INFLUENCE FUNCTIONS IN DEEP LEARNING ARE FRAGILE

Samyadeep Basu,* Phillip Pope *& Soheil Feizi
Department of Computer Science
University of Maryland, College Park
{sbasu12,pepope,sfeizi}@cs.umd.edu

ABSTRACT

Influence functions approximate the effect of training samples in test-time predictions and have a wide variety of applications in machine learning interpretability and uncertainty estimation. A commonly-used (first-order) influence function can be implemented efficiently as a post-hoc method requiring access only to the gradients and Hessian of the model. For linear models, influence functions are well-defined due to the convexity of the underlying loss function and are generally accurate even across difficult settings where model changes are fairly large such as estimating group influences. Influence functions, however, are not well-understood in the context of deep learning with non-convex loss functions. In this paper, we provide a comprehensive and large-scale empirical study of successes and failures of influence functions in neural network models trained on datasets such as Iris, MNIST, CIFAR-10 and ImageNet. Through our extensive experiments, we show that the network architecture, its depth and width, as well as the extent of model parameterization and regularization techniques have strong effects in the accuracy of influence functions. In particular, we find that (i) influence estimates are fairly accurate for shallow networks, while for deeper networks the estimates are often erroneous; (ii) for certain network architectures and datasets, training with weight-decay regularization is important to get high-quality influence estimates; and (iii) the accuracy of influence estimates can vary significantly depending on the examined test points. These results suggest that in general influence functions in deep learning are fragile and call for developing improved influence estimation methods to mitigate these issues in non-convex setups.

1 INTRODUCTION

In machine learning, influence functions (Cook & Weisberg, 1980) can be used to estimate the change in model parameters when the empirical weight distribution of the training samples is perturbed infinitesimally. This approximation is cheaper to compute compared to the expensive process of repeatedly re-training the model to retrieve the exact parameter changes. Influence functions could thus be used to understand the effect of removing an individual training point (or, groups of training samples) on the model predictions at the test-time. Leveraging a first-order Taylor's approximation of the loss function, (Koh & Liang, 2017) has shown that a (first-order) influence function, computed using the gradient and the Hessian of the loss function, can be useful to interpret machine learning models, fix mislabelled training samples and create data poisoning attacks.

Influence functions are in general well-defined and studied for models such as logistic regression (Koh & Liang, 2017), where the underlying loss-function is convex. For convex loss functions, influence functions are also accurate even when the model perturbations are fairly large (e.g. in the group influence case (Koh et al., 2019b; Basu et al., 2020)). However, when the convexity assumption of the underlying loss function is violated, which is the case in deep learning, the behaviour of influence functions is not well understood and is still an open area of research. With recent advances in computer vision (Szeliski, 2010), natural language processing (Sebastiani, 2002), high-stakes applications such as medicine (Lundervold & Lundervold, 2018), it has become particularly important

*Authors contributed equally

to interpret deep model predictions. This makes it critical to understand influence functions in the context of deep learning, which is the main focus of our paper.

Despite their non-convexity, it is sometimes believed that influence functions would work for deep networks. The excellent work of (Koh & Liang, 2017) successfully demonstrated one example of influence estimation for a deep network, a small (2600 parameters), "all-convolutional" network (Springenberg et al., 2015). To the best of our knowledge, this is the one of the *few* cases for deep networks where influence estimation has been shown to work. A question of key importance to practitioners then arises: for what other classes of deep networks does influence estimation work? In this work, we provide a comprehensive study of this question and find a pessimistic answer: *influence estimation is quite fragile for a variety of deep networks.*

In the case of deep networks, several factors might have an impact on influence estimates: (i) due to non-convexity of the loss function, different initializations of the perturbed model can lead to significantly different model parameters (with approximately similar loss values); (ii) even if the initialization of the model is fixed, the curvature values of the network (i.e. eigenvalues of the Hessian matrix) at optimal model parameters might be very large in very deep networks, leading to a substantial Taylor's approximation error of the loss function and thus resulting in poor influence estimates; (iii) for large neural networks, computing the exact inverse-Hessian Vector product, required in computation of influence estimates, can be computationally very expensive. Thus, one needs to use approximate inverse-Hessian Vector product techniques which might be erroneous; resulting in low quality influence estimates; and finally (iv) different architectures can have different loss landscape geometries near the optimal model parameters, leading to varying influence estimates.

In this paper, we study aforementioned issues of using influence functions in deep learning through an extensive experimental study on progressively-growing complex models and datasets. We first start our analysis with a case study of a small neural network for the Iris dataset where the exact Hessian matrix can be computed. We then progressively increase the complexity of the network and analyse a CNN architecture (depth of 6) trained on 10% of MNIST dataset, similar to (Koh & Liang, 2017). Next, we evaluate the accuracy of influence estimates for more complex deep architectures (e.g. ResNets) trained on MNIST and CIFAR-10. Finally, we compute influence estimates on the ImageNet dataset using ResNet-50.

We make the following observations through our analysis:

- We find that the network depth and width have a strong impact on influence estimates. In particular, we show that influence estimates are fairly accurate when the network is shallow, while for deeper models, influence estimates are often erroneous. We attribute this partially to the increasing curvature values of the network as the depth increases.
- We observe that the weight decay regularization is important to obtain high quality influence estimates in certain architectures and datasets.
- We show that the inverse-Hessian Vector product approximation techniques such as stochastic estimation (Agarwal et al., 2016) are erroneous, especially when the network is deep. This can contribute to the low quality of influence estimates in deep models.
- We observe that the choice of test-point has a substantial impact on the quality of influence estimates, across different datasets and architectures.
- In very large-scale datasets such as ImageNet, we have found that even ground-truth influence estimates (obtained by leave-one-out re-training) can be inaccurate and noisy partially due to the model's training and convergence.

These results highlight sensitivity of current influence functions in deep learning and call for developing robust influence estimators to be used in large-scale machine learning applications.

2 Related Works

Influence functions are primarily used to identify important training samples for test-time predictions and debug machine learning models (Koh & Liang, 2017). Similar to influence functions, (Chaudhuri & Mykland, 1993) tackles the problem of approximating a dataset using a subset of the dataset. In recent times, there is an increase in the applications of influence functions for tasks other than interpretability. For e.g.(Schulam & Saria, 2019) has used influence functions to audit

the reliability of test-predictions. In NLP, influence functions have been used to detect biases in word-embeddings (Brunet et al., 2018) whereas in the domain of ML security, influence functions have been shown to be effective in crafting stronger data-poisoning attacks (Koh et al., 2019a). Influence functions are also effective in the identification of important training groups (rather than an individual sample) (Basu et al., 2019; Koh et al., 2019b). Prior theoretical work (Giordano et al., 2018; 2019) have focused on quantifying finite sample error-bounds for influence estimates when compared to the ground-truth re-training procedures. Recently, alternative methods to find influential samples in deep networks have been proposed. In (Yeh et al., 2018), test-time predictions are explained by a kernel function evaluated at the training samples. Influential training examples can also be obtained by tracking the change in loss for a test-prediction through model-checkpoints, which are stored during the training time (Pruthi et al., 2020). While these alternative methods (Yeh et al., 2018; Pruthi et al., 2020) work well for deep networks in interpreting model predictions, they lack the "jackknife" like ability of influence functions which makes it useful in multiple applications other than interpretability (e.g. uncertainty estimation).

3 Basics of Influence Function

Consider h to be a function parameterized by θ which maps from an input feature space $\mathcal{X}$ to an output space denoted by $\mathcal{Y}$. The training samples are denoted by the set $\mathcal{S} = \{z_i : (x_i, y_i)\}_{i=1}^{n}$, while the loss function is represented by $\ell(h_\theta(z))$ for a particular training example z. The standard empirical risk minimization solves the following optimization problem:

$$\theta^* = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(z_i)). \tag{1}$$

Up-weighting a training example z by an infinitesimal amount ϵ leads to a new set of model parameters denoted by $\theta^{\epsilon}_{\{z\}}$. This set of new model parameters $\theta^{\epsilon}_{\{z\}}$ is obtained by solving:

$$\theta^{\epsilon}_{\{z\}} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(h_\theta(z_i)) + \epsilon \ell(h_\theta(z)). \tag{2}$$

Removing a training point z is similar to up-weighting its corresponding weight by $\epsilon = -1/n$ in Equation(2). The main idea used by (Koh & Liang, 2017) is to approximate $\theta^{\epsilon}_{\{z\}}$ by the first-order Taylor series expansion around the optimal model parameters represented by θ^*, which leads to:

$$\theta^{\epsilon}_{\{z\}} \approx \theta^* - \epsilon H^{-1}_{\theta^*} \nabla_\theta \ell(h_{\theta^*}(z)), \tag{3}$$

where H_{θ^*} represents the Hessian with respect to model parameters θ^*. Following the classical result of (Cook & Weisberg, 1980), the change in the model parameters ($\Delta\theta = \theta^{\epsilon}_{\{z\}} - \theta^*$) on up-weighting the training example z can be approximated by the influence function ($\mathcal{I}(z)$) as follows:

$$\mathcal{I}(z) = \frac{d\theta^{\epsilon}_{\{z\}}}{d\epsilon}|_{\epsilon=0} = -H^{-1}_{\theta^*} \nabla_\theta \ell\left(h_{\theta^*}(z)\right). \tag{4}$$

The change in the loss value for a particular test point z_t when a training point z is up-weighted can be approximated as a closed form expression by the chain rule (Koh & Liang, 2017):

$$\mathcal{I}(z, z_t) = -\nabla \ell(h_{\theta^*}(z_t))^T H^{-1}_{\theta^*} \nabla \ell(h_{\theta^*}(z)). \tag{5}$$

$\mathcal{I}(z, z_t)/n$ is approximately the change in the loss for the test-sample z_t when a training sample z is removed from the training set. This result is, however, based on the assumption that the underlying loss function is strictly convex in the model parameters θ and the Hessian H_{θ^*} is a positive-definite matrix (Koh & Liang, 2017). For large models, inverting the exact Hessian H_{θ^*} is expensive. In such cases, the inverse-Hessian Vector product can be computed efficiently with a combination of Hessian-vector product (Pearlmutter, 1994) and optimization techniques (see Appendix for details).

