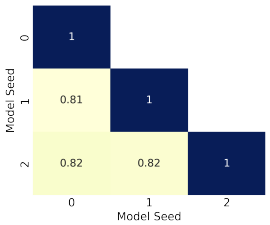
First, we analyze whether the choice of random data subsets in Step 2. of the algorithm affects the self-consistency of influence estimates. We compare a pair of influence estimates by their IOU, each computed with the same number of models t, but with no overlap in the random subsets used for training the models (ie. there are no shared Js across the pair). We compute the average IOU by repeating this process for 6 pairs of influence estimates and vary t and plot this in Figure 1 (left).

As seen in 1, the IOU is extremely low when t is low. The IOU for 25 models is near the 0 mark on the y axis (dashed to be visible). In fact, we see that with 100 models the IOU scores peak at 0.6 similarity, meaning that $> 40\%$ of the two influence sets are non-overlapping. In the range of 200-400 models, we see more consistent IOU scores for a wider range of thresholds. Hence, to get self-consistent influence estimates with the method from Feldman and Zhang [2020], we require training > 100 models, likely in the range of > 200 models for more self-consistent estimates.

In Figure 1 (right), we try to disentangle the choice of subsets for the t models from the nondeterminism associated with neural network training. In this experiment, we form a pair of influence estimates by training 50 Inception models on the *same* random data subsets J, but with different seeds, effectively changing data order, model initialization and data augmentations. Surprisingly, we see extremely low IOU scores across the influence thresholds, suggesting that the nondeterminism in neural network training poses significant challenges for influence estimation.

4 Simplifying the influence estimation setting

We previously saw that there is significant inconsistency in influence estimates between the *same* subsets in Feldman and Zhang [2020] due to differences in the random seeds during training. Hence,



(a) Model agreement example

Dataset	Model	Average Agreement	Mean Accuracy
CIFAR-10	Inception	0.947 ± 0.00	0.942 ± 0.00
	ResNet-18	0.959 ± 0.00	0.952 ± 0.00
CIFAR-100	Inception	0.754 ± 0.00	0.742 ± 0.00
	ResNet-18	0.818 ± 0.00	0.778 ± 0.00
DTD	Inception	0.405 ± 0.02	0.301 ± 0.01
	ResNet-18	0.548 ± 0.01	0.372 ± 0.01
Flowers	Inception	0.519 ± 0.02	0.542 ± 0.02
	ResNet-18	0.668 ± 0.01	0.513 ± 0.01

(b) Agreement and accuracy for models trained on *identical* data

Figure 2: Average agreement and accuracy for a variety of datasets and models, and simple example of model agreement with 3 ResNet-18 models on CIFAR-100

in this section we eliminate the concept of sampled subsets and consider standard influence estimation when $m = n$, i.e. when we train models over the entire dataset. This is simpler as it eliminates the hyperparameter m and our values for influence estimates are the "true" influence values, rather than being approximated by a smaller subset of the dataset.

4.1 Quantifying the nondeterminism of the training algorithm A

As seen in Section 3, nondeterminism associated with the training algorithm can cause significant differences in influence estimations. Here, we focus exclusively on quantifying the effects of nondeterminism in neural network training across a variety of classification datasets and models, where each experiment is run across 3 seeds.

Figure 2a is a simple example of measuring model agreement as defined in Section 2.2, where we look at the pairwise agreement in predictions for ResNet-18 on CIFAR-100. The average agreement is then the average over the 3 light green squares (ie. not including the diagonal terms).

As seen in Table 2b, models trained over the *same* data show relatively low model agreement. This is especially true for more challenging tasks such as the DTD and Flowers datasets, where the mean accuracy is lower and models have agreement from 40-70%. This means that for more challenging datasets, the effect of removing a single training example is difficult to observe compared to the nondeterminism associated with the training algorithm.

We include more extensive experiments where we vary the model initialization and data order independently in Appendix C, but we find that regardless of the source of nondeterminism, a similar level of model agreement is observed.

