If Influence Functions are the Answer, Then What is the Question?

Juhan Bae[1,2], **Nathan Ng**[1,2,3], **Alston Lo**[1,2], **Marzyeh Ghassemi**[3], **Roger Grosse**[1,2]
[1]University of Toronto, [2]Vector Institute, [3]Massachusetts Institute of Technology
{jbae, nng, rgrosse}@cs.toronto.edu
alston.lo@mail.utoronto.ca
mghassem@mit.edu

Abstract

Influence functions efficiently estimate the effect of removing a single training data point on a model's learned parameters. While influence estimates align well with leave-one-out retraining for linear models, recent works have shown this alignment is often poor in neural networks. In this work, we investigate the specific factors that cause this discrepancy by decomposing it into five separate terms. We study the contributions of each term on a variety of architectures and datasets and how they vary with factors such as network width and training time. While practical influence function estimates may be a poor match to leave-one-out retraining for nonlinear networks, we show that they are often a good approximation to a different object we term the *proximal Bregman response function (PBRF)*. Since the PBRF can still be used to answer many of the questions motivating influence functions such as identifying influential or mislabeled examples, our results suggest that current algorithms for influence function estimation give more informative results than previous error analyses would suggest.

1 Introduction

The influence function [Hampel, 1974, Cook, 1979] is a classic technique from robust statistics that estimates the effect of deleting a single data example (or a group of data examples) from a training dataset. Formally, given a neural network with learned parameters $\boldsymbol{\theta}^\star$ trained on a dataset $\mathcal{D}$, we are interested in the parameters $\boldsymbol{\theta}^\star_{-\mathbf{z}}$ learned by training on a dataset $\mathcal{D} - \{\mathbf{z}\}$ constructed by deleting a single training example $\mathbf{z}$ from $\mathcal{D}$. By taking the second-order Taylor approximation to the cost function around $\boldsymbol{\theta}^\star$, influence functions approximate the parameters $\boldsymbol{\theta}^\star_{-\mathbf{z}}$ without the computationally prohibitive cost of retraining the model. Since Koh and Liang [2017] first deployed influence functions in machine learning, influence functions have been used to solve various tasks such as explaining model's predictions [Koh and Liang, 2017, Han et al., 2020], relabelling harmful training examples [Kong et al., 2021], carrying out data poisoning attacks [Koh et al., 2022], increasing fairness in models' predictions [Brunet et al., 2019, Schulam and Saria, 2019], and learning data augmentation techniques [Lee et al., 2020].

When the training objective is strongly convex (e.g., as in logistic regression with L_2 regularization), influence functions are expected to align well with leave-one-out (LOO) or leave-k-out retraining [Koh and Liang, 2017, Koh et al., 2019, Izzo et al., 2021]. However, Basu et al. [2020a] showed that influence functions in neural networks often do not accurately predict the effect of retraining the model and concluded that influence estimates are often "fragile" and "erroneous". Because of the poor match between influence estimates and LOO retraining, influence function methods are often evaluated with alternative metrics such as the detection rate of maliciously corrupted examples using influence scores [Khanna et al., 2019, Koh and Liang, 2017, Schioppa et al., 2021, K and Søgaard,

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

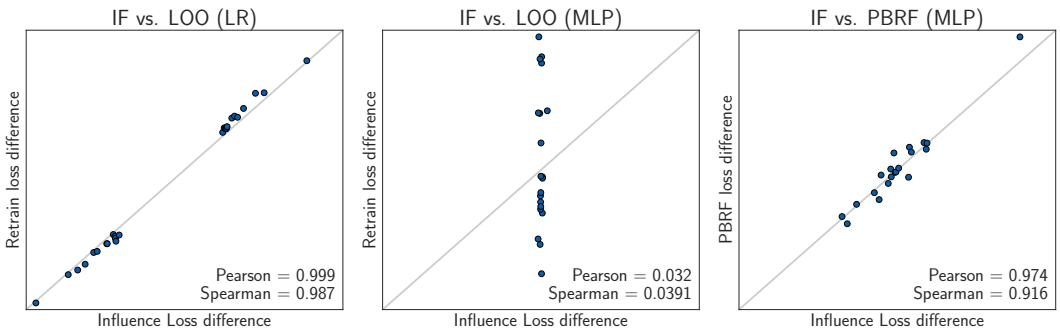


Figure 1: Comparison of test loss differences computed by influence function (IF), leave-one-out (LOO) retraining, and our proximal Bregman response function (PBRF). Each point corresponds to the individual effect of removing one training example. Influence estimates align well with true retraining for (**left**) logistic regression (LR) but poorly for (**middle**) multilayer perceptrons (MLP). While influence functions in neural networks do not accurately predict the effect of retraining the model, they are still a good approximation to (**right**) the PBRF.

2021]. However, these indirect signals make it difficult to develop algorithmic improvements to influence function estimation. If one is interested in improving certain aspects of influence function estimation, such as the linear system solver, it would be preferable to have a well-defined quantity that influence function estimators are approximating so that algorithmic choices could be directly evaluated based on the accuracy of their estimates.

In this work, we investigate the source of the discrepancy between influence functions and LOO retraining in neural networks. We decompose the discrepancy into five components: (1) the difference between cold-start and warm-start response functions (a concept elaborated on below), (2) an implicit proximity regularizer, (3) influence estimation on non-converged parameters, (4) linearization, and (5) approximate solution of a linear system. This decomposition was chosen to capture all gaps and errors caused by approximations and assumptions made in applying influence functions to neural networks. We empirically evaluate the contributions of each component on binary classification, regression, image reconstruction, image classification, and language modeling tasks and show that, across all tasks, components (1–3) are most responsible for the discrepancy between influence functions and LOO retraining. We further investigate how the contribution of each component changes in response to the change in network width and depth, weight decay, training time, damping, and the number of data points being removed.

Moreover, we show that while influence functions for neural networks are often a poor match to LOO retraining, they are a much better match to what we term the *proximal Bregman response function (PBRF)*. Intuitively, the PBRF approximates the effect of removing a data point while trying to keep the predictions consistent with those of the (partially) trained model. From this perspective, we reframe misalignment components (1–3) as simply reflecting the difference between LOO retraining and the PBRF. The gap between the influence function estimate and the PRBF only comes from sources (4) and (5), which we found empirically to be at least an order of magnitude smaller for most neural networks. As a result, on a wide variety of tasks, influence functions closely align with the PBRF while failing to approximate the effect of retraining the model, as shown in Figure 1.

