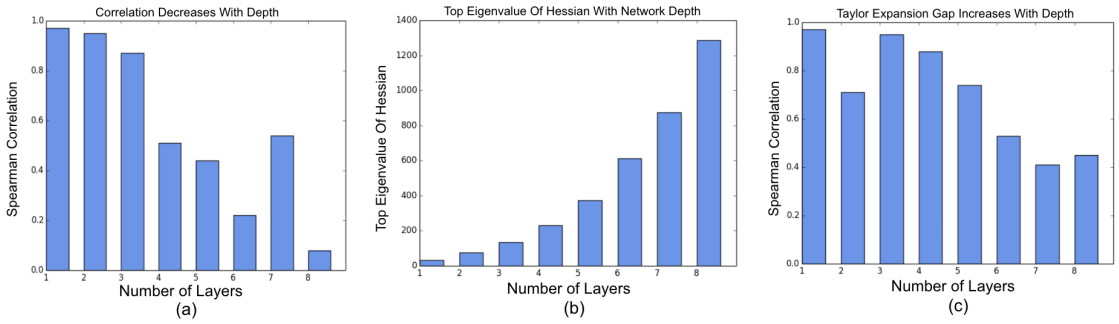

Figure 1: Additional Iris experimental results for ReLU networks: (a) Spearman correlation vs. network depth; (b) Top eigenvalue of the Hessian vs.network depth; (c) Spearman correlation between the norm of parameter changes computed with influence function vs. re-training.

1 Appendix

2 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. 1, 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. 2, 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.

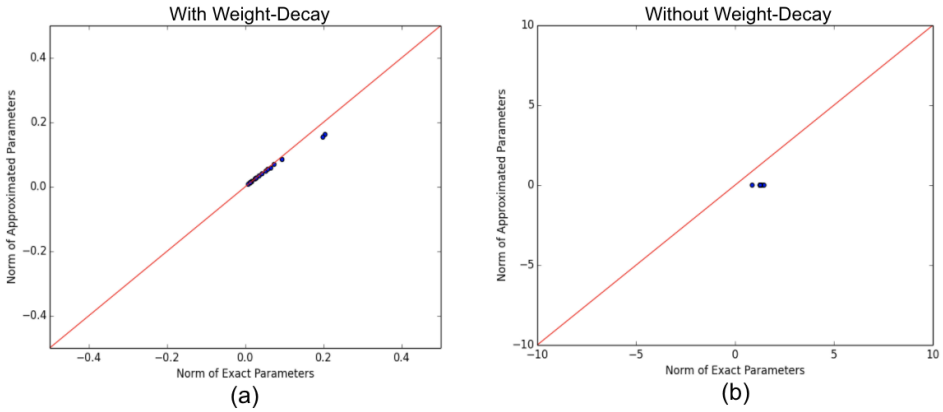

Figure 2: Additional Iris experimental results for tanh networks; (a) When trained with weight-decay, the Taylor's approximation gap is small; (b) When trained without weight-decay, the Taylor's expansion gap is large. These results are similar to our findings for ReLU networks which are reported in the main text.

3 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. 3, 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.

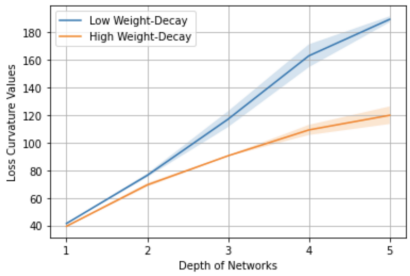

Figure 3: Correlations with different training samples

4 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. 4, 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. 5, 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 4: 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".

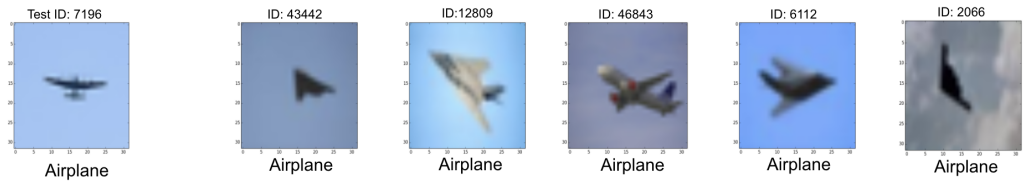

Figure 5: Top 5 influential points for the test point: 7196 (CIFAR-10). The model is a ResNet-18 trained with a weight-decay regularization; All the 5 training points are semantically similar to the test-point from the class "Airplane".