4.2 Fine-tuning vs. Re-training

Prior to Feldman and Zhang [2020], many prior works in influence estimation relied on fine-tuning to generate ground truth such as Koh and Liang [2017], Basu et al. [2021]. Here, a model is first trained until convergence on the full dataset. Then, to approximate the effect of removing a datapoint, a training example is removed from the dataset, and the model is fine-tuned for 6% of the total training time.

In our experiments, we compare this fine-tuning procedure to training from scratch without a datapoint. We compare the test predictions and accuracies of the fine-tuned vs. fully re-trained models (averaging over 3 seeds), on CIFAR-100 and the Flowers dataset and Inception and ResNet-18 models when dropping 10 randomly sampled training points. While we considered a more complex scheme to select the 10 training points to drop, we decided that randomly chosen points would still be indicative for the purposes of comparing fine-tuning and re-training.

Table 1 shows the results of our experiments. Evidently, there is significant model disagreement between the re-trained and fine-tuned models, showing that this is not a viable strategy for generating influence estimates. In Basu et al. [2021], the authors compare the fine-tuning and re-training strategies by measuring the difference in the norm of the parameters of model. However, we believe that model agreement is a more indicative metric, since influence functions themselves explicitly compare the pointwise predictions of models with and without a point.

Dataset	Model	R vs F Agreement	Retrain (R) Accuracy	Fine-tune (F) Accuracy
CIFAR-100	Inception	0.756 $\pm$ 0.01	0.740 $\pm$ 0.01	0.742 $\pm$ 0.00
	ResNet-18	0.820 $\pm$ 0.00	0.777 $\pm$ 0.00	0.778 $\pm$ 0.00
Flowers	Inception	0.544 $\pm$ 0.04	0.538 $\pm$ 0.03	0.542 $\pm$ 0.02
	ResNet-18	0.671 $\pm$ 0.02	0.511 $\pm$ 0.01	0.512 $\pm$ 0.01

Table 1: Average agreement for predictions between a model fine-tuned for 6% of the total epochs and re-trained without a data point, across 3 seeds and 10 different dropped points

4.3 An analysis of the number of models needed to detect an influence score

Here, we take a more mathematical approach to try and quantify how many models are needed to estimate an influence score. We now define two Bernoulli parameters p and q to represent the quantities in the influence estimation task: $p = \mathbb{P}_{h \leftarrow A(S)}[h(x) = y]$ and $q = \mathbb{P}_{h \leftarrow A(S \setminus x_i)}[h(x) = y]$. For simplicity, we denote $h \leftarrow A(S)$ as case X and $h \leftarrow A(S \setminus x_i)$ as case Y.

As shown in Section 4.1, the variance associated with estimating p and q through limited samples often obscures the difference between p and q, rendering the influence value indiscernible. A natural question is to ask how many models does one need to train in order to confidently discern X from Y. This question can be distilled into the following coin-flipping problem: given a biased coin and two possible probabilities of heads p and q, how many coin flips does one need to perform to confidently determine whether the coin follows p or q (assuming one of them is the correct probability). Note that we need to know p and q a-priori to form the decision with confidence: for example, if $p = 1$ and $q < 1$, then one only need to observe one tails to conclude that the coin follows q with 100% confidence.

A similar mathematical analysis was done for a simple coin flipping problem by BruceET, which we rewrite here in the context of influence estimation. First, we find the fraction of times that $h(x) = y$ under X and $h(x) = y$ under Y where these probabilities are equal. As done by BruceET we denote this quantity as m (where $m \in (0, 1)$ and $p < m < q$, assuming $p < q$ WLOG), which can be found by solving: $p^m(1-p)^{1-m} = q^m(1-q)^{1-m}$.