4 What Can Go Wrong for Influence Functions In Deep Learning?

First-order influence functions (Koh & Liang, 2017) assume that the underlying loss function is convex and the change in model parameters is small when the empirical weight distribution of the training data is infinitesimally perturbed. In essence, this denotes the Taylor's gap in Equation (3)

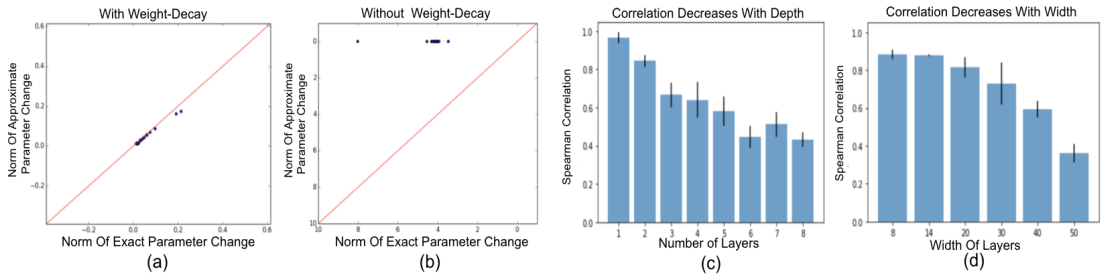

Figure 1: Iris dataset experimental results - (a,b) Comparison of norm of parameter changes computed with influence function vs re-training; (a) trained with weight-decay; (b) trained without weight-decay. (c) Spearman correlation vs. network depth. (d) Spearman correlation vs. network width.

to be small for an accurate influence estimate. However in the case of non-convex loss functions, this assumption is *not* generally true. Empirically, we find that the Taylor's gap is strongly affected by common hyper-parameters for deep networks. For example, in Fig. (1)-(a,b), we find that for networks trained without a weight-decay regularization on Iris, the Taylor's gap is large resulting in low quality influence estimates. In a similar vein, when the network depth and width is considerably large (i.e. the over-parameterized regime), the Taylor's gap increases and substantially degrades the quality of influence estimates (Fig. (2)). Empirically this increase in Taylor's gap strongly correlates with the curvature values of the loss function evaluated at the optimal model parameters as observed in Fig. (2-(b)).

Further complications may arise for larger models, where influence estimations in such settings require an additional approximation to compute the inverse-Hessian vector product. Nonetheless, we observe in Fig. (2)-(a), that on Iris this approximation has only a marginal impact on the influence estimation. These results show that that network architecture, hyper-parameters, and loss curvatures are important factors for proper influence estimations. In the next section, we discuss these issues in details through controlled experiments on datasets and models of increasing complexity.

5 Experiments

Datasets: We first study the behaviour of influence functions in a small Iris dataset (Anderson, 1936), where the exact Hessian can be computed. Further, we progressively increase the complexity of the model and datasets: we use small MNIST (Koh & Liang, 2017) to evaluate the accuracy of influence functions in a small CNN architecture with a depth of 6. Next, we study influence functions on modern deep architectures trained on the standard MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2000) datasets. Finally, to understand how influence functions scale to large datasets, we use ImageNet (Deng et al., 2009) to compute the influence estimates.

Evaluation Metrics: We evaluate the accuracy of influence estimates at a given test point z_t using both Pearson (Kirch, 2008) and Spearman rank-order correlation (Spearman, 1904) with the ground-truth (obtained by re-training the model) across a set of training points. Most of the existing interpretability methods desire that influential examples are ranked in the correct order of their importance (Ghorbani et al., 2017). Therefore, to evaluate the accuracy of influence estimates, Spearman correlation is often a better choice.

5.1 Understanding Influence Functions when the Exact Hessian Can be Computed

Setup: Computing influence estimates with the exact Hessian has certain advantages in our study: a) it bypasses inverse-Hessian Vector product approximation techniques which induce errors in computing influence estimates. Thus, we can compare influence estimates computed with exact vs. approximate inverse-Hessian Vector products to quantify this type of error; b) The deviation of the parameters computed with the influence function from the exact parameters can be computed exactly. This information can be useful to further quantify the error incurred by (first-order) influence estimates in the non-convex setup. However, computations of the exact Hessian matrix and its inverse are only computationally feasible for models with small number of parameters. Thus, we use the Iris dataset along with a small feed-forward neural network to analyse the behaviour of influence

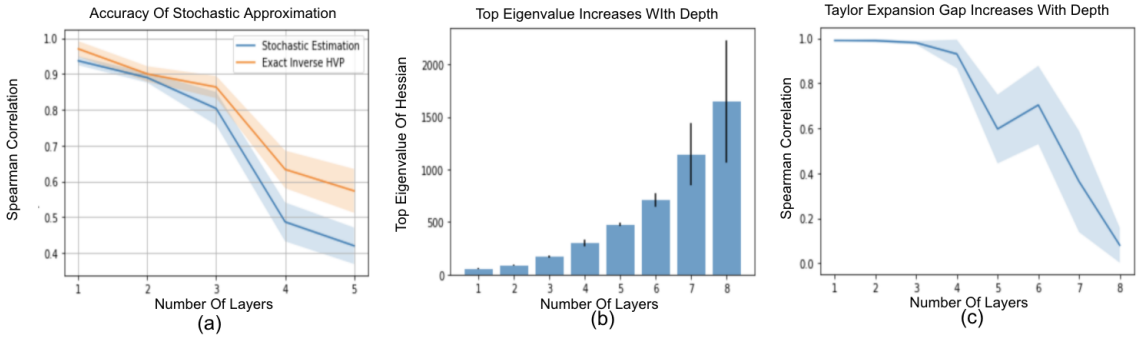

Figure 2: Iris dataset experimental results; (a) Spearman correlation of influence estimates with the ground-truth estimates computed with stochastic estimation vs. exact inverse-Hessian vector product. (b) Top eigenvalue of the Hessian vs. the network depth. (c) Spearman correlation between the norm of parameter changes computed with influence function vs. re-training.

function computed with the exact Hessian in a non-convex setup. We train models to convergence for 60k iterations with full-batch gradient descent. To obtain the ground-truth estimates, we re-train the models for 7.5k steps, starting from the optimal model parameters. For our analysis, we choose the test-point with the maximum loss and evaluate the accuracy of influence estimates with the ground-truth amongst of the top 16.6% of the training points. Through our experiments with the exact Hessian, we answer some relevant questions related to how properties of the network such as depth, width and regularizers (e.g. weight-decay) affect the influence estimates.

The Effect of Weight-Decay: One of the simpler and common regularization techniques used to train neural networks is weight-decay regularization. In particular, a term $\lambda\|\theta\|_2^2$, penalizing the scaled norm of the model parameters is added to the objective function, during training, where λ is a hyperparameter which needs to be tuned. We train a simple feed-forward network[1] with and without weight-decay regularization. For the network trained with weight-decay, we observe a Spearman correlation of 0.97 between the influence estimates and the ground-truth estimates. In comparison, for the network trained without a weight-decay regularization, the Spearman correlation estimates decrease to 0.508. In this case, we notice that the Hessian matrix is singular, thus a damping factor of 0.001 is added to the Hessian matrix, to make it invertible. To further understand the reason for this decrease in the quality of influence estimates, we compare the following metric across all training examples: a) Norm of the model parameter changes computed by re-training; b) Norm of the model parameter changes computed using the influence function (i.e. $\|H_{\theta^*}^{-1}\nabla\ell(z_i)\|_2 \quad \forall i \in [1, n]$) (Fig. 1-(a,b)). We observe that when the network is trained without weight-decay, changes in model parameters computed with the influence function have a substantially larger deviation from those computed using re-training. This essentially suggests that the gap in Taylor expansion, using (first-order) influence estimates is large, when the model is trained without weight-decay. We observe similar results with smooth activation functions such as tanh (see the Appendix for details).