The PBRF can be used for many of the same use cases that have motivated influence functions, such as finding influential or mislabeled examples [Schioppa et al., 2021] and carrying out data poisoning attacks [Koh and Liang, 2017, Koh et al., 2022], and can therefore be considered an alternative to LOO retraining as a gold standard for evaluating influence functions. Hence, we conclude that influence functions applied to neural networks are not inherently "fragile" as is often believed [Basu et al., 2020a], but instead can be seen as giving accurate answers to a different question than is normally assumed.

2 Related Work

Instance-based interpretability methods are a class of techniques that explain a model's predictions in terms of the examples on which the model was trained. Methods of this type include TracIn [Pruthi et al., 2020], Representer Point Selection [Yeh et al., 2018], Grad-Cos and Grad-Dot [Charpiat et al., 2019, Hanawa et al., 2021], MMD-critic [Kim et al., 2016], unconditional counterfactual

explanations [Wachter et al., 2018], and of central focus in this paper, influence functions. Since its adoption in machine learning by Koh and Liang [2017], multiple extensions and improvements upon influence functions have also been proposed, such as variants that use Fisher kernels [Khanna et al., 2019], higher-order approximations [Basu et al., 2020b], tricks for faster and scalable inference [Guo et al., 2021, Schioppa et al., 2021], group influence formulations [Koh et al., 2019, Basu et al., 2020b], and relative local weighting [Barshan et al., 2020]. However, many of these methods rely on the same strong assumptions made in the original influence function derivation that the objective needs to be strongly convex and influence functions must be computed on the optimal parameters.

In general, influence functions are assumed to approximate the effects of leave-one-out (LOO) retraining from scratch, the parameters of the network that are trained without a data point of interest. Hence, measuring the quality of influence functions is often performed by analyzing the correlation between LOO retraining and influence function estimations [Koh and Liang, 2017, Basu et al., 2020a,b, Yang and Chaudhuri, 2022]. However, recent empirical analyses have demonstrated the fragility of influence functions and a fundamental misalignment between their assumed and actual effects [Basu et al., 2020a, Ghorbani et al., 2019, K and Søgaard, 2021]. For example, Basu et al. [2020a] argued that the accuracy of influence functions in deep networks is highly sensitive to network width and depth, weight decay strength, inverse-Hessian vector product estimation methodology, and test query point by measuring the alignment between influence functions and LOO retraining. Because of the inherent misalignment between influence estimations and LOO retraining in neural networks, many works often evaluate the accuracy of the influence functions on an alternative metric, such as the recovery rate of maliciously mislabelled or poisoned data using influence functions [Khanna et al., 2019, Koh and Liang, 2017, Schioppa et al., 2021, K and Søgaard, 2021]. In this work, instead of interpreting the misalignment between influence functions and LOO retraining as a failure, we claim that it simply reflects that influence functions answer a different question than is typically assumed.

3 Background

Consider a prediction task from an input space $\mathcal{X}$ to a target space $\mathcal{T}$ where we are given a finite training dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{N}$. Given a data point $\mathbf{z} = (\mathbf{x}, \mathbf{t})$, let $\mathbf{y} = f(\boldsymbol{\theta}, \mathbf{x})$ be the prediction of the network parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$ and $\mathcal{L}(\mathbf{y}, \mathbf{t})$ be the loss (e.g., squared error or cross-entropy). We aim to solve the following optimization problem:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{J}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}^{(i)}), \mathbf{t}^{(i)}), \tag{1}$$

where $\mathcal{J}(\cdot)$ is the cost function. If the regularization (e.g., L_2 regularization) is imposed in the cost function, we fold the regularization terms into the loss function. We summarize the notation used in this paper in Appendix A.

3.1 Downweighting a Training Example

The training objective in Eqn. 1 aims to find the parameters that minimize the average loss on all training examples. Herein, we are interested in studying the change in optimal model parameters when a particular training example $\mathbf{z} = (\mathbf{x}, \mathbf{t}) \in \mathcal{D}_{\text{train}}$ is removed from the training dataset, or more generally, when the data point $\mathbf{z}$ is downweighted by an amount $\epsilon \in \mathbb{R}$. Formally, this corresponds to minimizing the following downweighted objective:

$$\boldsymbol{\theta}^\star_{-\mathbf{z},\epsilon} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{Q}_{-\mathbf{z}}(\boldsymbol{\theta}, \epsilon) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{J}(\boldsymbol{\theta}) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon. \tag{2}$$

When $\epsilon = 1/N$, the downweighted objective reduces to the cost over the dataset with the example $\mathbf{z}$ removed, up to a constant factor. To see how the optimum of the downweighted objective responds to changes in the downweighting factor ϵ, we define the *response function* $r^\star_{-\mathbf{z}} \colon \mathbb{R} \to \mathbb{R}^d$ by:

$$r^\star_{-\mathbf{z}}(\epsilon) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{Q}_{-\mathbf{z}}(\boldsymbol{\theta}, \epsilon), \tag{3}$$

where we assume that the downweighted objective is strongly convex and hence the solution to the downweighted objective is unique given some factor ϵ. Under these assumptions, note that $r^\star_{-\mathbf{z}}(0) = \boldsymbol{\theta}^\star$ and the response function is differentiable at 0 by the Implicit Function Theorem [Krantz

and Parks, 2002, Griewank and Walther, 2008]. Influence functions approximate the response function by performing a first-order Taylor expansion around $\epsilon_0 = 0$:

$$r^\star_{-\mathbf{z},\text{lin}}(\epsilon) = r^\star_{-\mathbf{z}}(\epsilon_0) + \left.\frac{\mathrm{d}r^\star_{-\mathbf{z}}}{\mathrm{d}\epsilon}\right|_{\epsilon=\epsilon_0}(\epsilon - \epsilon_0) = \boldsymbol{\theta}^\star + (\nabla^2_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}^\star))^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}), \mathbf{t})\epsilon. \tag{4}$$

We refers readers to Van der Vaart [2000] and Appendix B for a detailed derivation. The optimal parameters trained without $\mathbf{z}$ can then be approximated by plugging in $\epsilon = {}^1\!/_N$ to Eqn. 4.