Next, we wish to find the minimum number of trials t and number of times that $h(x) = y$, which we denote as k such that $\mathbb{P}(k < mt|X) \geq 1 - \delta$ and $\mathbb{P}(k < mt|Y) \leq \delta$. We compute the CDF of the binomial distribution over an exhaustive search of the values of t until the above condition is met, and plot the minimum number of trials to distinguish them with a δ value of 0.05. Our results are presented in Figure 3 (left). Here we see that the more similar p and q are, the more models are needed, as expected. Note, that unlike the analysis for Feldman and Zhang [2020], this plot shows how many models *per train-test point pair* would need to be trained. For example, with $p = 0.6$ and $q = 0.4$, we see that > 100 models would need to be trained for that example to detect the influence score.

Figure 3 (middle) shows an alternative view on how the number of models scales to detect a certain influence score. In particular, we see that smaller influence scores require significantly more models to detect. This is promising for applications where we care about the *high-influence* training examples and shows that these can be detected with far less models than small influence scores.

Lastly, Figure 3 (right) shows the empirical distribution of p on CIFAR-100 for the Inception network. Here, we averaged the bernoulli prediction $h(x) = y$ for each test example z over 20 seeds and plot its empirical distribution. The majority of examples have p close to 1, suggesting that influence estimation for these scores should require less models (if we compare with our plot on the left, we can see that p ranging from 0.9-1 corresponds to < 10 models for a wide range of q values). However, we do see a fairly long tail for p, suggesting that for these examples, significantly more models may be necessary.

Overall, this analysis provides some intuitions about how many models are truly needed for influence estimation. The answer is not clear-cut and depends on the test accuracy of an example but this mathematical framework is a first step in addressing this question. The code used to generate the binomial estimates is found in Appendix F.

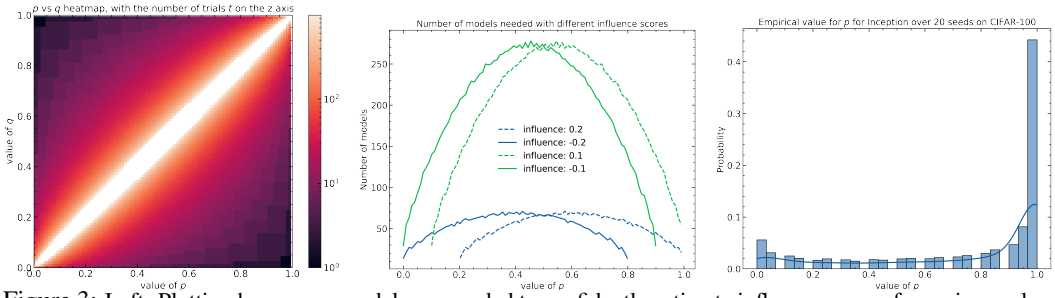


Figure 3: Left: Plotting how many models are needed to confidently estimate influence scores for various values of p and q. Middle: Showing how the number of models scales with different influence scores and values of p. Right: the empirical distribution for p for the Inception model on CIFAR-100 (averaged over 20 seeds)

5 Limitations

The experiments in our work are limited to smaller computer vision datasets and two families of architectures, so there may be different conclusions to be drawn when analyzing influence estimation on larger datasets or for different modalities such as language. Our mathematical analysis is not particularly formal and is meant to provide an intuition regarding the number of models needed for influence estimation, rather than a hard rule. In general, we view our work as a stepping stone and useful analysis for future researchers, given the challenging nature of influence estimation due to nondeterminism and the need to run large sets of experiments.

6 Related Work

Our work is related to prior work in the field of influence estimation, which has been an active area of research in deep learning. There have been a multitude of influence estimation techniques that have been proposed, such as Koh and Liang [2017], Feldman and Zhang [2020], Pruthi et al. [2020], Guo et al. [2020], Achille et al. [2021], Kobayashi et al. [2020]. Our work is highly inspired by the method in Feldman and Zhang [2020], which to our knowledge, is the first work that proposes training a large amount of models to perform more robust influence estimation. We also take inspiration from the meta-analysis performed in Basu et al. [2021], although that work is centered around demonstrating the fragility of the influence function estimation method introduced by Koh and Liang [2017], while our work is more interested in “true" leave-one-out influence estimation that actually performs model re-training rather than using an approximation.