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 1: Computational running times for influence function across different architectures

5 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 1, we provide the computational running times for estimating influence functions in different network architectures.

6 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)[1], 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 7.

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 8 and 9. 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 6.

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.

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

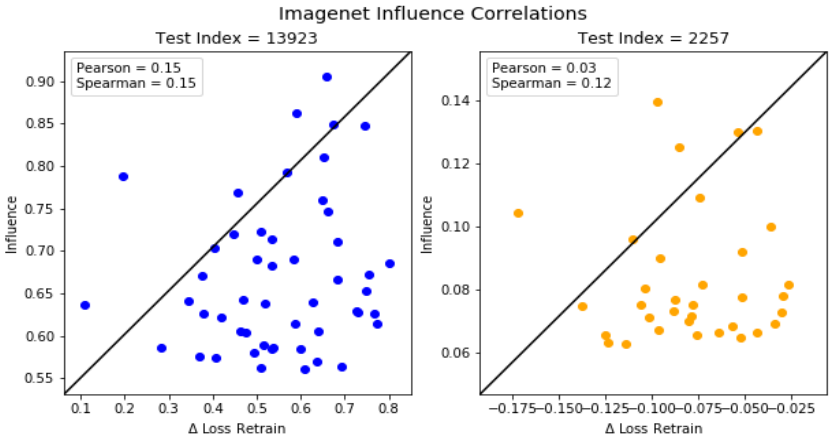

Figure 6: ImageNet influence estimation results for the selected test points 13,923 (left) and 2,257 (right). X-axis is change in test loss after removal of a training point and retraining as described in the text. Y-axis is the change in test loss estimated with influence function. Pearson and Spearman correlations are shown in the caption. Correlations are low, showing the weakness of this influence estimation.

Figure 7: Selected test points for influence estimation.