The Effect Of Network Depth: From Fig. 1-(c), we see that network depth has a dramatic effect on the quality of influence estimates. For example, when the depth of the network is increased to 8, we notice a considerable decrease in the Spearman correlation estimates. To further our understanding about the decrease in the quality of influence estimates when the network is deeper, we compute the gap in the approximation between the ground-truth parameter changes (computed by re-training) and the approximate parameter changes (computed using the influence function). To quantify the error gap, we compute the Spearman correlation estimates between the norm of true and approximate parameter changes across the top 16.6% of the influential examples. We find that with increasing depth, the Spearman correlation estimates between the norm of the true and approximate parameter changes decrease. From Fig. 2-(c), we see that the approximation error gap is particularly large when the depth of the network is more than 5. We also notice a consistent increase in the curvature of the loss function (Fig. 2-(b)), as the network becomes deeper. This possibly suggests that the curvature information of the network can be an upper bound in the approximation error gap between

[1]With width of 5, depth of 1 and ReLU activations

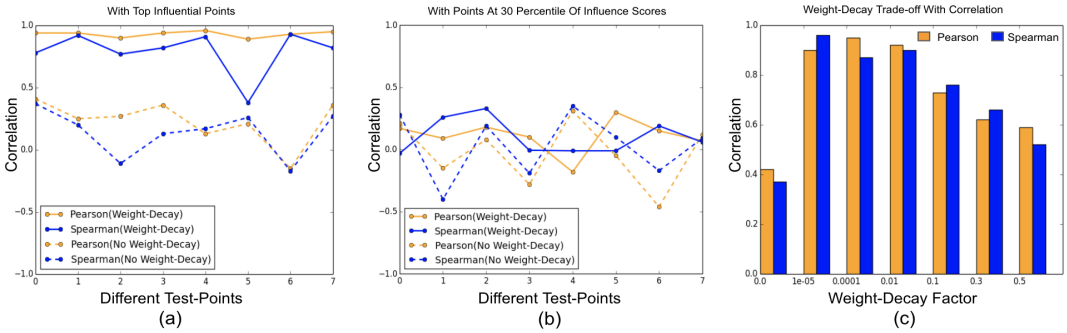

Figure 3: Experiments on small MNIST using a CNN architecture. (a) Estimation of influence function with and without weight decay on (a) the top influential points, (b) training points at 30^{th} percentile of influence score distribution. (c) Correlation vs the weight decay factor (evaluated on the top influential points).

the true parameters and the ones computed using the influence function. Even in case of non-smooth activation functions like ReLU, we have a similar observation. (see the Appendix for more details).

The Effect Of Network Width: To see the effect of the network width on the quality of influence estimates, we evaluate the influence estimates for a feed-forward network of constant depth, by progressively increasing its width. From Fig. 1-(d), we observe that with an increase in network width, the Spearman correlation decreases consistently. For example, we find that the Spearman correlation decreases from 0.82 to 0.56, when the width of the network is increased from 8 to 50. This observation suggests that over-parameterizing a network by increasing its width has a strong impact in the quality of influence estimates.

The Effect of Stochastic Estimation on inverse-Hessian Vector Product: For large deep networks, the inverse-Hessian Vector product is computed using stochastic estimation(Agarwal et al., 2016), as the exact Hessian matrix cannot be computed and inverted. To understand the effectiveness of stochastic approximation, we compute the influence estimates with both the exact Hessian and stochastic estimation. We observe that across different network depths, the influence estimates computed with stochastic estimation have a marginally lower Spearman correlation when compared to the ones computed with the exact Hessian. From Fig. 2-(a), we find that the error in the approximation is more, when the network is deeper.

5.2 Understanding Influence Functions in Shallow CNN Architectures

Setup: In this section, we perform a case study using a CNN architecture[2] on the small MNIST dataset (i.e. 10% of MNIST); a similar setup used in (Koh & Liang, 2017). To assess the accuracy of influence estimates, we select a set of test-points with high test-losses computed at the optimal model parameters. For each of the test points, we select 100 training samples with the highest influence scores and compute the ground-truth influence by re-training the model. We also select 100 training points with influence scores at the 30^{th} percentile of the entire influence score distribution. These training points have low influence scores and a lower variance in their scores when compared to the top influential points. The model is trained with and without weight-decay regularization.

When trained with a weight-decay and evaluated based on the top influential points, we find that the correlation estimates are consistently significant (Fig. 3-(a)). This is consistent with the results reported in (Koh & Liang, 2017). However, when the evaluation is done with the set of training samples at the 30^{th} percentile of the influence score distribution, the correlation estimates decrease significantly (Fig. 3-(b)). This shows that influence estimates of only the top influential points are precise when compared to ground-truth re-trainings. Furthermore, without the weight-decay regularization, influence estimates in both cases are poor across all the test-points (Fig. 3-(a,b)).

To further understand the impact of weight-decay on influence estimates, we train the network with different weight-decay regularization factors. From Fig. 3-(c), we see that the selection of weight-

[2]The model has 2600 parameters and is trained for 500k iterations to reach convergence with the optimal model parameters θ^*. The ground-truth estimates are obtained by re-training the models from the optimal parameter set θ^* for 30k iterations. When trained with a weight-decay, a regularization factor of 0.001 is used.

Dataset	MNIST						CIFAR-10					
	A (With Decay)		B (With Decay)		A (Without Decay)		A (With Decay)		B (With Decay)		A (Without Decay)	
Architecture	P	S	P	S	P	S	P	S	P	S	P	S
Small CNN	**0.95**	**0.87**	**0.92**	**0.82**	**0.41**	0.35	-	-	-	-	-	-
LeNet	**0.83**	**0.51**	0.28	0.29	0.18	0.12	**0.81**	**0.69**	0.45	0.46	0.19	0.09
VGG13	0.34	**0.44**	0.29	0.18	0.38	0.31	**0.67**	**0.63**	**0.66**	**0.63**	**0.79**	**0.73**
VGG14	0.32	0.26	0.28	0.22	0.21	0.11	**0.61**	**0.59**	**0.49**	0.41	**0.75**	**0.64**
ResNet18	**0.49**	0.26	0.39	0.35	0.14	0.11	**0.64**	**0.42**	0.25	0.26	**0.72**	**0.69**
ResNet50	0.24	0.22	0.29	0.19	0.08	0.13	**0.46**	0.36	0.24	0.09	0.32	0.14

Table 1: Correlation estimates on MNIST And CIFAR-10 ; A=Test-point with highest loss; B=Test-point at the 50^{th} percentile of test-loss spectrum; P=Pearson correlation; S=Spearman correlation

decay factor is important in getting high-quality influence estimates. For this specific CNN architecture, we notice that the correlations start decreasing when the weight-decay factor is greater than 0.01. Moreover, from Fig. 3-(a,b), we find that the selection of test-point also has a strong impact on the quality of influence estimates. For example, when the network is trained with weight-decay and the influence estimates are computed for top influential training points, we notice that the Spearman correlation estimates range from 0.92 to 0.38 across different test-points and have a high variance.

These results show that despite some successful applications of influence functions in this non-convex setup, as reported in (Koh & Liang, 2017), their performances are very sensitive to hyper-parameters of the experiment as well as to the training procedure. In the next two sections, we assess the quality of influence estimates on more complex architectures and datasets including MNIST, CIFAR-10 and ImageNet. In particular, we desire to understand, if the insights gained from experiments on smaller networks can be generalized to more complex networks and datasets.

5.3 Understanding Influence Functions in Deep Architectures

Setup: In this section, we evaluate the accuracy of influence estimates using MNIST and CIFAR-10 datasets across different network architectures including small CNN(Koh & Liang, 2017), LeNet (Lecun et al., 1998), ResNets (He et al., 2015), and VGGNets (Simonyan & Zisserman, 2015)[3]. To compute influence estimates, we choose two test points for each architecture: a) the test-point with the highest loss, and b) the test-point at the 50^{th} percentile of the losses of all test points. For each of these two test points, we select the top 40 influential training samples and compute the correlation of their influence estimates with the ground-truth estimates. To compute the ground-truth influence estimates, we follow the strategy of (Koh & Liang, 2017), where we re-train the models from optimal parameters for 6% of the steps used for training the optimal model. When the networks are trained with a weight-decay regularization, we use a constant weight-decay factor of 0.001 across all the architectures (see Appendix for more details).

Results On MNIST: From Table 1, we observe that for the test-point with the highest loss, the influence estimates in the small CNN and LeNet architectures (trained with the weight-decay regularization) have high qualities. These networks have 2.6k and 44k parameters, respectively, and are relatively smaller and less deep than the other networks used in our experimental setup. As the depth of the network increases, we observe a consistent decrease in quality of influence estimates. For the test-point with a loss at the 50^{th} percentile of test-point losses, we observe that influence estimates *only* in the small CNN architecture have good qualities.