Our work also has ties to nondeterminism in neural network training, most notably Summers and Dinneen [2021], who perform extensive experiments to understand different sources of noise in the training process and how that affects model predictions. Other work in this area includes Somepalli et al. [2022], Zhuang et al. [2021].

Another recent work that uses a large amount of models trained on random subsets to understand the effects of data is Ilyas et al. [2022] who construct *datamodels* to estimate how subsets of training examples affect the output predictions of neural networks, by training 600,000 - 1,500,000 models on data subsets. However, *datamodels* are a more recent development and are not as well established as influence functions, so this lies outside the scope of our work.

7 Conclusion

In our work, we perform a number of experiments to try and understand how influence estimation scales with the number of trained models. The need for multiple models is fundamentally due to the nondeterminism associated with neural network training. While we do not explicitly answer the question posed by the title of our paper, *How many trained neural networks are needed for influence estimation in modern deep learning?*, our work does give both empirical and analytical intuitions as to how many models are needed for standard benchmarks, which lie in the order of hundreds of models for the method introduced by Feldman and Zhang [2020] and is training example dependent in the case of standard vanilla estimation (but can range to hundreds of models).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section G
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 4.3
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide extensive experimental details in Section A.1 and plan to release the code upon acceptance.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See A.1
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Our results are highly concerned with nondetermenism, so we average over multiple seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section A.1, though we were not able to estimate the total amount of compute sensibly.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section A.1
   (b) Did you mention the license of the assets? [Yes] See Section A.1
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

A. Achille, A. Golatkar, A. Ravichandran, M. Polito, and S. Soatto. Lqf: Linear quadratic fine-tuning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15724–15734, 2021.

S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile, 2021.

BruceET. How to determine the number of coin tosses to identify one biased coin from another? Mathematics Stack Exchange. URL `https://math.stackexchange.com/q/2033739`. URL:https://math.stackexchange.com/q/2033739 (version: 2016-11-28).

M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.

V. Feldman and C. Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation, 2020.

H. Guo, N. F. Rajani, P. Hase, M. Bansal, and C. Xiong. Fastif: Scalable influence functions for efficient model interpretation and debugging. *CoRR*, abs/2012.15781, 2020. URL `https://arxiv.org/abs/2012.15781`.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

A. Ilyas, S. M. Park, L. Engstrom, G. Leclerc, and A. Madry. Datamodels: Predicting predictions from training data, 2022. URL `https://arxiv.org/abs/2202.00622`.

S. Kobayashi, S. Yokoi, J. Suzuki, and K. Inui. Efficient estimation of influence of a training instance. *ArXiv*, abs/2012.04207, 2020.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions, 2017. URL `https://arxiv.org/abs/1703.04730`.

S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better?, 2018. URL `https://arxiv.org/abs/1805.08974`.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

G. Pruthi, F. Liu, M. Sundararajan, and S. Kale. Estimating training data influence by tracking gradient descent. *CoRR*, abs/2002.08484, 2020. URL `https://arxiv.org/abs/2002.08484`.

G. Somepalli, L. Fowl, A. Bansal, P. Yeh-Chiang, Y. Dar, R. Baraniuk, M. Goldblum, and T. Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective, 2022. URL `https://arxiv.org/abs/2203.08124`.

C. Summers and M. J. Dinneen. Nondeterminism and instability in neural network optimization, 2021. URL `https://arxiv.org/abs/2103.04514`.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL `http://arxiv.org/abs/1512.00567`.

D. Zhuang, X. Zhang, S. L. Song, and S. Hooker. Randomness in neural network training: Characterizing the impact of tooling, 2021. URL `https://arxiv.org/abs/2106.11872`.