Influence functions can further approximate the loss of a particular test point $\mathbf{z}_{\text{test}} = (\mathbf{x}_{\text{test}}, \mathbf{t}_{\text{test}})$ when a data point $\mathbf{z}$ is eliminated from the training set using the chain rule [Koh and Liang, 2017]:

$$\begin{aligned}
&\mathcal{L}(f(r^\star_{-\mathbf{z},\text{lin}}({}^1\!/_N), \mathbf{x}_{\text{test}}), \mathbf{t}_{\text{test}}) \\
&\quad\approx \mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}_{\text{test}}), \mathbf{t}_{\text{test}}) + \frac{1}{N}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}_{\text{test}}), \mathbf{t}_{\text{test}})^\top \left.\frac{\mathrm{d}r^\star_{\mathbf{z}}}{\mathrm{d}\epsilon}\right|_{\epsilon=0} \\
&\quad= \mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}_{\text{test}}), \mathbf{t}_{\text{test}}) + \frac{1}{N}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}_{\text{test}}), \mathbf{t}_{\text{test}})^\top (\nabla^2_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}^\star))^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}), \mathbf{t}).
\end{aligned} \tag{5}$$

3.2 Influence Function Estimation in Neural Networks

Influence functions face two main challenges when deployed on neural networks. First, the influence estimation (shown in Eqn. 4) requires computing an inverse Hessian-vector product (`iHVP`). Unfortunately, storing and inverting the Hessian requires $O(d^3)$ operations and is infeasible to compute for modern neural networks. Instead, Koh and Liang [2017] tractably approximate the `iHVP` using truncated non-linear conjugate gradient (`CG`) [Martens et al., 2010] or the `LiSSA` algorithm [Agarwal et al., 2016]. Both approaches avoid explicit computation of the Hessian inverse (see Appendix G for details) and only require $O(Nd)$ operations to approximate the influence function.

Second, the derivation of influence functions assumes a strongly convex objective, which is often not satisfied for neural networks. The Hessian may be singular, especially when the parameters have not fully converged, due to non-positive eigenvalues. To enforce positive-definiteness of the Hessian, Koh and Liang [2017] add a damping term in the `iHVP`. Teso et al. [2021] further approximate the Hessian with the Fisher information matrix (which is equivalent to the Gauss-Newton Hessian [Martens, 2014] for commonly used loss functions such as cross-entropy) as follows:

$$r^\star_{-\mathbf{z},\text{damp},\text{lin}}(\epsilon) \approx \boldsymbol{\theta}^\star + (\mathbf{J}^\top_{\mathbf{y}\boldsymbol{\theta}^\star}\mathbf{H}_{\mathbf{y}^\star}\mathbf{J}_{\mathbf{y}\boldsymbol{\theta}^\star} + \lambda\mathbf{I})^{-1}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^\star, \mathbf{x}), \mathbf{t})\epsilon, \tag{6}$$

where $\mathbf{J}_{\mathbf{y}\boldsymbol{\theta}^\star}$ is the parameter-output Jacobian and $\mathbf{H}_{\mathbf{y}^\star}$ is the Hessian of the cost with respect to the network outputs both evaluated on the optimal parameters $\boldsymbol{\theta}^\star$. Here, $\mathbf{G}^\star = \mathbf{J}^\top_{\mathbf{y}\boldsymbol{\theta}^\star}\mathbf{H}_{\mathbf{y}^\star}\mathbf{J}_{\mathbf{y}\boldsymbol{\theta}^\star}$ is the Gauss-Newton Hessian (GNH) and $\lambda > 0$ is a damping term to ensure the invertibility of GNH. Unlike the Hessian, the GNH is guaranteed to be positive semidefinite as long as the loss function is convex as a function of the network outputs [Martens et al., 2010].

4 Understanding the Discrepancy between Influence Function and LOO Retraining in Neural Networks

In this section, we investigate several factors responsible for the misalignment between influence functions and LOO retraining. Specifically, we decompose the misalignment into five separate terms: (1) the warm-start gap, (2) the damping gap, (3) the non-convergence gap, (4) the linearization error, and (5) the solver error. This decomposition captures all approximations and assumption violations when deploying influence functions in neural networks. By summing the parameter (or outputs) differences introduced by each term we can bound the parameter (or outputs) difference between LOO retraining and influence estimates. We use the term "gap" rather than "error" for the first three terms to emphasize that they reflect differences between solutions to different influence-related questions, rather than actual errors.

For all models we investigate, we find that the first three sources dominate the misalignment, indicating that the misalignment reflects not algorithmic errors but rather the fact that influence function estimators are answering a different question from what is normally assumed. All proximal objectives are summarized in Table 1 and we provide the derivations in Appendix B.

4.1 Warm-Start Gap: Non-Strongly Convex Training Objective

By taking a first-order Taylor approximation of the response function at $\epsilon_0 = 0$ (Eqn. 4), influence functions approximate the effect of removing a data point $\mathbf{z}$ at a local neighborhood of the optimum $\boldsymbol{\theta}^\star$. Hence, influence approximation has a more natural connection to the retraining scheme that initializes the network at the current optimum $\boldsymbol{\theta}^\star$ (*warm-start retraining*) than the scheme that initializes the network randomly (*cold-start retraining*). The warm-start optimum is equivalent to the cold-start optimum when the objective is strongly convex (where the solution to the response function is unique), making the influence estimation close to the LOO retraining on logistic regression with L_2 regularization.

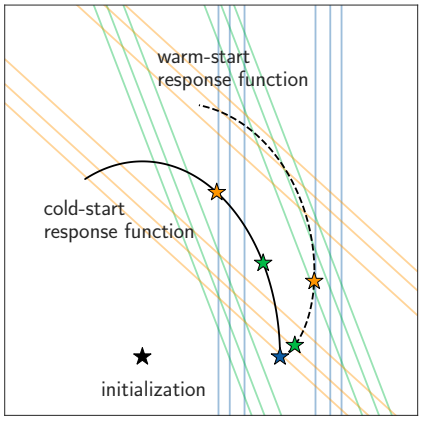


Figure 2: Cold-start (initialized from black star) and warm-start (initialized from blue star) response functions for quadratic cost function. Each contour represents the cost function at some ϵ. Because gradient descent converges to a minimum-norm solution, the warm-start and cold-start optima are not equivalent.

