Neural Networks 162 (2023) 581–588

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Revisiting the fragility of influence functions

Jacob R. Epifano [a,*], Ravi P. Ramachandran [a], Aaron J. Masino [b], Ghulam Rasool [c]

[a] Rowan University, Department of Electrical and Computer Engineering, 201 Mullica Hill Rd, Glassboro, 08028, NJ, USA
[b] University of Pennsylvania Perelman School of Medicine, Department of Biostatistics, Epidemiology, Informatics, 423 Guardian Drive, Philadelphia, 19104, PA, USA
[c] Moffitt Cancer Center, Department of Machine Learning, 12902 USF Magnolia Drive, Tampa, 33612, FL, USA

ARTICLE INFO

Article history:
Received 6 April 2022
Received in revised form 24 February 2023
Accepted 21 March 2023
Available online 24 March 2023

Dataset link: https://github.com/jrepifano/xai_is_fragile

Keywords:
Machine learning
Supervised learning
Deep learning
Explainable AI
Influence functions
Bayesian neural networks

ABSTRACT

In the last few years, many works have tried to explain the predictions of deep learning models. Few methods, however, have been proposed to verify the accuracy or faithfulness of these explanations. Recently, influence functions, which is a method that approximates the effect that leave-one-out training has on the loss function, has been shown to be fragile. The proposed reason for their fragility remains unclear. Although previous work suggests the use of regularization to increase robustness, this does not hold in all cases. In this work, we seek to investigate the experiments performed in the prior work in an effort to understand the underlying mechanisms of influence function fragility. First, we verify influence functions using procedures from the literature under conditions where the convexity assumptions of influence functions are met. Then, we relax these assumptions and study the effects of non-convexity by using deeper models and more complex datasets. Here, we analyze the key metrics and procedures that are used to validate influence functions. Our results indicate that the validation procedures may cause the observed fragility.

1. Introduction

Due to the black-box nature of Deep Neural Networks (DNNs), explaining the predictions of these models remains a challenging problem. Several techniques for addressing this challenge have been proposed such as saliency maps (Simonyan, Vedaldi, & Zisserman, 2014), influence functions (Koh & Liang, 2017), concept activation vectors (Kim et al., 2018), and activation atlases (Carter, Armstrong, Schubert, Johnson, & Olah, 2019). These techniques are not without problems. The fragility of these methods have been well studied, but few works have tried to understand where these methods break down (Basu, Pope, & Feizi, 2020; Ghorbani, Abid, & Zou, 2019).[1]

Influence functions were originally proposed to diagnose and debug linear models by predicting the parameter or loss change due to removing a training instance (Cook & Weisberg, 1982). Their extension to deep learning models, however, did not occur until recently (Koh & Liang, 2017). Influence functions and their applications have been well studied since their reemergence and have since been adopted as a mainstream tool for the interpretation of deep models in a variety of data modalities (Cohen, Sapiro, & Giryes, 2020; Guo, Rajani, Hase, Bansal, & Xiong, 2020; Han, Wallace, & Tsvetkov, 2020; Lee, Park, Pham, & Yoo, 2020), including high-risk areas such as mortality prediction for patients in the Intensive Care Unit (Epifano, Ramachandran, Patel, & Rasool, 2020). Due to the diversity of the use cases for influence functions, understanding their limitations is imperative if they are to be used to explain model behavior. Without key validation procedures, we run the risk of providing misleading or incorrect information to the model users.

To validate these methods, we must first agree on a metric to rate explanations. Spearman correlation between the approximate and true loss differences has been used as a metric to determine the accuracy of influence estimates. The approximate loss differences are given by the influence functions and the true loss differences are obtained by retraining an already trained network after removing a specific training sample (Koh & Liang, 2017). Recent works have used this metric to study the effects that increases in model and dataset size have on the influence functions. It has been found that influence functions are extremely sensitive to these increases (Basu, Pope, & Feizi, 2020).

It is well known that increases in model and dataset size affect the curvature of the loss function (Alain, Roux, & Manzagol, 2019; Ghorbani, Krishnan, & Xiao, 2019; Sagun, Bottou, & LeCun, 2016; Sagun, Evci, Guney, Dauphin, & Bottou, 2017). Convexity of the loss function is a critical assumption of influence functions as they heavily rely on the approximation of the inverse

* Corresponding author.
E-mail address: jrepifano@gmail.com (J.R. Epifano).