A Appendix

A.1 Experimental Details

A.1.1 Datasets

We perform experiments on the standard computer vision datasets CIFAR-10 and CIFAR-100 Krizhevsky [2009]. We also train models on fine-grained classification datasets: Describable Textures Dataset (DTD) Cimpoi et al. [2014] and Flowers 102 Nilsback and Zisserman [2008].

The licensing information for CIFAR is unclear, however the authors request the citation of their work for use of the dataset. Describable Textures Dataset is released for research purpose only. Flowers 102 is licensed under the GNU General Public License, version 2.

A.2 Models

For the fine-grained classification datasets, we use the standard ResNet-18 He et al. [2015] and Szegedy et al. [2015] implementations found in the **torchvision** library.

For the CIFAR datasets, the training images have a height and width of 32x32 pixels, so we adapt the ResNet-18 for CIFAR, following the convention from He et al. [2015], replacing the first 7x7 convolution with a 3x3 convolution and eliminating the first maxpool operation. We also use a small Inception variant proposed by Feldman and Zhang [2020], with their implementation found on GitHub.

A.3 Training details

CIFAR-10 and CIFAR-100: We train for 200 epochs, using SGD and a learning rate of 0.1, weight decay of $5.e-4$, batch size of 128, and we reduce the learning rate by a factor of 10 at epochs $[60, 120, 160]$.

We use a standard random crop operation with padding 4 and a horizontal flip in our data augmentation pipeline.

DTD and Flowers: We train for 400 epochs with an effective batch size of 256 (using gradient accumulation where applicable). We apply a cosine learning rate decay throughout training.

We use the standard ImageNet augmentation pipeline (implemented with **randomResizedCrop** and a horizontal flip) and normalize the images according to the mean and variance of the datasets. The input dimensinons for the images are 224x224 for ResNet-18 and 299x299 for Inception v3.

To find the learning rate and weight decay to use for SGD, we follow the hyperparameter sweep defined in Kornblith et al. [2018]: "Thus, our grid consisted of 7 logarithmically spaced learning rates between 0.001 to 1.0 and 7 logarithmically spaced weight decay to learning rate ratios between 10^{-5} to 10^{-2}, as well as no weight decay." (Note that in the original text, the quoted text had the values for optimal fine-tuning, rather than training from scratch, so we have replaced the ranges of values with those meant for training from scratch found later in the text).

A.4 Compute details

We ran experiments on P100 GPUs on an internal cluster, using the PyTorch framework by Paszke et al. [2017] and Tensorflow Abadi et al. [2015].

B Test-Time Augmentation (TTA)

In Summers and Dinneen [2021], the authors attempt to stabilize neural network predictions by performing accelerated ensembling and test-time augmentation. Since accelerated ensembling requires modifying the training procedure (in Summers and Dinneen [2021], they use a cyclical learning rate), we instead explore the use of test-time augmentation for stablizing test predictions.

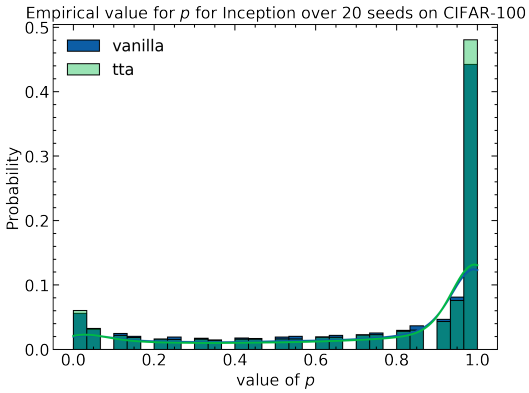


Figure 4: Plot showing how TTA affects the empirical distribution of p values per model point on CIFAR-100

In particular, for each test image, we create 10 augmented images with a horizontal flip and 5 crops of the original test image. We then take the mean over the logits to obtain an average prediction. We use the TTach PyTorch library for this implementation.