7 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.,

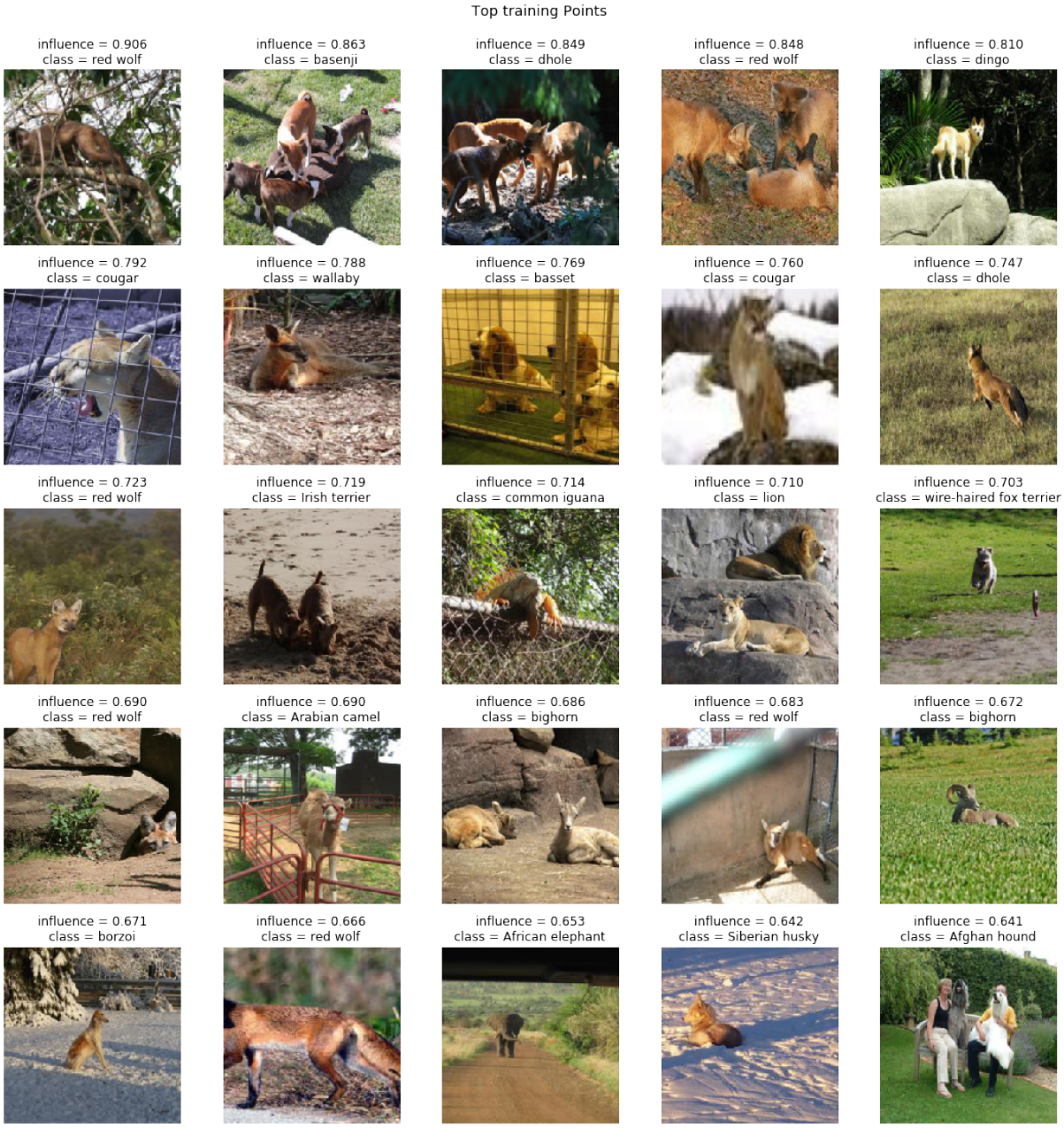

Figure 8: Top 25 ImageNet training points by influence for test point 13,293, kit fox. Many of the identified classes are furred mammals, e.g. red wolf, basenji, and dingo, which have visual similarity to the test point. Other examples are questionable, e.g. the common iguana, and African elephant. Although 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