[1] Code and Raw output files are available at the following url: https://github.com/jrepifano/xai_is_fragile.

https://doi.org/10.1016/j.neunet.2023.03.029
0893-6080/

Hessian-vector product. The stochastic estimation algorithm used to compute the inverse Hessian-vector product assumes that the Hessian is positive semidefinite (Agarwal, Bullins, & Hazan, 2017; Pearlmutter, 1994). Preliminary work has been done to try to remedy these problems via higher-order approximations (Koh, Ang, Teo, & Liang, 2019) and group influences (Basu, You, & Feizi, 2020), i.e., computing loss differences for more than one training instance at a time.

When discussing fragility, we must look at the whole system, not just the method in question. Deep neural networks have been shown to be sensitive to small perturbations via the weight initialization or by the order in which the data is given to the model (Madhyastha & Jain, 2019; Smilkov, Thorat, Kim, Viégas, & Wattenberg, 2017). The problem lies in the noisy nature of the gradients. This problem has been linked to poor model convergence as well as explainability and attempts to address it include Gaussian averaging (Smilkov et al., 2017), Stochastic Weight Averaging (SWA) (Izmailov, Podoprikhin, Garipov, Vetrov, & Wilson, 2018; Madhyastha & Jain, 2019) and model averaging through Bayesian Inference (Blundell, Cornebise, Kavukcuoglu, & Wierstra, 2015).

In this paper, we examine the cases where influence functions seemingly fail, i.e. have low Spearman correlation between approximate and true loss differences. We obtain the operands for the correlation using the retraining procedure introduced in Koh and Liang (2017), where the approximate loss differences are computed for the test point with the maximal loss using influence functions. Each training point is removed one at a time and the neural network is retrained from the optimal parameters until convergence in order to obtain the true difference in the loss function values. We determined that this training procedure is not valid for most applications of deep learning and present evidence for these cases.

2. Background

2.1. Influence functions

Consider a standard classification problem where a label y is predicted for each feature vector x. Let $z_i = (x_i, y_i)$, where $i = 1, 2, \ldots, N$, for N instances in the dataset. It is assumed that we have a trained model where θ represents the trained network parameters. Our loss function can be written as $L(z, \theta) = \sum_{i=1}^{N} L(z_i, \theta)$. Our optimal model parameters are the set of parameters that minimize the loss: $\hat{\theta} = \arg\min_{\theta \in \Theta} \sum_{i=1}^{N} L(z_i, \theta)$ (Koh & Liang, 2017). Koh and Liang (2017) offers insight on how to approximate the effect that removing a training point z has on the parameters θ. We compute the parameter change with z upweighted by a small value, ϵ. Using this upweighting scheme we obtain a new set of parameters, $\hat{\theta}_{\epsilon,z} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(z_i, \theta) + \epsilon L(z, \theta)$ (Koh & Liang, 2017). Cook and Weisberg (1982) has shown that as ϵ approaches zero the influence of z on the parameters is:

$$\mathcal{I}_{\text{up,params}}(z) = \left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}), \tag{1}$$

where $H_{\hat{\theta}} = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta^2 L(z_i, \hat{\theta})$ is the Hessian. If we let $\epsilon = -\frac{1}{n}$, then we can approximate the parameter change as $\hat{\theta}_{-z} - \hat{\theta} \approx -\frac{1}{n} \mathcal{I}_{\text{up,params}}(z)$ (Koh & Liang, 2017). To study the effect of removing a training point on a test point z_{test} on the loss function, we apply the chain rule: Koh and Liang (2017):

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}) \tag{2}$$

2.2. Influence function guidance

Ideally, a model must be trained until the optimal parameters $\hat{\theta}$ are obtained in order to compute the influence functions. For a single test instance, z_{test}, we would then compute the inverse Hessian-vector product, $\nabla_\theta L(z_{\text{test}}, \hat{\theta})^T H_{\hat{\theta}}^{-1}$, using stochastic estimation (Pearlmutter, 1994). In reality, due to non-linearities in our networks, our objective function may become non-convex and we obtain our parameters $\tilde{\theta}$ via SGD, where $\tilde{\theta} \neq \hat{\theta}$. In this case, the Hessian may have negative eigenvalues which would cause the stochastic estimation algorithm to not converge. To address this, we adopt a regularization scheme similar to L2 regularization discussed by Koh and Liang (2017). We regularize the computation of the Hessian-vector product using a damping term of $\lambda = 0.01$. We can then compute the gradient of the loss as $\nabla_\theta L(z, \hat{\theta})$. The inner product of the Hessian-vector product and the gradient of the training instance results in a scalar value that tells us the approximate change in loss to expect on z_{test} if we were to remove the training instance z. Note that we compute the gradient of the loss function with respect to **only** the parameters of the last layer (Koh & Liang, 2017).