Table 2 shows slight improvements across the model agreement between 3 seeds of the network, however, these are not very significant, especially for the more complex Flowers dataset. Similarly, Figure 4 shows the empirical distribution of p values on CIFAR-100 with Inception, which only marginally changes when using TTA.

Dataset	Model	Average Agreement	Mean Accuracy	TTA
CIFAR-100	Inception	0.754 ± 0.00	0.742 ± 0.00	
		0.777 ± 0.00	**0.753 ± 0.00**	✓
	ResNet-18	0.818 ± 0.00	0.778 ± 0.00	
		0.837 ± 0.00	**0.785 ± 0.00**	✓
Flowers	Inception	0.519 ± 0.02	0.542 ± 0.02	
		0.522 ± 0.02	**0.544 ± 0.02**	✓
	ResNet-18	0.668 ± 0.01	0.513 ± 0.01	
		0.669 ± 0.01	**0.518 ± 0.01**	✓

Table 2: Average agreement and accuracy with test-time augmentation

C Comparing how sources of nondeterminism affect model agreement

Table 3 shows the results of changing different sources of nondeterminism (initialization and data order) on the agreement and accuracy of models across different datasets. This agrees with prior work in Summers and Dinneen [2021], who also found that models show similar levels of model disagreement and variance in the predictions, regardless of the source of nondeterminism.

D Using softmax predictions instead of hard (0-1) predictions

One of the design choices made in Feldman and Zhang [2020] is the use of hard (0-1) predictions in finding influence estimates. One hypothesis may be that it is necessary to train a lot of models because we are receiving a discrete bernoulli variable corresponding to $h(x) = y$ per model (which requires a lot of sampling). Hence, we use the softmax score, given by $h(x)_y$, corresponding to a normalized probability from 0-1 for the correct class.

Figure 5 shows that using softmax outputs *does* seem to result in more consistent influence scores, as measured by IOU, especially for very high influence scores. However, if we treat the 0-1 outputs as the "ground-truth", in the same fashion as Feldman and Zhang [2020], then after a certain amount of models (100-200), it appears that the 0-1 bernoulli variables are a better sampling strategy than using

Dataset	Model	Init/Order Settings	Average Agreement	Mean Accuracy
CIFAR-10	Inception	Diff init	0.947 ± 0.00	0.943 ± 0.00
		Diff data order	0.950 ± 0.00	0.944 ± 0.00
		Diff init, Diff data order	0.947 ± 0.00	0.942 ± 0.00
	ResNet-18	Diff init	0.959 ± 0.00	0.952 ± 0.00
		Diff data order	0.958 ± 0.00	0.953 ± 0.00
		Diff init, Diff data order	0.959 ± 0.00	0.952 ± 0.00
CIFAR-100	Inception	Diff init	0.753 ± 0.00	0.739 ± 0.01
		Diff data order	0.744 ± 0.01	0.734 ± 0.01
		Diff init, Diff data order	0.754 ± 0.00	0.742 ± 0.00
	ResNet-18	Diff init	0.818 ± 0.00	0.778 ± 0.00
		Diff data order	0.820 ± 0.00	0.780 ± 0.00
		Diff init, Diff data order	0.818 ± 0.00	0.778 ± 0.00
DTD	Inception	Diff init	0.413 ± 0.01	0.308 ± 0.00
		Diff data order	0.412 ± 0.01	0.329 ± 0.02
		Diff init, Diff data order	0.405 ± 0.02	0.301 ± 0.01
	ResNet-18	Diff init	0.564 ± 0.00	0.386 ± 0.01
		Diff data order	0.566 ± 0.01	0.368 ± 0.00
		Diff init, Diff data order	0.548 ± 0.01	0.372 ± 0.01
Flowers	Inception	Diff init	0.585 ± 0.02	0.547 ± 0.01
		Diff data order	0.408 ± 0.08	0.462 ± 0.10
		Diff init, Diff data order	0.519 ± 0.02	0.542 ± 0.02
	ResNet-18	Diff init	0.672 ± 0.02	0.507 ± 0.01
		Diff data order	0.671 ± 0.01	0.511 ± 0.01
		Diff init, Diff data order	0.668 ± 0.01	0.513 ± 0.01