However, the equivalence between warm-start and cold-start optima is not typically guaranteed in neural networks [Vicol et al., 2022a]. Particularly, in the over-parametrized regime ($N < d$), neural networks exhibit multiple global optima, and their converged solutions depend highly on the specifics of the optimization dynamics [Lee et al., 2019, Arora et al., 2019, Bartlett et al., 2020, Amari et al., 2020]. For quadratic cost functions, gradient descent with initialization $\boldsymbol{\theta}^0$ converges to the optimum that achieves the minimum L_2 distance from $\boldsymbol{\theta}^0$ [Hastie et al., 2022]. This phenomenon of the converged parameters being dependent on the initialization hinders influence functions from accurately predicting the effect of retraining the model from scratch as shown in Figure 2. We denote the discrepancy between cold-start and warm-start optima as **warm-start gap**.

4.2 Proximity Gap: Addition of Damping Term in `iHVP`

In practical settings, we often impose a damping term (Eqn. 6) in influence approximations to ensure that the cost Hessian is positive-definite and hence invertible. As adding a damping term in influence estimations is equivalent to adding L_2 regularization to the cost function [Martens et al., 2010], when damping is used, influence functions can be seen as linearizing the following *proximal response function* at $\epsilon_0 = 0$:

$$r^\star_{-\mathbf{z},\text{damp}}(\epsilon) = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \mathcal{Q}_{-\mathbf{z}}(\boldsymbol{\theta}, \epsilon) + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^\star\|^2. \tag{7}$$

See Appendix B.2 for the derivation. Note that $\lambda > 0$ is a damping strength and our use of "proximal" is based on the notion of proximal equilibria [Farnia and Ozdaglar, 2020]. Intuitively, the proximal objective in Eqn. 7 not only minimizes the downweighted objective but also encourages the parameters to stay close to the optimal parameters at $\epsilon_0 = 0$. Hence, when the damping term is used in the `iHVP`, influence functions aim at approximating the warm-start retraining scheme with a proximity term that penalizes the L_2 distance between the new estimate and the optimal parameters. We call the discrepancy between the warm-start and proximal warm-start optima the **proximity gap**.

Interestingly, past works have observed that for quadratic cost functions, early stopping has a similar effect to L_2 regularization [Vicol et al., 2022a, Ali et al., 2019]. Therefore, the proximal response function can be thought of as capturing how gradient descent will respond to a dataset perturbation if it takes only a limited number of steps starting from the warm-start solution.

4.3 Non-Convergence Gap: Influence Estimation on Non-Converged Parameters

Thus far, our analysis has assumed that influence functions are computed on fully converged parameters $\boldsymbol{\theta}^\star$ at which the gradient of the cost is $\mathbf{0}$. However, in neural network training, we often terminate the optimization procedure before reaching the exact optimum due to several reasons, including having limited computational resources or to avoid overfitting [Bengio, 2012]. In such situations, much of the change in the parameters from LOO retraining simply reflects the effect of training for

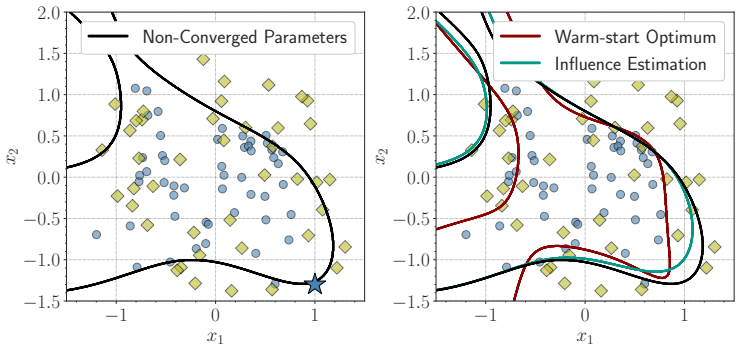


Figure 3: Decision boundaries for a partially trained binary classifier. We consider removing a data point located at right-bottom corner denoted as $\star$. While the influence estimation makes a local change on the data point of interest, the (warm-start) LOO retraining globally updates the parameters to better fit other data points (a nuisance from the perspective of understanding influence).

Error	Objective	Init
Cold-start	$\mathcal{J}(\boldsymbol{\theta}) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon$	$\boldsymbol{\theta}^0$
+ Warm-start	$\mathcal{J}(\boldsymbol{\theta}) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon$	$\boldsymbol{\theta}^s$
+ Proximity	$\mathcal{J}(\boldsymbol{\theta}) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^s\|^2$	$\boldsymbol{\theta}^s$
+ Non-Convergence	$\frac{1}{N}\sum_{i=1}^N D_{\mathcal{L}^{(i)}}(f(\boldsymbol{\theta}, \mathbf{x}^{(i)}), f(\boldsymbol{\theta}^s, \mathbf{x}^{(i)})) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^s\|^2$	$\boldsymbol{\theta}^s$
+ Linearization	$\frac{1}{N}\sum_{i=1}^N D_{\mathcal{L}^{(i)}_{\text{quad}}}(f_{\text{lin}}(\boldsymbol{\theta}, \mathbf{x}^{(i)}), f(\boldsymbol{\theta}^s, \mathbf{x}^{(i)})) - \nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^s, \mathbf{x}), \mathbf{t})^\top \boldsymbol{\theta}\epsilon + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^s\|^2$	$\boldsymbol{\theta}^s$

Table 1: Summary of proximal objectives that influence functions aim to approximate when the network is non-strongly convex, a damping term is used, and influence functions are computed on non-converged parameters. The final linearization reflects the second-order approximation that influence functions utilize.

longer, rather than the effect of removing a training example, as illustrated in Figure 3. What we desire from influence functions is to understand the effect of removing the training example; the effect of extended training is simply a nuisance. Therefore, to the extent that this factor contributes to the misalignment between influence functions and LOO retraining, influence functions are arguably *more useful* than LOO retraining.