Results On CIFAR-10: For CIFAR-10, across all architectures trained with the weight-decay regularization, we observe that the correlation estimates for the test-point with the highest loss are highly significant. For example, the correlation estimates are above 0.6 for a majority of the network architectures. However, for the test-point evaluated at the 50^{th} percentile of the loss, the correlations decrease marginally across most of the architectures. We find that on CIFAR-10, even architectures trained without weight-decay regularization have highly significant correlation estimates when evaluated with the test-point which incurs the highest loss.

[3]For CIFAR-10, evaluations on small CNN have not been performed due to the poor test accuracy.

In case of MNIST, we have found that in shallow networks, the influence estimates are fairly accurate while for deeper networks, the quality of influence estimates decrease. For CIFAR-10, although the influence estimates are significant, we found that the correlations are marginally lower in deeper networks such as ResNet-50. The improved quality of influence estimates in CIFAR-10 can be attributed to the fact that for a similar depth, architectures trained on CIFAR-10 are less over-parameterized compared to architectures trained on MNIST. Note that, in Section 5.1, where the exact Hessian matrix can be computed, we observed that over-parameterization decreases the quality of influence estimates. From Table(1), we also observed that the selection of test-point has a sizeable impact on the quality of influence estimates. Furthermore, we noticed large variations in the quality of influence estimates across different architectures. In general we found that influence estimates for small CNN and LeNet are reasonably accurate, while for ResNet-50, the quality of estimates decrease across both MNIST and CIFAR-10. Precise reasons for these variations are difficult to establish. We hypothesize that it can be due to the following factors: (i) Different architectures trained on different datasets have contrasting characteristics of loss landscapes at the optimal parameters which can have an impact on influence estimates. (ii) The weight-decay factor may need to be set differently in various architectures, to obtain high quality influence estimates.

Results on CIFAR-100: In the case of CIFAR-100, we train a ResNet-18 model with a weight-decay regularization factor of $5e^{-4}$. The influence estimates are then computed for test-points with the highest losses (Index: 6017, 2407, 9383) and test-points around the 50^{th} percentile of the test loss (Index: 783, 7106) over multiple model initialisations. Unlike in the case of MNIST and CIFAR-10, from Fig. 4 we observe the correlation estimates to be of substantially poor quality. We provide additional visualizations of the influential training examples in the Appendix section.

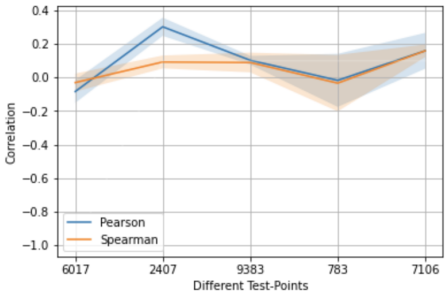

Figure 4: Influence for CIFAR-100

5.4 Is Scaling Influence Estimates To ImageNet Possible?

The application of influence functions to ImageNet scale models provides an appealing yet challenging opportunity. It is appealing because, if successful, it opens a range of applications to large-scale image models, including interpretability, robustness, data poisoning, and uncertainty estimation. It is challenging for a number of reasons. Notable among these is the high computational cost of training and re-training, which limits the number of ground truth evaluations. In addition, all of the previously discussed difficulties in influence estimations still remain, including (i) non-convexity of the loss, (ii) selection of scaling and damping hyperparameters in the stochastic estimation of the Hessian, and (iii) the lack of convergence of the model parameters. The scale of ImageNet raises additional questions about the feasibility of leave-one-out retraining as the ground truth estimator. Given that there are 1.2M images in the training set, *is it even possible that the removal of one image can significantly alter the model?* In other words, we question whether or not reliable ground truth estimates may be obtained through leave-one-out re-training at this scale.

To illustrate this, we conduct an additional influence estimation on ImageNet. After training an initial model to 92.302% top5 test accuracy, we select two test points at random, calculate influence over the entire training set, and then select the top 50 points by their influences as candidates for re-training. We then use the re-training procedure suggested by (Koh & Liang, 2017), which starts leave-one-out re-training from the parameter set obtained after the initial training. We re-train for an additional 2 epochs, approximately 5% of the original training time, and calculate the correlations. We observe that for both test points, both Pearson and Spearman correlations are very low (less than 0.15, see details in the Appendix).

In our experiments, we observe high variability among ground-truth estimates obtained by re-training the model (see the appendix for details). We conjecture that this may be partially due to the fact that the original model has not be fully converged. To study this, we train the original model with *all* training points for an additional 2 epochs and measure the change in the test loss. We find that the overall top5 test accuracy has improved slightly to 92.336 % (+0.034) and the loss for one

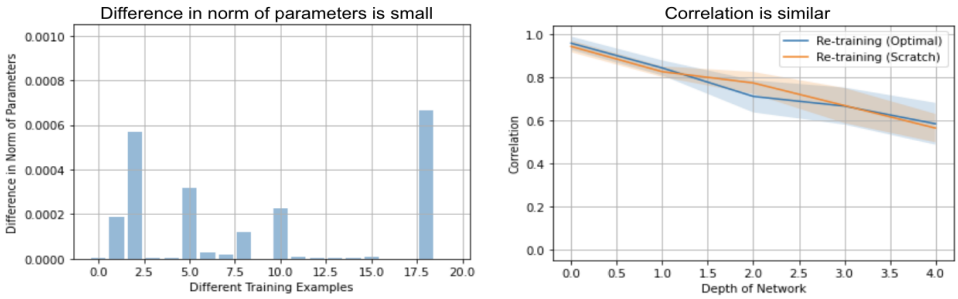

Figure 5: (a) Difference in norm of parameters obtained by re-training from scratch vs. re-training from optimal parameters. (b) Correlation estimates with re-training from scratch vs. re-training from optimal parameters.

of the considered test points has decreased by relatively a significant amount of 0.679. However, the loss for the other point has increased slightly by 0.066. Such changes in loss values can therefore out-power the effect of leave-one-out re-training procedure. Second, we calculate the 2-norm of the weight gradients, which should be close to zero near an optimal point, and compare it to a standard pre-trained ImageNet ResNet-50 model as a baseline. We find these norms to be 20.18 and and 15.89, respectively, showing our model has similar weight gradient norm to the baseline. Although these norms are relatively small given that there are 25.5M parameters, further re-training the model still changes loss values for some samples considerably, making the ground-truth estimates noisy. We suggest that one way to obtain reliable ground-truth influence estimates in such large models can be through assessing the influence of a group of samples, rather than a single one.

6 Discussion on ground-truth influence

In our experimental setup, to obtain the ground-truth influence, we follow the strategy of re-training from optimal model parameters as shown in (Koh & Liang, 2017; Koh et al., 2019b). Even for moderately sized datasets and architectures, re-training from scratch (instead of re-training from optimal model parameters) is computationally expensive. Although re-training from optimal model parameters is an approximation compared to re-training from scratch, we notice that the approximation works quite well in practice. To validate the effectiveness of this strategy, we first compute the norm of the difference in parameters obtained by re-training from scratch vs. re-training from optimal parameters. Next we compute the correlation between the influence estimates and ground-truth using both the re-training strategies. From Fig. 5, we observe the norm of parameter differences using the two re-training strategies to be small. Similarly using both the re-training strategies as ground-truth yield similar correlation estimates. These results highlight that re-training from optimal parameters (although an approximation) is close to re-training from scratch.

7 Conclusion

In this paper, we present a comprehensive analysis of the successes and failures of influence functions in deep learning. Through our experiments on datasets including Iris, MNIST, CIFAR-10, CIFAR-100, ImageNet and architectures including LeNet, VGGNets, ResNets, we have demonstrated that influence functions in deep learning are fragile in general. We have shown that several factors such as the weight-decay, depth and width of the network, the network architecture, stochastic approximation and the selection of test points, all have strong effects in the quality of influence estimates. In general, we have observed that influence estimates are fairly accurate in shallow architectures such as small CNN(Koh & Liang, 2017) and LeNet, while in very deep and wide architectures such as ResNet-50, the estimates are often erroneous. Additionally, we have scaled up influence computations to the ImageNet scale, where we have observed influence estimates are highly imprecise. These results call for developing robust influence estimators in the non-convex setups of deep learning.

8 ACKNOWLEDGEMENTS

Authors thank Daniel Hsu, Alexander D'Amour and Pang Wei Koh for helpful discussions. This project was supported in part by NSF CAREER AWARD 1942230, HR001119S0026-GARD-FP-052, AWS Machine Learning Research Award, a sponsorship from Capital One, and Simons Fellowship on "Foundations of Deep Learning".

REFERENCES

Naman Agarwal, Brian Bullins, and Elad Hazan. Second order stochastic optimization in linear time. *ArXiv*, abs/1602.03943, 2016.

Anderson. Iris flower dataset. In -, 1936.

Samyadeep Basu, Xuchen You, and Soheil Feizi. Second-order group influence functions for black-box predictions. *ArXiv*, abs/1911.00418, 2019.

Samyadeep Basu, Xuchen You, and Soheil Feizi. On second-order group influence functions for black-box predictions. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 715–724. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/basu20b.html.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *in COMPSTAT*, 2010.

Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard S. Zemel. Understanding the origins of bias in word embeddings. *CoRR*, abs/1810.03611, 2018. URL http://arxiv.org/abs/1810.03611.

Probal Chaudhuri and Per A. Mykland. Nonlinear experiments: Optimal design and inference based on likelihood. *Journal of the American Statistical Association*, 88(422):538–546, 1993. doi: 10.1080/01621459.1993.10476305. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1993.10476305.

R. Dennis Cook and Sanford Weisberg. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980. ISSN 0040-1706. doi: 10.1080/00401706.1980.10486199.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Amirata Ghorbani, Abubakar Abid, and James Y. Zou. Interpretation of neural networks is fragile. In *AAAI*, 2017.

Ryan Giordano, Will Stephenson, Runjing Liu, Michael I. Jordan, and Tamara Broderick. A swiss army infinitesimal jackknife. In *AISTATS*, 2018.

Ryan Giordano, Michael I. Jordan, and Tamara Broderick. A higher-order swiss army infinitesimal jackknife. *ArXiv*, abs/1907.12116, 2019.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour. *CoRR*, abs/1706.02677, 2017. URL http://arxiv.org/abs/1706.02677.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Jeremy Howard et al. fastai. https://github.com/fastai/fastai, 2018.

Alon Jacovi and Yoav Goldberg. Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness? *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020. doi: 10.18653/v1/2020.acl-main.386. URL http://dx.doi.org/10.18653/v1/2020.acl-main.386.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Wilhelm Kirch (ed.). *Pearson's Correlation Coefficient*, pp. 1090–1091. Springer Netherlands, Dordrecht, 2008. ISBN 978-1-4020-5614-7. doi: 10.1007/978-1-4020-5614-7_2569. URL https://doi.org/10.1007/978-1-4020-5614-7_2569.

P. W. Koh, J. Steinhardt, and P. Liang. Stronger data poisoning attacks break data sanitization defenses. *arXiv preprint arXiv:1811.00741*, 2019a.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/koh17a.html.

Pang Wei Koh, Kai-Siang Ang, Hubert H. K. Teo, and Percy Liang. On the accuracy of influence functions for measuring group effects. *CoRR*, abs/1905.13289, 2019b. URL http://arxiv.org/abs/1905.13289.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *CVPR*, 2000. URL cifar.com.

Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pp. 2278–2324, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on MRI. *CoRR*, abs/1811.10052, 2018. URL http://arxiv.org/abs/1811.10052.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160, January 1994. ISSN 0899-7667. doi: 10.1162/neco.1994.6.1.147. URL http://dx.doi.org/10.1162/neco.1994.6.1.147.

Garima Pruthi, Frederick Liu, Mukund Sundararajan, and Satyen Kale. Estimating training data influence by tracking gradient descent. *ArXiv*, abs/2002.08484, 2020.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL http://arxiv.org/abs/1609.04747.

Peter G. Schulam and Suchi Saria. Can you trust this prediction? auditing pointwise reliability after learning. In *AISTATS*, 2019.

Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1): 1–47, March 2002. ISSN 0360-0300. doi: 10.1145/505282.505283. URL http://doi.acm.org/10.1145/505282.505283.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks, 2018.

Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. -, 1994.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Leslie N. Smith. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL http://arxiv.org/abs/1803.09820.

C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15:88–103, 1904.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6806.

DAWNBench: An End-to-End Deep Learning Benchmark and Competition, 2017. Stanford. URL https://dawn.cs.stanford.edu/benchmark/papers/nips17-dawnbench.pdf.

Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 2010. ISBN 1848829345, 9781848829343.

Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. Representer point selection for explaining deep neural networks. *CoRR*, abs/1811.09720, 2018. URL http://arxiv.org/abs/1811.09720.

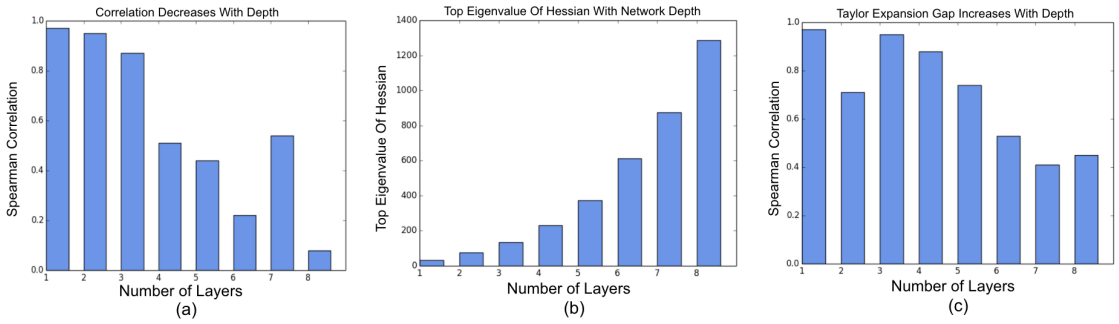

Figure 6: Additional Iris experimental results for ReLU networks: (a) Spearman correlation vs. network depth; (b) Top eigenvalue of the Hessian vs.network depth; (c) Spearman correlation between the norm of parameter changes computed with influence function vs. re-training.

9 Appendix

9.1 Additional Experimental Results on Iris Dataset

In this section, we provide additional experimental results to understand the effect of network depth on the correlation estimates for ReLU networks. From Fig. 6, we observe that even in case of architectures trained with non-smooth activation functions such as ReLU, the correlation estimates consistently decrease with depth. Similar to our findings in case of networks trained with tanh activation (as shown in the main text), we observe that the top eigenvalue of the Hessian matrix and the Taylor's approximation gap increases with depth. In the main text, we reported that when a network with ReLU activation is trained with a weight-decay regularization, the correlation estimates are significant and the Taylor's approximation gap is less. We find a similar result even with smoother activation functions such as tanh. From Fig. 7, we observe that when a network with tanh activation is trained with a weight-decay regularization, the Taylor's approximation gap is less. However when the network is trained without a weight-decay regularization, the Taylor's expansion gap is large resulting in poor quality of influence estimates.

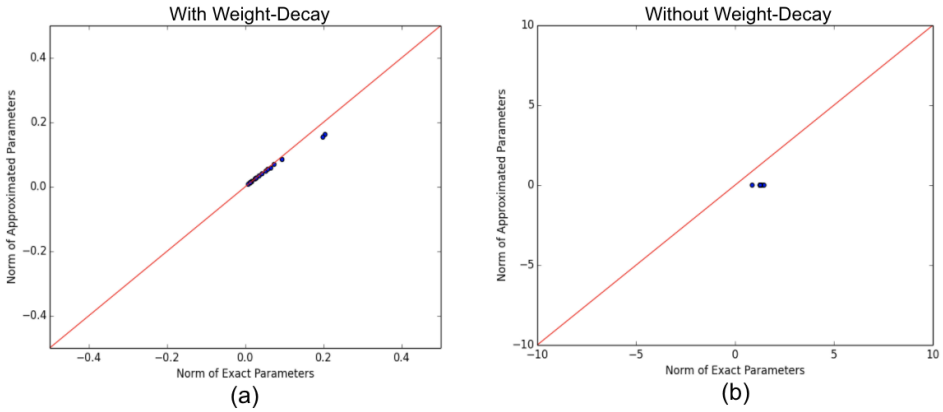

Figure 7: Additional Iris experimental results for tanh networks; (a) When trained with weight-decay, the Taylor's approximation gap is small; (b) When trained without weight-decay, the Taylor's expansion gap is large. These results are similar to our findings for ReLU networks which are reported in the main text.

9.2 WHAT DOES WEIGHT-DECAY DO?

In our experiments, we observe that with increasing network depth, the correlation between the influence estimates and the ground-truth estimates decrease considerably. Additionally with increasing depth, the loss curvature values increase. We notice that with a high-value of weight-decay, the loss curvature for deeper networks decrease, which also leads to improvement in correlation values between the influence estimates and the ground-truth. For e.g. in Fig. 8, with a weight-decay value of 0.03, the Spearman correlation estimates are 0.47. With a relatively higher weight-decay factor of 0.075, the correlation values improve to 0.72. Increasing the weight-decay factor from 0.03 to 0.075, also decreases the loss curvature values substantially. These results highlight that the selection of weight-decay factor is crucial to obtain high-quality influence estimates, especially for deeper overparameterized networks.

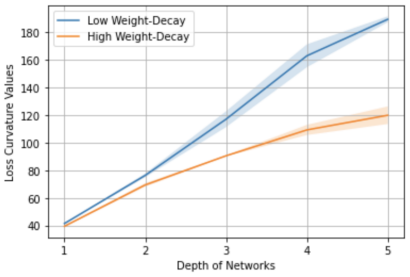

Figure 8: Correlations with different training samples

9.3 VISUALISATION OF TOP INFLUENTIAL POINTS

In this section, we visualise the top influential training samples corresponding to a given test-point. In the main text, we noted that the selection of test-points has a strong impact on the quality of influence estimates. Additionally, we also observe that the selection of test-points has an impact on the semantic-level similarities between inferred influential training points and the test-points being evaluated. For example, in Fig. 9, we observe that 2 out of the top 5 influential points are not from the same class as the test-point with index 1479. However in Fig. 10, we observe that all the top 5 influential training samples are semantically similar and from the same class as the evaluated test-point with index 7196.