Table 3: Average model agreement across datasets and associated accuracy, for models trained on *identical* data

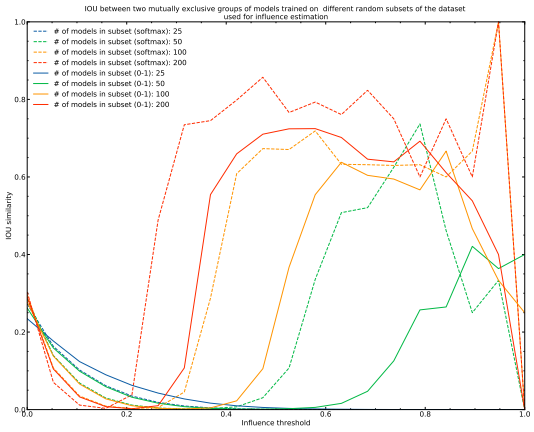


Figure 5: Comparing using softmax probabilities instead of hard 0/1 outputs

softmax probabilities, as seen in Figure 6. However, this is still an interesting future direction for future work.

E Note about IOU similarity

In Feldman and Zhang [2020], a large focus of their work is on memorization, which is essentially defined as self-influence, where the "test point" z is x_i itself, ie. $\text{infl}(A, S, i, (x_i, y_i))$. When computing the IOU similarity, they have a constraint to only consider pairs (i, j) where the memorization score is > 0.25. We also follow this convention in our work, but omit this in the main body for brevity.

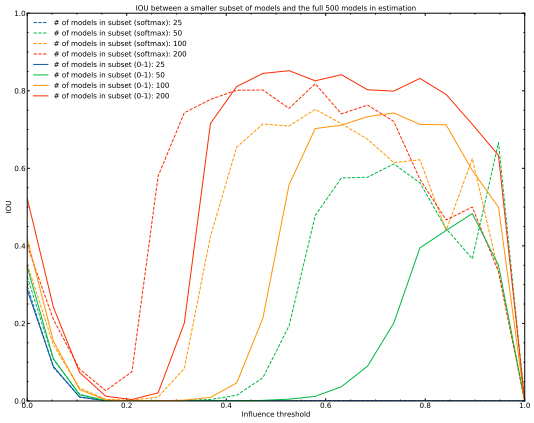


Figure 6: Comparing using softmax probabilities instead of hard 0/1 outputs against full distribution of influence scores computed using **0-1** outputs

F Binomial Simulation Code

```
import scipy.stats

def compute_num_trials(p, q, threshold):
  max_trial_num = 1000
  if p == q:
    return float("inf")

  if p > q:
    p, q = q, p
  for i in range(1, max_trial_num):

    if compute_if_within_threshold_in_n_trials(p, q, i, threshold):
      return i
  return float("inf")

def compute_if_within_threshold_in_n_trials(p, q, n, threshold):

  cutoff_point = n * np.log((1 - q)/(1-p)) / np.log(p*(1 - q)/(q * (1 - p)))

  result_for_p = scipy.stats.binom.cdf(k=cutoff_point,n=n,p=p,loc=0)
  result_for_q = scipy.stats.binom.cdf(k=cutoff_point,n=n,p=q,loc=0)

  return result_for_p >= 1 - threshold and result_for_q <= threshold
```

G Broader impact

Work on influence estimation is useful for interpretability purposes for deep learning systems and could be used to gain insights and function alongside a human decision-maker. Our work does not introduce an influence estimation method, rather we explore and provide insights into existing methods, as well as their computational requirements. Reducing the computational requirements for neural network training is positive for both ML practitioners and the environment (via resources used for electricity, etc.) so our work could be useful in that regard as well.