Since training the network to convergence may be impractical or undesirable, we instead modify the response function by replacing the original training objective with a similar one *for which the (possibly non-converged) final parameters $\boldsymbol{\theta}^s$ are optimal*. Here, we assume the loss function $\mathcal{L}(\cdot, \cdot)$ is convex as a function of the network outputs; this is true for commonly used loss functions such as squared error or cross-entropy. We replace the training loss with a term that penalizes mismatch to the predictions made by $\boldsymbol{\theta}^s$ (hence implying that $\boldsymbol{\theta}^s$ is optimal). Our *proximal Bregman response function (PBRF)* is defined as follows:

$$r^b_{-\mathbf{z},\text{damp}}(\epsilon) = \operatorname*{arg\,min}_{\boldsymbol{\theta} \in \mathbb{R}^d} \frac{1}{N}\sum_{i=1}^N D_{\mathcal{L}^{(i)}}(f(\boldsymbol{\theta}, \mathbf{x}^{(i)}), f(\boldsymbol{\theta}^s, \mathbf{x}^{(i)})) - \mathcal{L}(f(\boldsymbol{\theta}, \mathbf{x}), \mathbf{t})\epsilon + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^s\|^2, \tag{8}$$

where $D_{\mathcal{L}^{(i)}}(\cdot, \cdot)$ is the Bregman divergence defined as:

$$D_{\mathcal{L}^{(i)}}(\mathbf{y}, \mathbf{y}^s) = \mathcal{L}(\mathbf{y}, \mathbf{t}^{(i)}) - \mathcal{L}(\mathbf{y}^s, \mathbf{t}^{(i)}) - \nabla_{\mathbf{y}}\mathcal{L}(\mathbf{y}^s, \mathbf{t}^{(i)})^\top(\mathbf{y} - \mathbf{y}^s). \tag{9}$$

The PBRF defined in Eqn. 8 is composed of three terms. The first term measures the functional discrepancy between the current estimate and the parameters $\boldsymbol{\theta}^s$ in Bregman divergence, and its role is to prevent the new estimate from drastically altering the predictions on the training dataset. One way of understanding this term in the cases of squared error or cross-entropy losses is that it is equivalent to the training error on a dataset where the original training labels are replaced with soft targets obtained from the predictions made by $\boldsymbol{\theta}^s$. The second term is the negative loss on the data point $\mathbf{z} = (\mathbf{x}, \mathbf{t})$, which aims to respond to the deletion of a training example. The final term is simply the proximity term described before. In Appendix B.3, we further show that the influence

function on non-converged parameters is equivalent to the first-order approximation of PBRF instead of the first-order approximation of proximal response function for linear models.

Rather than computing the LOO retrained parameters by performing K additional optimization steps under the original training objective, we can instead perform K optimization steps under the proximal Bregman objective. The difference between the resulting parameter vectors is what we call the **non-convergence gap**.

4.4 Linearization Error: A First-order Taylor Approximation of the Response Function

The key idea behind influence functions is the linearization of the response function. To simulate the local approximations made in influence functions, we define the linearized PBRF as:

$$r^{b}_{-\mathbf{z},\text{damp,lin}}(\epsilon) = \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} D_{\mathcal{L}^{(i)}_{\text{quad}}}(f_{\text{lin}}(\boldsymbol{\theta}, \mathbf{x}^{(i)}), f(\boldsymbol{\theta}^s, \mathbf{x}^{(i)})) \\ - \nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{\theta}^s, \mathbf{x}), \mathbf{t})^\top \boldsymbol{\theta}\epsilon + \frac{\lambda}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}^s\|^2, \tag{10}$$

where $\mathcal{L}_{\text{quad}}(\cdot,\cdot)$ is the second-order expansion of the loss around $\mathbf{y}^s$ and $f_{\text{lin}}(\cdot,\cdot)$ is the linearization of the network outputs with respect to the parameters. The optimal solution to the linearized PBRF is equivalent to the influence estimation at the parameters $\boldsymbol{\theta}^s$ with the GNH approximation and a damping term λ (see Appendix B.4 for the derivation).

As the linearized PBRF relies on several local approximations, the linearization error increases when the downweighting factor magnitude $|\epsilon|$ is large or the PBRF is highly non-linear. We refer to the discrepancy between the PBRF and linearized PBRF as the **linearization error**.

4.5 Solver Error: A Crude Approximation of `iHVP`

As the precise computation of the `iHVP` is computationally infeasible, in practice, we use truncated `CG` or `LiSSA` to efficiently approximate influence functions [Koh and Liang, 2017]. Unfortunately, these efficient linear solvers introduce additional error by crudely approximating the `iHVP`. Moreover, different linear solvers can introduce specific biases in the influence estimation. For example, Vicol et al. [2022b] show that the truncated `LiSSA` algorithm implicitly adds an additional damping term in the `iHVP`. We use **solver error** to refer to the difference between the linearized PBRF and the influence estimation computed by a linear solver.

Interestingly, Koh and Liang [2017] reported that the `LiSSA` algorithm gave more accurate results than `CG`. We have determined that this difference resulted not from any inherent algorithmic advantage to `LiSSA`, but rather from the fact that the software used different damping strengths for the two algorithms, thereby resulting in different weightings of the proximity term in the proximal response function.

5 PBRF: The Question Influence Functions are Really Answering

The PBRF (Eqn. 8) approximates the effect of removing a data point while trying to keep the predictions consistent with those of the (partially) trained model. Since the discrepancy between the PBRF and influence function estimates is only due to the linearization and solver errors, the PBRF can be thought of as better representing the question that influence functions are trying to answer. Reframing influence functions in this way means that the PBRF can be regarded as a gold-standard ground truth for evaluating methods for influence function approximation. Existing analyses of influence functions [Basu et al., 2020a] rely on generating LOO retraining ground truth estimates by imposing strong L_2 regularization or training till convergence without early stopping. However, these conditions do not accurately reflect the typical way neural networks are trained in practice. In contrast, our PBRF formulation does not require the addition of any regularizers or modified training regimes and can be easily optimized.

In addition, although the PBRF may not necessarily align with LOO retraining due to the warm-start, proximity, and non-convergence gaps, the motivating use cases for influence functions typically do not rely on exact LOO retraining. This means that the PBRF can be used in place of LOO retraining for

many tasks such as identifying influential or mislabelled examples, as demonstrated in Appendix D.3. In these cases, influence functions are still useful since they provide an efficient way of approximating PBRF estimates.