Figure 9: Top 5 influential points for the test point: 1479 (CIFAR-10). The model is a ResNet-18 trained with a weight-decay regularization; Only 3 out of the 5 points are semantically similar to the test-point with class "Bird".

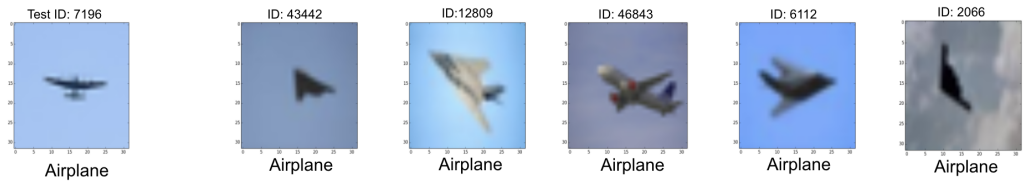

Figure 10: Top 5 influential points for the test point: 7196 (CIFAR-10). The model is a ResNet-18 trained with a weight-decay regularization; All the 5 training points are semantically similar to the test-point from the class "Airplane".

Architecture	Influence Computation Time (MNIST)	Influence Computation Time (CIFAR-10)
Small CNN	141.13 ± 0.51	N/A
LeNet	162.6 ± 2.20	136.39 ± 3.16
VGG13	3886.23 ± 3.45	4416.54 ± 2.01
VGG14	4619.11 ± 5.08	4620.69 ± 6.11
ResNet-18	960.08 ± 4.67	910.58 ± 8.49
ResNet-50	4323.13 ± 8.26	3857.66 ± 21.6

Table 2: Computational running times for influence function across different architectures

9.4 Running Times

In this section, we provide computational running times for (first-order) influence function estimations. We note that in models with a large number of parameters, the influence computation is relatively slow. However, even in large deep models, it is still faster than re-training the model for every training example. In our implementation, for a given test-point z_{test}, we first compute $c = H_{\theta^*}^{-1} \nabla \ell(h_{\theta^*}(z_{test}))$ once which is the most computationally expensive step. We then compute a vector dot product i.e. $c^T \nabla \ell(h_{\theta^*}(z_i)) \, \forall i \in [1, n]$. In Table 2, we provide the computational running times for estimating influence functions in different network architectures.

9.5 Additional Experimental Details on ImageNet Influence Calculations

In this section we give further details on the influence estimation on ImageNet. To help address the high computational cost of training and re-training, we utilize highly optimized ImageNet training schemes such as those submitted to the DAWNBench competition (Sta, 2017). In particular we use the scheme published from (Howard et al., 2018)[4], for the ResNet-50 architecture which uses several training tricks including progressive image resizing, weight decay tuning, dynamic batch sizes (Goyal et al., 2017), learning rates (Smith, 2018), and half-precision floats. Although these techniques are unorthodox, they are sufficient for our purposes since we need only to compare between the fully trained and re-trained models. We replicate this scheme and obtain a top-5 validation accuracy of 92.302%.

We now give further details on the test points selected. The first has a test loss at the 83rd percentile (loss=2.634, index = 13,923, class=kit fox), the second has the test loss at the 37th percentile (loss=0.081, index = 2,257, class =gila monster), where the indices refer to where they appear in `test_loader.loader.dataset` . We visualize these test points in Figure 12.

Next, for each of these test points, we compute influence across the entire dataset and select the top 50 *training* points by influence scores. We visualize 25 of these points in Figures 13 and 14. We observe that there is qualitative similarity between the test points and *some* of their respective most influential training points, but not others. Although there is qualitative similarity is some cases, the results are still overall weak quantitatively

We plot the obtained correlations in Figure 11.

For computing the weight gradient norm, we take the mean norm in batches of size 128 over the entire dataset for both our model and a standard PyTorch pretrained model as a baseline, both of which are ResNet-50 models with around 25.5M parameters.

[4]https://www.fast.ai/2018/08/10/fastai-diu-imagenet/

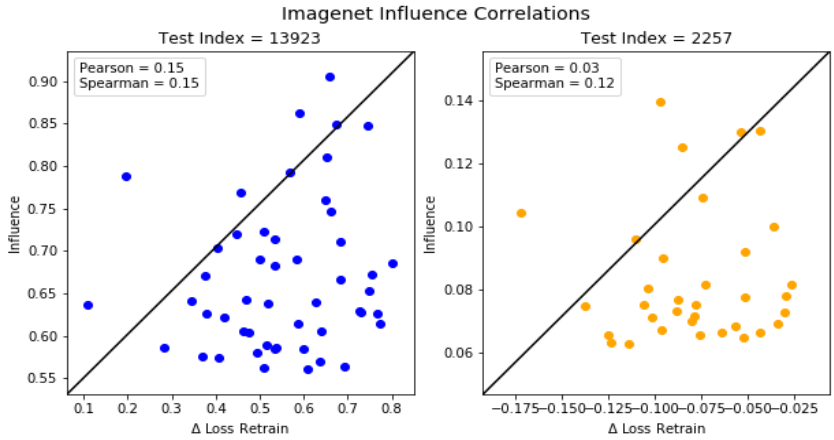

Figure 11: ImageNet influence estimation results for the selected test points 13,923 (left) and 2,257 (right). X-axis is change in test loss after removal of a training point and retraining as described in the text. Y-axis is the change in test loss estimated with influence function. Pearson and Spearman correlations are shown in the caption. Correlations are low, showing the weakness of this influence estimation.

Figure 12: Selected test points for influence estimation.

9.6 COMPUTING INVERSE-HESSIAN VECTOR PRODUCT

In large over-parameterized deep networks, computing and inverting the exact Hessian H_{θ^*} is expensive. In such cases, the Hessian-vector product rule (Pearlmutter, 1994) is used along with conjugate-gradient (Shewchuk, 1994) or stochastic estimation (Agarwal et al., 2016) to compute the approximate inverse-Hessian Vector product. More specifically, to compute $t = H_{\theta^*}^{-1} v$, we solve the following optimization problem using conjugate-gradient: $t^* = \arg\min_t \{\frac{1}{2} t^T H_{\theta^*} t - v^T t\}$, where $v = \nabla_\theta \ell(h_{\theta^*}(z_t))$. This optimization, however, requires the Hessian H_{θ^*} to be a positive definite matrix, which is not true in case of deep networks due to the presence of negative eigenvalues. In practice, the Hessian can be regularized by adding a damping factor of λ to its eigenvalues (i.e. $H_{\theta^*} + \lambda I$) to make it positive definite.

In deep models, with a large number of parameters and large training set, conjugate-gradient is often expensive as it requires computing the Hessian-vector product (Pearlmutter, 1994) for every data sample in the training set. In those cases, stochastic estimation techniques (Agarwal et al.,