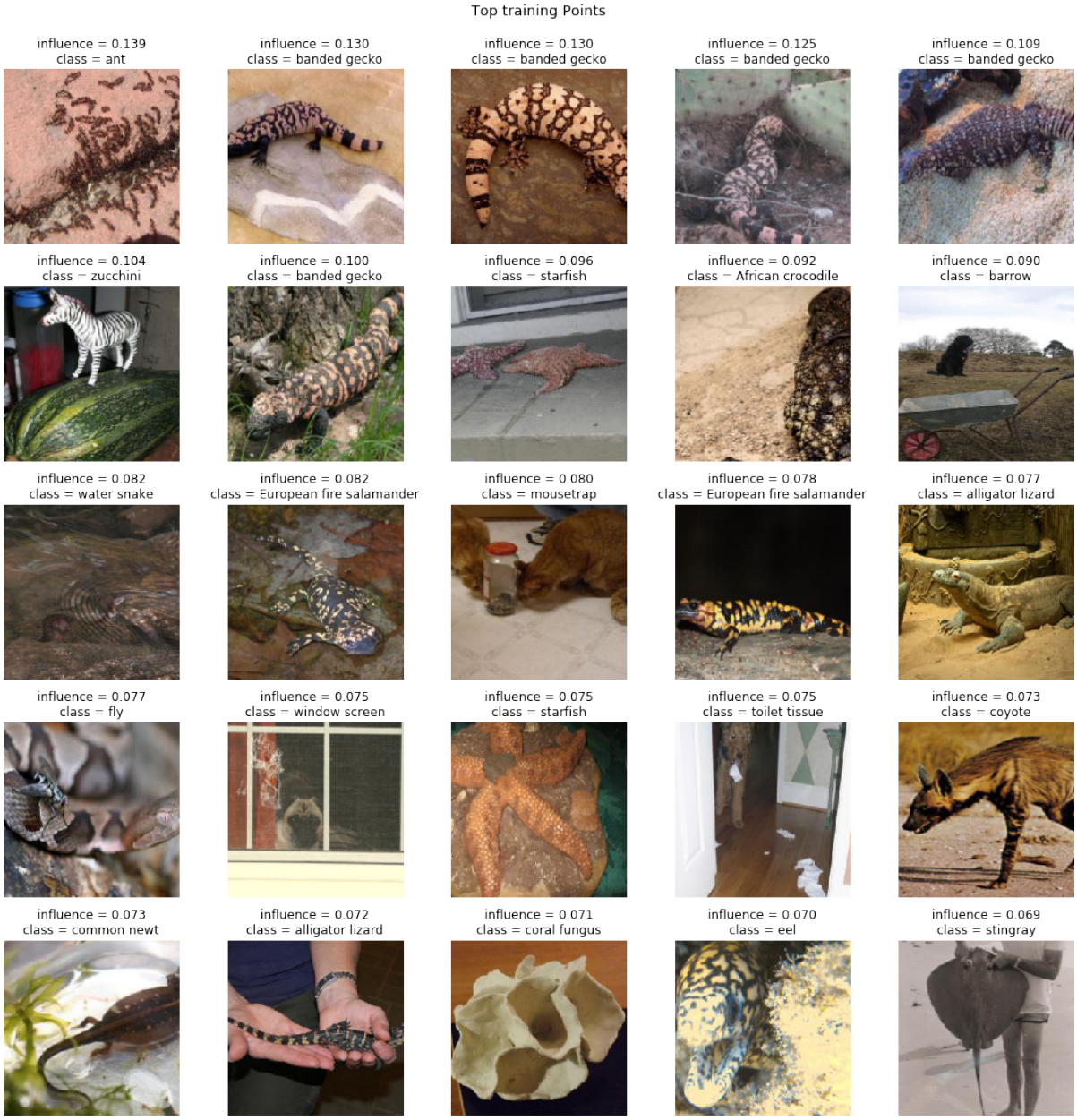

Figure 9: Top 25 ImageNet training points by influence for test point 2,257, gila monster. Many of the identified classes are spotted lizards, e.g. banded gecko ad European fire salamander, which have visual similarity to the test point. Other examples are questionable, e.g. the stingray, coral fungus, 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 [2]. In

[2]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.

8 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 10, 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).

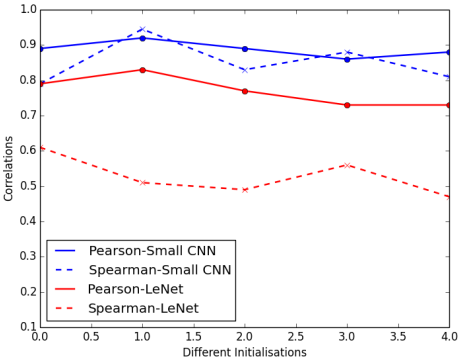

Figure 10: Correlations with different network initialisation

9 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.

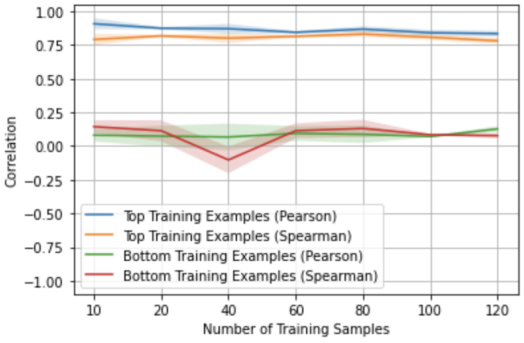

Figure 11: Correlations with different training samples

10 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. 4, 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. 5, 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.