6 Experiments

Our experiments investigate the following questions: (1) What factors discussed in Section 4 contribute most to the misalignment between influence functions and LOO retraining? (2) While influence functions fail to approximate the effect of retraining, do they accurately approximate the PBRF? (3) How do changes in weight decay, damping, the number of total epochs, and the number of removed training examples affect each source of misalignment?

In all experiments, we first train the base network with the entire dataset to obtain the parameters $\boldsymbol{\theta}^s$. We repeat the training procedure 20 times with a different random training example deleted. The cold-start retraining begins from the same initialization used to train $\boldsymbol{\theta}^s$. All proximal objectives are trained with initialization $\boldsymbol{\theta}^s$ for 50% of the epochs used to train the base network. Lastly, we use the `LiSSA` algorithm with GNH approximation to compute influence functions.

Since we are primarily interested in the effect of deleting a data point on model's predictions, we measure the discrepancy of each gap and error using the average L_2 distance between networks' outputs $\mathbb{E}_{(\mathbf{x},\cdot)\sim\mathcal{D}_{\text{train}}}[\|f(\boldsymbol{\theta},\mathbf{x}) - f(\boldsymbol{\theta}',\mathbf{x})\|]$ on the training dataset. We provide the full experimental set-up and additional experiments in Appendix C and D, respectively.

6.1 Influence Misalignment Decomposition

We first applied our decomposition to various models trained on a broad range of tasks covering binary classification, regression, image reconstruction, image classification, and language modeling. The summary of our results is provided in Figure 4 and Table 5 (Appendix E). Across all tasks, we found that the first three sources dominate the misalignment, indicating influence function estimators are answering a different question from what is normally assumed. Small linearization and solver errors indicate that influence functions accurately answer the modified question (PBRF).

Logistic Regression. We analyzed the logistic regression (LR) model trained on the Cancer and Diabetes classification datasets from the UCI collection [Dua and Graff, 2017]. We trained the model using `L-BFGS` [Liu and Nocedal, 1989] with L_2 regularization of 0.01 and damping term of $\lambda = 0.001$. As the training objective is strongly convex and the base model parameters were trained till convergence, in Table 5, we observed that each source of misalignment is significantly low. Hence, in the case of logistic regression with L_2 regularization, influence functions accurately capture the effect of retraining the model without a data point.

Multilayer Perceptron. Next, we applied our analysis to the 2-hidden layer Multilayer Perceptron (MLP) with ReLU activations. We conducted the experiments in two settings: (1) regression on the Concrete and Energy datasets from the UCI collection and (2) image classification on 10% of the MNIST [Deng, 2012] and FashionMNIST [Xiao et al., 2017] datasets, following the set-up from Koh and Liang [2017] and Basu et al. [2020a]. We trained the networks for 1000 epochs using stochastic gradient descent (SGD) with a batch size of 128 and set a damping strength of $\lambda = 0.001$.

As opposed to linear models, MLPs violate the assumptions in the influence derivation and we observed an increase in gaps and errors on all five factors. We observed that warm-start, proximity, and the non-convergence gaps contribute more to the misalignment than linearization and solver errors. The average network's predictions for PBRF were similar to that computed by the `LiSSA` algorithm, demonstrating that influence functions are still a good approximation to PBRF.

Autoencoder. Next, we applied our framework to an 8-layer autoencoder (AE) on the full MNIST dataset. We followed the experimental set-up from Martens and Grosse [2015], where the encoder and decoder each consist of 4 fully-connected layers with sigmoid activation functions. We trained the network for 1000 epochs using SGD with momentum. We set the batch size to 1024, used L_2 regularization of 10^{-5} with a damping factor of $\lambda = 0.001$. In accordance with the findings from our MLP experiments, the warm-start, proximity, and non-convergence gaps were more significant than the linearization and solver errors, and influence functions accurately predicted the PBRF.

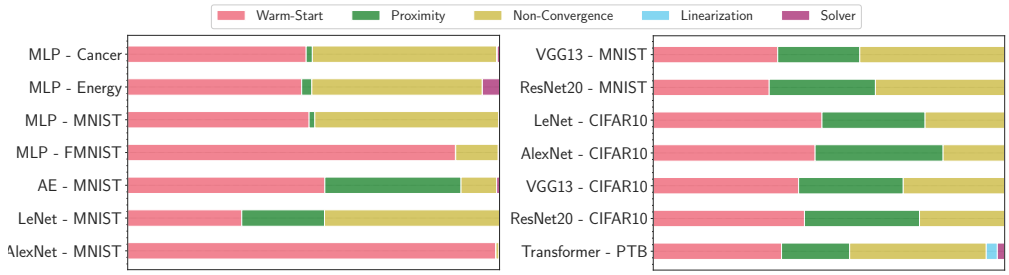


Figure 4: Decomposition of the discrepancy between influence functions and LOO retraining into (1) warm-start gap, (2) proximal gap, (3) non-convergence gap, (4) linearization error, and (5) solver error for each model and dataset. The size of each component is measured by the L_2 distance between the networks' outputs on the training dataset.

Convolutional Neural Networks. To investigate the source of discrepancy on larger-scale networks, we trained a set of convolutional neural networks of increasing complexity and size. Namely, LeNet [Lecun et al., 1998], AlexNet [Krizhevsky et al., 2012], VGG13 Simonyan and Zisserman [2014], and ResNet-20 [He et al., 2015] were trained on 10% of the MNIST dataset and the full CIFAR10 [Krizhevsky, 2009] dataset. We trained the base network for 200 epochs on both datasets with a batch size of 128. For MNIST, we kept the learning rate fixed throughout training, while for CIFAR10, we decayed the learning rate by a factor of 5 at epochs 60, 120, and 160, following Zagoruyko and Komodakis [2016]. We used L_2 regularization with strength $5 \cdot 10^{-4}$ and a damping factor of $\lambda = 0.001$. Consistent with the findings from our MLP and autoencoder experiments, the first three gaps were more significant than linearization and solver errors.

Model	Cold-Start		Warm-Start		PBRF	
	P	S	P	S	P	S
MLP	-0.55	0.01	0.22	0.35	**0.98**	**0.99**
LeNet	-0.19	0.12	0.32	0.25	**0.93**	**0.52**
AlexNet	-0.16	-0.08	0.51	0.58	**0.99**	**0.99**
VGG13	0.45	-0.07	-0.28	-0.51	**0.98**	**0.77**
ResNet-20	0.09	-0.06	0.02	0.09	**0.81**	**0.76**

Table 2: Comparison of test loss differences computed by influence function, cold-start retraining, warm-start retraining, and PBRF on MNIST dataset. We show Pearson (P) and Spearman rank-order (S) correlation when compared to influence estimates.