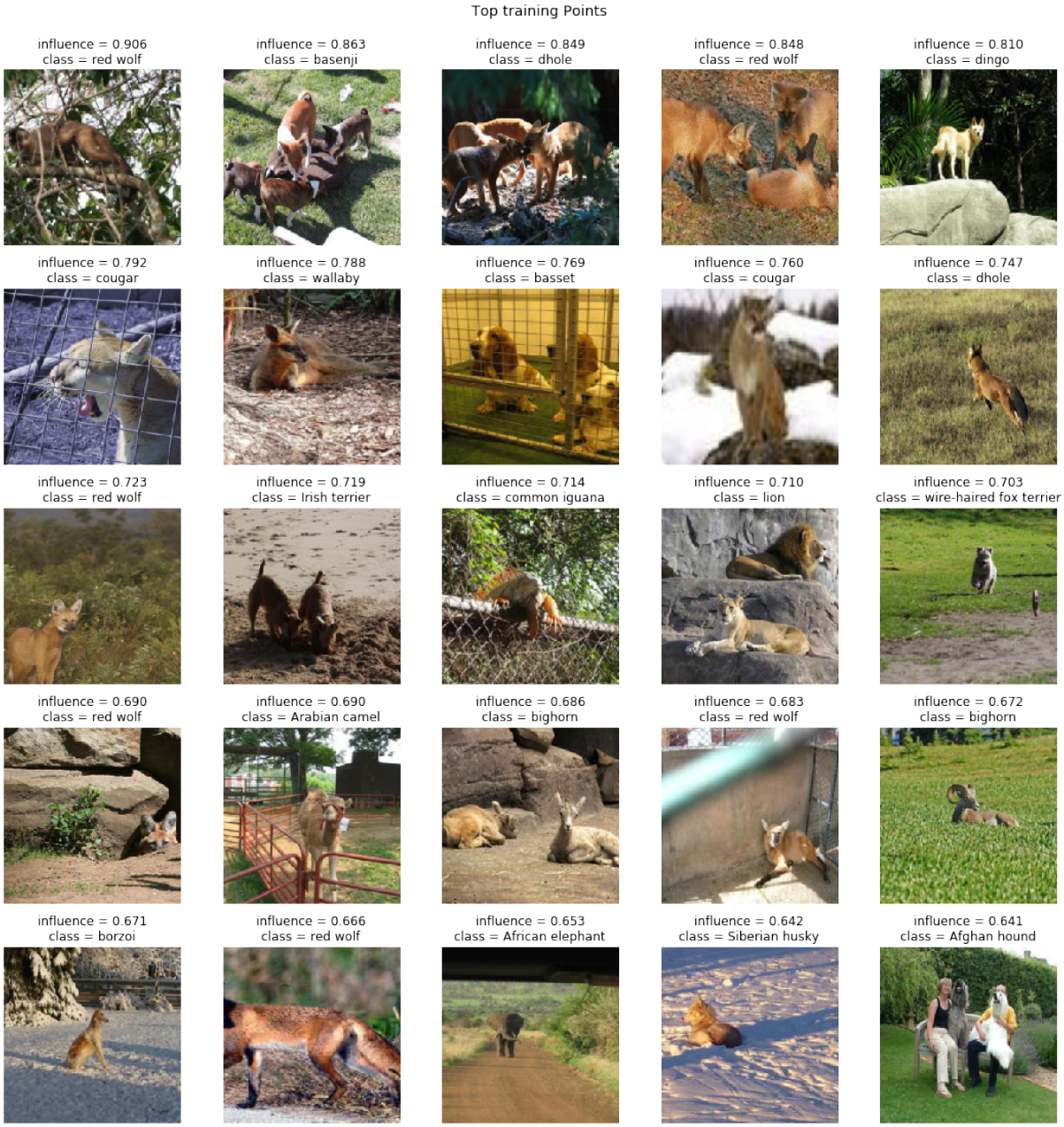

Figure 13: Top 25 ImageNet training points by influence for test point 13,293, kit fox. Many of the identified classes are furred mammals, e.g. red wolf, basenji, and dingo, which have visual similarity to the test point. Other examples are questionable, e.g. the common iguana, and African elephant. Although there is qualitative similarity is some cases, the results are still overall weak quantitatively.

2016) have been used which are fast as they do not require going through all the training samples. In stochastic estimation, the inverse Hessian is computed using a recursive reformulation of the Taylor expansion: $H_j^{-1} = I + (I - H)H_{j-1}^{-1}$ where j is the recursion depth hyperparameter . A training example z_i is uniformly sampled and $\nabla^2 \ell(h_{\theta^*}(z_i))$ is used as an estimator for computing H. This

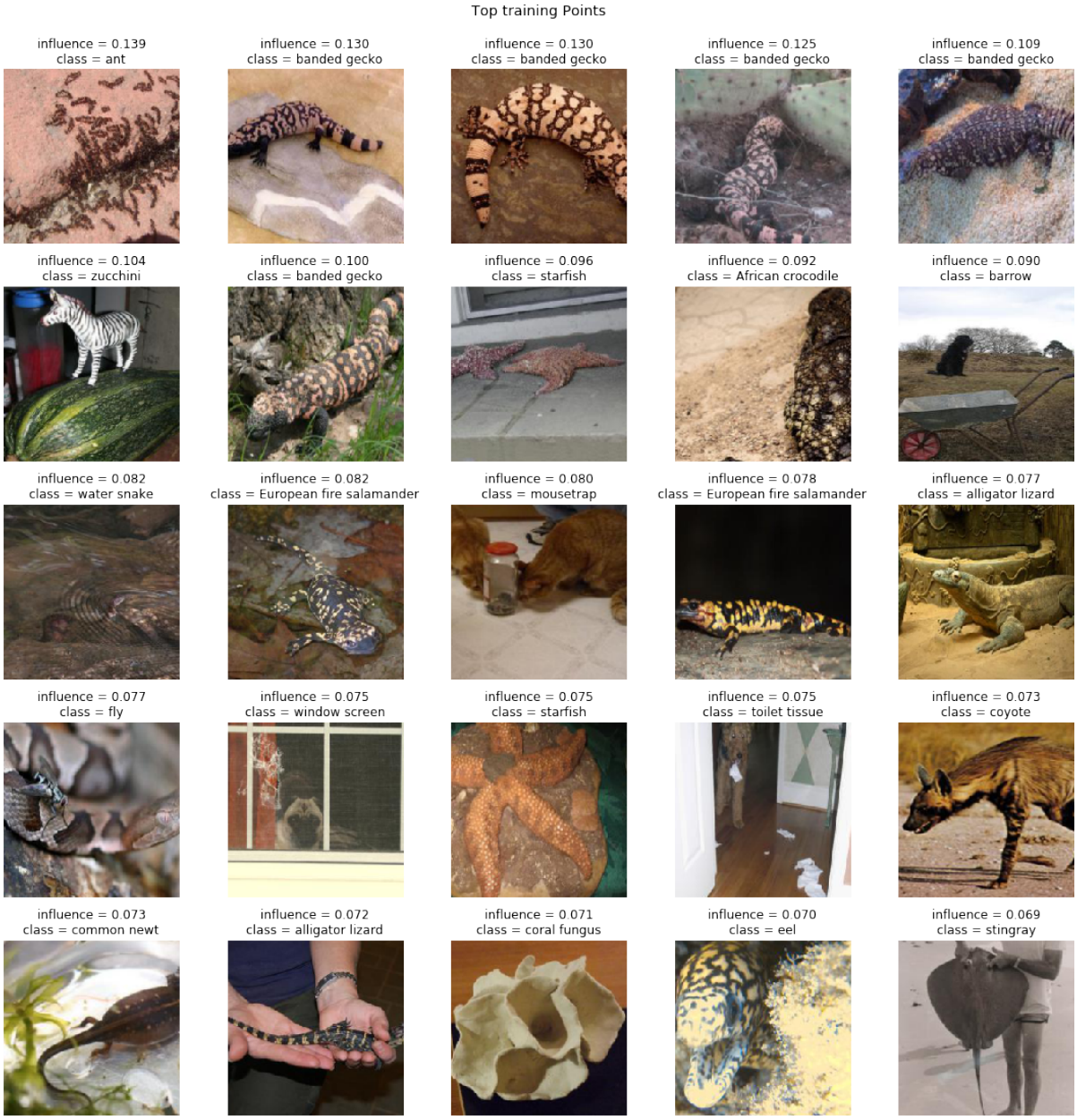

Figure 14: Top 25 ImageNet training points by influence for test point 2,257, gila monster. Many of the identified classes are spotted lizards, e.g. banded gecko ad European fire salamander, which have visual similarity to the test point. Other examples are questionable, e.g. the stingray, coral fungus, and barrow. Although there is qualitative similarity is some cases, the results are still overall weak quantitatively.

technique also requires tuning a scaling hyperparameter γ and a damping hyperparameter β [5]. In

[5]It is assumed that $\forall i, I - \nabla^2 \ell(h_{\theta^*}(z_i)) \succcurlyeq 0$; (Koh & Liang, 2017) notes that if this is not true, the loss can be scaled down without affecting the parameters. The scaling factor is a hyperparameter which helps the convergence of the Taylor series. The damping coefficient is added to the diagonal of the Hessian matrix to make it invertible.

our experiments with large deep models, we use the stochastic estimation method to compute the inverse-Hessian Vector product.

9.7 Effect of Initialisation and Optimizers on Influence Estimates

To understand the effect of network initialisation on the quality of influence estimates, we compute the influence scores across different random initialisations. The influence estimates are computed for the small CNN architecture (Koh & Liang, 2017) and LeNet (Lecun et al., 1998), both trained on the MNIST dataset. Both the architectures are trained with a constant weight-decay factor of 0.001. In Fig 15, we observe that across different network initialisations, although both the Pearson and Spearman correlations between the influence estimates and the ground-truth are inconsistent, the variance amongst them is particularly low. Note that for both the network architectures, we compute the influence estimates for the test-point with the highest loss at the optimal model parameters. The correlation between the influence estimates and leave-out retrainings are computed with the top 40 influential training examples. Additionally to understand the impact of the selection of optimizer on the influence estimates, we train the LeNet architecture on MNIST with different optimizers namely Adam (Kingma & Ba, 2014), Gradient Descent (Bottou, 2010), Nesterov and RMSProp (Ruder, 2016). We notice that the Pearson correlation (0.72 ± 0.04) has a marginally lower variance when compared to the Spearman rank-order correlation (0.56 ± 0.11).