11 CIFAR-100 Influential Examples

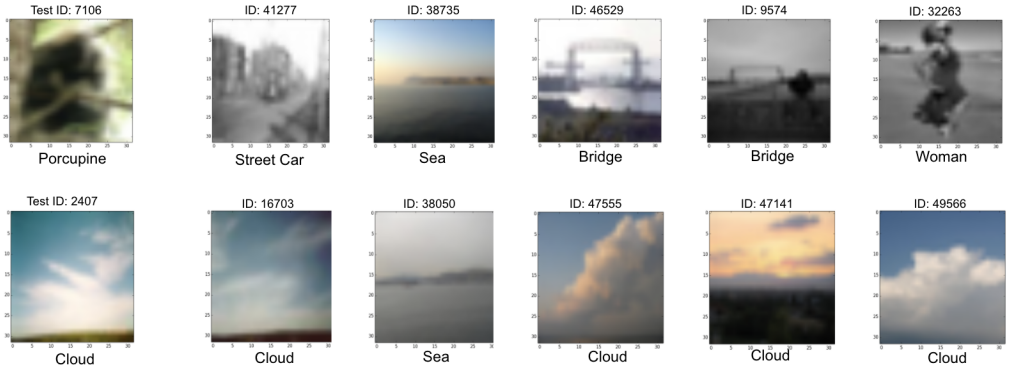

Figure 12: Top 5 influential points for the test points: 7106 and 2407 (CIFAR-100). The model is a ResNet-18 trained with a weight-decay regularization. For the test-point with index 7106, the influential training samples are semantically dissimilar from the test-point. However for the test-point with index 2407, 4 out of the top 5 samples share semantic similarity with the test-point.

12 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., 2019; 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. 13, we observe that the correlation estimates for small group sizes are accurate, whereas for larger group sizes, the estimates are of

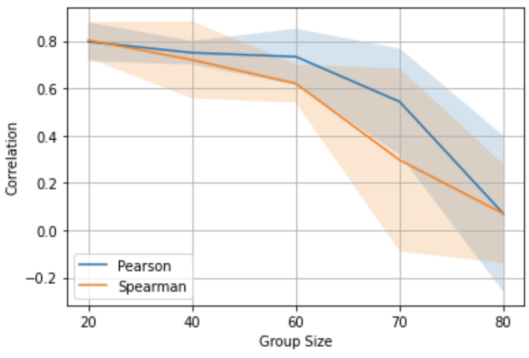

Figure 13: Group Influence on Iris

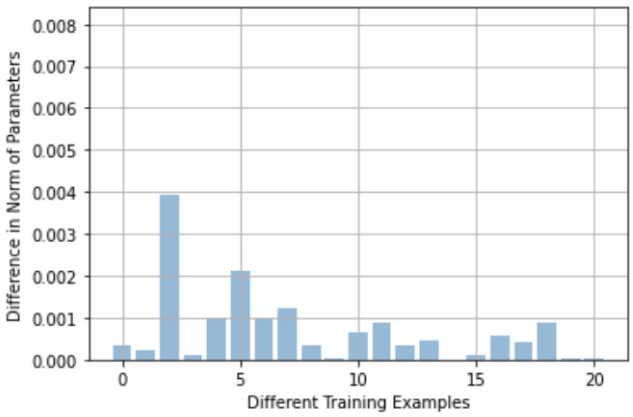

Figure 14: Norm of difference in parameters obtained by training from scratch vs. re-training from optimal parameters

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 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.

13 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.

14 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.

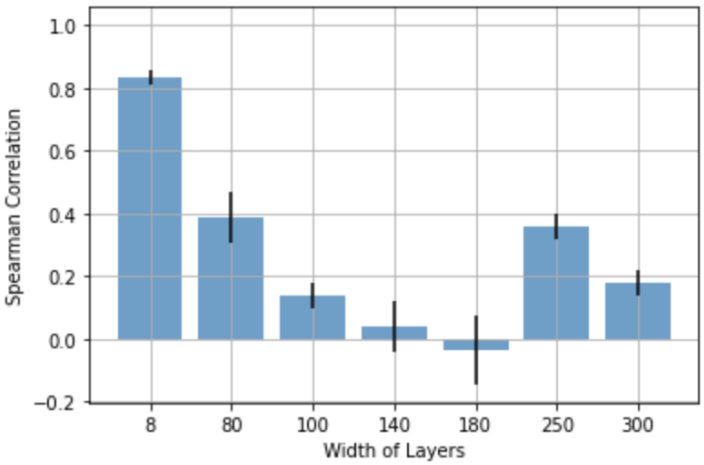

Figure 15: Width vs. Spearman Correlation for a one-layered network

REFERENCES

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

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

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

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.

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`.

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.

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, 2019. URL `http://arxiv.org/abs/1905.13289`.

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.

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.

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

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.

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.

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