We further compared influence functions' approximations on the difference in test loss when a random training data point is removed with the value obtained from cold-start retraining, warm-start retraining, and PBRF in Table 2. We used both Pearson [Sedgwick, 2012] and Spearman rank-order correlation [Spearman, 1961] to measure the alignment. While the test loss predicted by influence functions does not align well with the values obtained by cold-start and warm-start retraining schemes, they show high correlations when compared to the estimates given by PBRF.

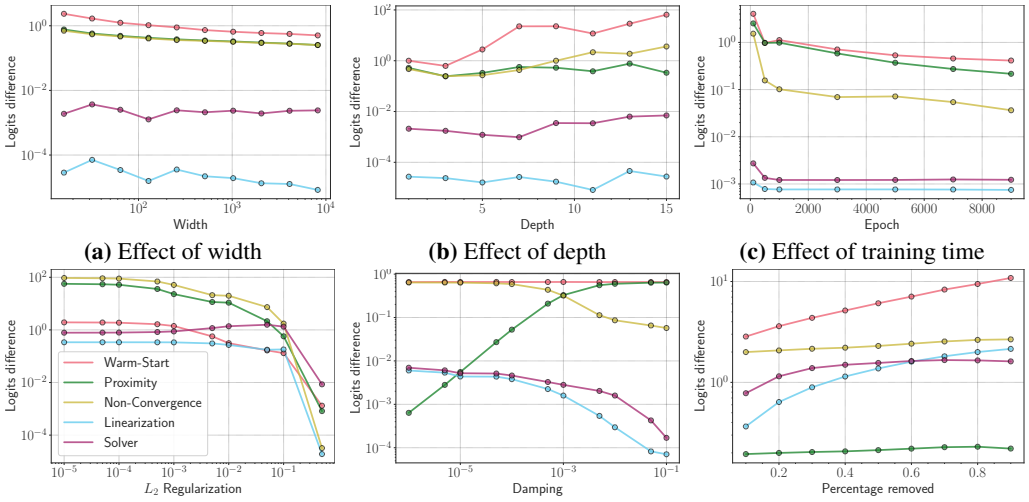
Transformer. Finally, we trained 2-layer Transformer language models on the Penn Treebank (PTB) [Marcus et al., 1993] dataset. We set the number of hidden dimensions to 256 and the number of attention heads to 2. As we observed that model overfits after a few epochs of training, we trained the base network for 10 epochs using Adam. Notably, we observed that the non-convergence gap had the most considerable contribution to the discrepancy between influence functions and LOO retraining. Consistent with our previous findings, the first tree gaps had more impact on the discrepancy compared to linearization and solver errors.

6.2 Factors in Influence Misalignment

We further analyzed how the contribution of each component changes in response to changes in network width and depth, training time, weight decay, damping, and the percentage of data removed. We used an MLP trained on 10% of the MNIST dataset and summarized results in Figure 5.

Width and Depth. As we increase the width of the network, we observe a decrease in the linearization error. This is consistent with previous observations that networks behave more linearly as the width is increased [Lee et al., 2019]. In contrast to the findings from Basu et al. [2020a], we did not observe a strong relationship between the contribution of the components and the depth of the network.

Training Time. Unsurprisingly, as we increase the number of training epochs, we observe a decrease in the non-convergence gap. We hypothesize that, as we increase the training epoch, the cost gradient reaches $\mathbf{0}$, resulting in better alignment between the proximal response function and PBRF.



(a) Effect of width **(b)** Effect of depth **(c)** Effect of training time

(d) Effect of L_2 regularization **(e)** Effect of damping strength **(f)** Effect of % data removed

Figure 5: Ablations on how various factors affect the contribution of the gaps and errors to the discrepancy between influence approximation and LOO retraining.

Weight Decay. The weight decay allows the training objective to be better conditioned. Consequently, as weight decay increases, the training objective may act more as a strictly convex objective, resulting in a decrease in overall discrepancy for all components. Basu et al. [2020a] also found that the alignment between influence functions and LOO retraining increases as weight decay increases.

Damping. A higher damping term makes linear systems better conditioned, allowing solvers to find accurate solutions in fewer iterations [Demmel, 1997], thereby reducing the solver error. Furthermore, the higher proximity term keeps the parameters close to $\boldsymbol{\theta}^s$, reducing the linearization error. On the other hand, increasing the effective proximity penalty directly increases the proximity gap.

Percentage of Training Examples Removed. As we remove more training examples from the dataset the PBRF becomes more non-linear and we observe a sharp increase in the linearization error. The cost landscape is also more likely to change as we remove more training examples, and we observe a corresponding increase in the warm-start gap.

7 Conclusion

In this paper, we investigate the sources of the discrepancy between influence functions and LOO retraining in neural networks. We decompose this difference into five distinct components: the warm-start gap, proximity gap, non-convergence gap, linearization error, and solver error. We empirically evaluate the contributions of each of these components on a wide variety of architectures and datasets and investigate how they change with factors such as network size and regularization. Our results show that the first three components are most responsible for the discrepancy between influence functions and LOO retraining. We further introduce the proximal Bregman response function (PBRF) to better capture the behavior of influence functions in neural networks. Compared to LOO retraining, the PBRF is more easily calculated and correlates better with influence functions, meaning it is an attractive alternative gold standard for evaluating influence functions. Although the PBRF may not necessarily align with LOO retraining, it can still be applied in many of the motivating use cases for influence functions. We conclude that influence functions in neural networks are not necessarily "fragile", but instead are giving accurate answers to a different question than is normally assumed.

Acknowledgements

We would like to thank Pang Wei Koh for the helpful discussions. Resources used in this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute (`www.vectorinstitute.ai/partners`).

References

N. Agarwal, B. Bullins, and E. Hazan. Second-order stochastic optimization in linear time. *stat*, 1050:15, 2016.

A. Ali, J. Z. Kolter, and R. J. Tibshirani. A continuous-time view of early stopping for least squares regression. In *The 22nd international conference on artificial intelligence and statistics*, pages 1370–1378. PMLR, 2019.