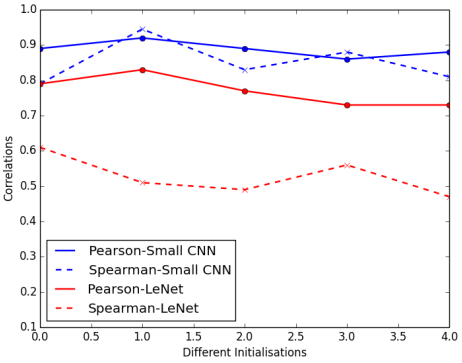

Figure 15: Correlations with different network initialisation

9.8 Effect of training sample selection for ground-truth influence

In this section, we understand the effect of selecting different number of training samples on the correlation estimates. We investigate this with a case study for a CNN architecture trained on small MNIST. Keeping a test-point with a high loss fixed, we sample different sets of training examples with the highest and the lowest influence scores over different network initialisations. Note that in this setting as shown in the main paper, the quality of influence estimates are relatively good. We observe that when the influence estimates are evaluated with the top influential points, both the Pearson and Spearman correlations are relevant. This is true across different number of training samples. However when the evaluation carried out with respect to the lowest influential training samples, the correlation estimates are of poor quality. These results highlight the importance of the selection of the type of training samples with respect to which the correlation estimates are computed.

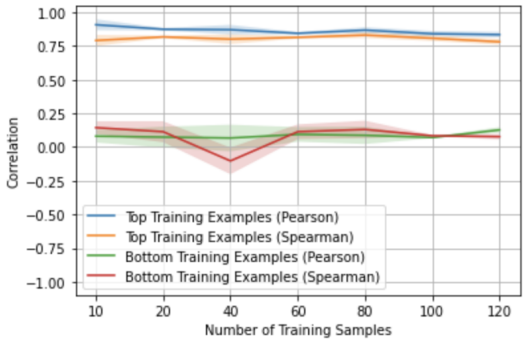

Figure 16: Correlations with different training samples

9.9 Faithfulness and Plausibility of Influence functions

The authors (Jacovi & Goldberg, 2020) primarily tackle the importance and trade-offs between plausibility (i.e. if the interpretations are convincing to humans) and faithfulness (i.e. how accurate an interpretation is to the "true reasoning process of the model") of existing interpretation methods. To the best of our knowledge such an analysis has not been done for influence functions. We observe that explanations from influence functions for deep networks are sometimes plausible and sometimes

not. For instance, in Appendix Fig. 9, we observe that the selection of test-point with (class = bird) leads to training examples with (class = deer) amongst the top influential points. On the other hand, in Appendix Fig. 10, we observe many plausible explanations. Influence functions that work are faithful because they answer the following question:"what would this model have done if certain data were excluded?". This class of questions, while not exhaustive, have special relevance because they are counterfactuals, which hold both intuitive appeal and for their special status in causal reasoning. However, we must be cautious because they may not be faithful when they incur approximation errors, as highlighted in our paper.

9.10 CIFAR-100 Influential Examples

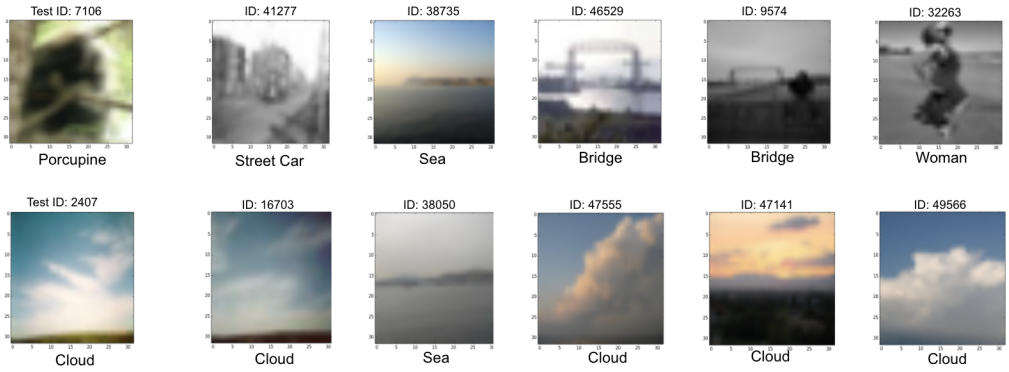

Figure 17: Top 5 influential points for the test points: 7106 and 2407 (CIFAR-100). The model is a ResNet-18 trained with a weight-decay regularization. For the test-point with index 7106, the influential training samples are semantically dissimilar from the test-point. However for the test-point with index 2407, 4 out of the top 5 samples share semantic similarity with the test-point.

9.11 Preliminary Results on Group Influence

Understanding model changes when a group of training samples are up-weighted is indeed an important research problem. Influence functions (Cook & Weisberg, 1980; Koh & Liang, 2017) in general are accurate when the model perturbation is small. However when a group of samples are up-weighted, the model perturbation is large, which violates the small perturbation assumption of influence functions. Previously it has been shown (Koh et al., 2019b; Basu et al., 2019) that group influence functions are fairly accurate for linear and convex models, even when the model perturbation is substantial. In this section, we present some preliminary results on the behaviour of group influence functions for non-convex models. Our main observation is that group influence functions are fairly accurate for small networks. Nonetheless for large and complex networks, the influence estimates are of poor quality. For e.g. in Fig. 18, we observe that the correlation estimates for small group sizes are accurate, whereas for larger group sizes, the estimates are of poor quality. For a ResNet-18 model trained on MNIST (with a weight-decay regularization factor of 0.001), we observe the correlation estimates across different group sizes to vary from 0.01 to 0.21. Similarly for a ResNet-18 trained on CIFAR-100, we observe the group influence correlation

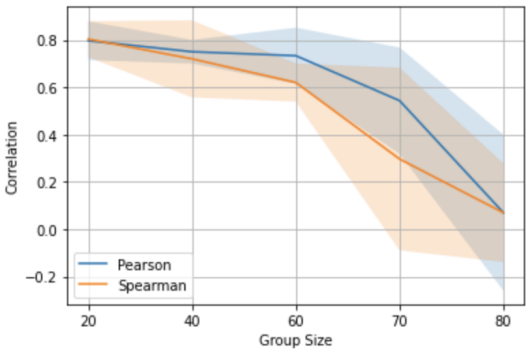

Figure 18: Group Influence on Iris

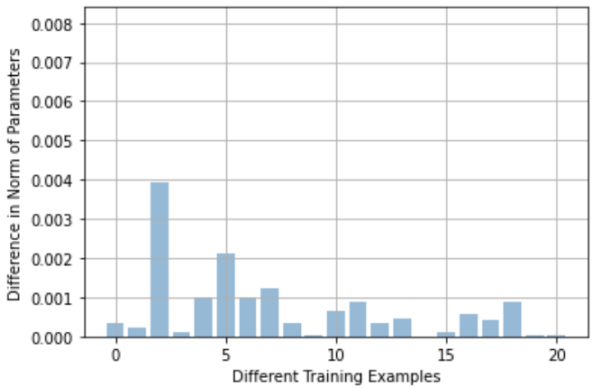

Figure 19: Norm of difference in parameters obtained by training from scratch vs. re-training from optimal parameters

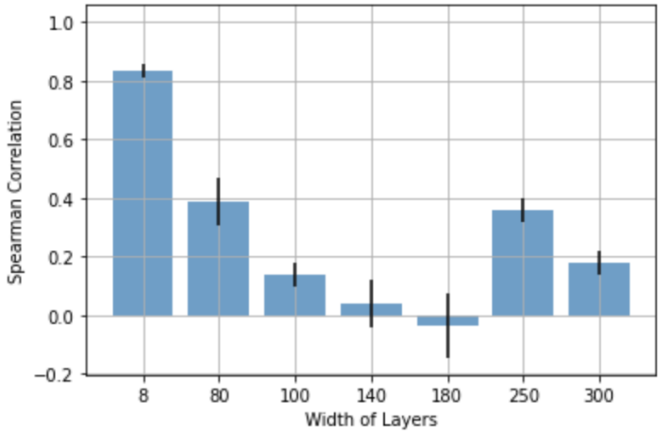

Figure 20: Width vs. Spearman Correlation for a one-layered network

estimates to range from 0.01 to 0.18. We leave the complete investigation of group influence in deep learning as a direction for future work.

9.12 Additional experiments with Multiple Test-Points

In our experimental setup, we evaluate the correlation estimates with respect to one test-point at a time. Although the evaluation of the correlation estimates with multiple test-points is more robust, it comes at the expense of high computational cost. To illustrate the quality of influence estimates with multiple test-points, we compute the influence estimates for small MNIST with 8 different test-points. We sample two test-points each from : (a) 100^{th} percentile of the test-loss; (b) 75^{th} percentile of the test-loss; (c) 50^{th} percentile of the test-loss; (d) 25^{th} percentile of the test-loss. The Pearson and Spearman correlations are 0.91 and 0.78 respectively. In a similar setting, for a complex architecture such as ResNet-18 trained on CIFAR-100, the Pearson and Spearman correlations are 0.15 and 0.11 respectively.

9.13 IMPACT OF ACTIVATION FUNCTIONS

In our experiments we observe that even with non-smooth activation functions such as ReLU, we obtain high quality influence estimates for certain networks. Understanding influence estimates with ReLU has an additional challenge since there measure zero subsets where the function is non-differentiable. Recently (Serra et al., 2018) has provided improved bounds on the number of linear regions for shallow ReLU networks. Understanding the impact of the number of linear regions in ReLU networks on the influence estimates is an interesting research direction, however we defer it for future work.