S.-i. Amari, J. Ba, R. Grosse, X. Li, A. Nitanda, T. Suzuki, D. Wu, and J. Xu. When does preconditioning help or hurt generalization? *arXiv preprint arXiv:2006.10732*, 2020.

S. Arora, S. Du, W. Hu, Z. Li, and R. Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *International Conference on Machine Learning*, pages 322–332. PMLR, 2019.

E. Barshan, M.-E. Brunet, and G. K. Dziugaite. RelatIF: Identifying explanatory training samples via relative influence. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1899–1909. PMLR, 26–28 Aug 2020. URL `https://proceedings.mlr.press/v108/barshan20a.html`.

P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

S. Basu, P. Pope, and S. Feizi. Influence functions in deep learning are fragile. *arXiv preprint arXiv:2006.14651*, 2020a.

S. Basu, X. You, and S. Feizi. On second-order group influence functions for black-box predictions. In *International Conference on Machine Learning*, pages 715–724. PMLR, 2020b.

Y. Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

M.-E. Brunet, C. Alkalay-Houlihan, A. Anderson, and R. Zemel. Understanding the origins of bias in word embeddings. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 803–811. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/brunet19a.html`.

G. Charpiat, N. Girard, L. Felardos, and Y. Tarabalka. Input similarity from the neural network perspective. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/c61f571dbd2fb949d3fe5ae1608dd48b-Paper.pdf`.

R. D. Cook. Influential observations in linear regression. *Journal of the American Statistical Association*, 74(365):169–174, 1979.

J. W. Demmel. *Applied numerical linear algebra*. SIAM, 1997.

L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

F. Farnia and A. Ozdaglar. Do gans always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR, 2020.

A. Ghorbani, A. Abid, and J. Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688, Jul. 2019. doi: 10.1609/aaai.v33i01.33013681. URL https://ojs.aaai.org/index.php/AAAI/article/view/4252.

A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

H. Guo, N. Rajani, P. Hase, M. Bansal, and C. Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.808. URL https://aclanthology.org/2021.emnlp-main.808.

F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

X. Han, B. C. Wallace, and Y. Tsvetkov. Explaining black box predictions and unveiling data artifacts through influence functions. In D. Jurafsky, J. Chai, N. Schluter, and J. R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5553–5563. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.492. URL https://doi.org/10.18653/v1/2020.acl-main.492.

K. Hanawa, S. Yokoi, S. Hara, and K. Inui. Evaluation of similarity-based explanations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=9uvhpyQwzM_.

T. Hastie, A. Montanari, S. Rosset, and R. J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Z. Izzo, M. A. Smart, K. Chaudhuri, and J. Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021.

K. K and A. Søgaard. Revisiting methods for finding influential examples. *CoRR*, abs/2111.04683, 2021. URL https://arxiv.org/abs/2111.04683.

R. Khanna, B. Kim, J. Ghosh, and S. Koyejo. Interpreting black box predictions using fisher kernels. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 3382–3390. PMLR, 16–18 Apr 2019. URL https://proceedings.mlr.press/v89/khanna19a.html.

B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

P. W. Koh, J. Steinhardt, and P. Liang. Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47, Jan 2022. ISSN 1573-0565. doi: 10.1007/s10994-021-06119-y. URL https://doi.org/10.1007/s10994-021-06119-y.

P. W. W. Koh, K.-S. Ang, H. Teo, and P. S. Liang. On the accuracy of influence functions for measuring group effects. *Advances in neural information processing systems*, 32, 2019.

S. Kong, Y. Shen, and L. Huang. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*, 2021.

S. G. Krantz and H. R. Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

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

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 1097–1105, Red Hook, NY, USA, 2012. Curran Associates Inc.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

D. Lee, H. Park, T. Pham, and C. D. Yoo. Learning augmentation network via influence functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10961–10970, 2020.

J. Lee, L. Xiao, S. Schoenholz, Y. Bahri, R. Novak, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993. URL `https://aclanthology.org/J93-2004`.

J. Martens. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*, 2014.

J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

J. Martens et al. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

G. Pruthi, F. Liu, S. Kale, and M. Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

A. Schioppa, P. Zablotskaia, D. Vilar, and A. Sokolov. Scaling up influence functions. *CoRR*, abs/2112.03052, 2021. URL `https://arxiv.org/abs/2112.03052`.

P. Schulam and S. Saria. Can you trust this prediction? Auditing pointwise reliability after learning. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 1022–1031. PMLR, 16–18 Apr 2019. URL `https://proceedings.mlr.press/v89/schulam19a.html`.

P. Sedgwick. Pearson's correlation coefficient. *Bmj*, 345, 2012.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

C. Spearman. The proof and measurement of association between two things. 1961.

S. Teso, A. Bontempelli, F. Giunchiglia, and A. Passerini. Interactive label cleaning with example-based explanations. *Advances in Neural Information Processing Systems*, 34, 2021.

A. W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

P. Vicol, J. Lorraine, D. Duvenaud, and R. Grosse. Implicit regularization in overparameterized bilevel optimization. 2022a.

P. Vicol, J. Lorraine, F. Pedregosa, D. Duvenaud, and R. Grosse. On implicit bias in overparameterized bilevel optimization. In *International Conference on Machine Learning*. PMLR, 2022b.

S. Wachter, B. D. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box automated decisions and the GDPR. *Harvard Journal of Law & Technology*, 31(2), 2018.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Y.-Y. Yang and K. Chaudhuri. Understanding rare spurious correlations in neural networks. *arXiv preprint arXiv:2202.05189*, 2022.

C.-K. Yeh, J. Kim, I. E.-H. Yen, and P. K. Ravikumar. Representer point selection for explaining deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper/2018/file/8a7129b8f3edd95b7d969dfc2c8e9d9d-Paper.pdf`.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

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] See Section 5.
   (b) Did you describe the limitations of your work? [Yes] See Section 4 for ways in which the PBRF differs from LOO retraining and differs from influence estimates.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work is fundamental in nature and does not enable immediate misuse.
   (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 background in Section 3 and Appendix B.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
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)? [No] We have not attached the code.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provide training details in section 6 and in C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See full results in Table 5.
   (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 Appendix C.
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 dataset citations in section 6.1

(b) Did you mention the license of the assets? [Yes] All datasets are licensed under MIT data license.

(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]