2.3. Non-convexity and eigenvalues of the Hessian

Due to the importance of the Hessian in the computation of influence functions, the convexity of the loss function and its effects on the Hessian are important. Recall that influence functions assume the Hessian is positive definite such that it is invertible. Koh and Liang (2017) have shown that even with negative Hessian eigenvalues it is still possible to obtain good influence estimates. It is understood that large overparameterized networks affect the convexity of the loss function (Ghorbani, Krishnan, & Xiao, 2019; Sagun et al., 2017), which we observe via the eigenvalues of the Hessian. Basu, Pope, and Feizi (2020) have shown that larger eigenvalues are correlated with decreases in the Spearman correlation metric when network depth and width are increased. This contradicts the literature where the long tail of the Hessian Eigen Spectral Density (ESD) has been well studied for large DNNs and it has been shown that the largest eigenvalue does not tend to increase as width of the network increases (Sagun et al., 2016). In this paper, we utilize a method developed by Yao, Gholami, Keutzer, and Mahoney (2020) to compute the eigenvalues of the Hessian in an effort to quantify the effect if any, of non-convexity and non-convergence on Influence functions.

2.4. Bayesian deep neural networks

The current state of the art for influence functions, suggests that by applying L2 regularization to our networks during training, we can reduce the negative effects that are associated with overparameterization (Basu, Pope, & Feizi, 2020). Variational Bayesian Learning has been shown to result in superior regularization, better model averaging and built-in uncertainty prediction (Blundell et al., 2015). We select this method specifically for its regularization strength.

In this subsection, we present a modified version of the Extended Variational Inference model proposed by Dera, Rasool, and Bouaynaya (2019). We assume the covariance is zero and only propagate variance for each parameter.

For a given classification problem, we want to estimate the posterior distribution of the weights given the data, i.e., $p(\theta|D)$. This, however, is intractable due to the high dimensionality of the parameter space. We can approximate the true posterior by defining a variational distribution $q(\theta)$, which is assumed to be Gaussian. Since we want the variational distribution to be

close to the true posterior, we minimize the Kullback–Leibler (KL) divergence.

$$\arg\min_{\theta} \mathrm{KL}(q(\theta) \parallel p(\theta|D)) = -E_{q(\theta)}[\log p(D|\theta)] + \mathrm{KL}(q(\theta) \parallel p(\theta)) \tag{3}$$

To quantify the loss for the variational learning approach, we use the Evidence Lower Bound (ELBO), $\mathcal{L}(\theta, D)$ which consists of two parts, namely, the expected log-likelihood of the training data given the weights and a regularization term,

$$\mathcal{L}(\theta, D) = E_{q(\theta)}[\log p(D|\theta)] - \mathrm{KL}[q(\theta)|p(\theta)] \tag{4}$$

where θ represents the weights of the network and D represents the data label pairs. Continuing the derivation gives

$$\begin{aligned}\mathcal{L}(\theta, D) = &\frac{1}{N}\sum_{i=1}^{N}\log\left(\prod\sigma_{\hat{y}}^2\right) \\ &+ \frac{1}{2}\sum_{i=1}^{N}((\hat{y}-\mu_{\hat{y}})^2(\sigma_{\hat{y}}^2)^{-1}) \\ &+ \frac{1}{2}\sum_{n=1}^{l}(j\log\sigma_l^2 - \|\mu_l\|_F^2 - j\sigma_l^2)\end{aligned} \tag{5}$$

where: $\hat{y}$ is the label, $\mu_{\hat{y}}$ is the output mean, $\sigma_{\hat{y}}^2$ is the output variance, l is the number of hidden layers and j is the number of nodes in that layer.

For a neural network, we define the equations to propagate the first two moments. First, we assume the input x is deterministic. The weights and biases of the first layer, w are assumed to be Gaussian. If the incoming input is deterministic (input layer), then the output, z is:

$$\mu_z = \mu_w^T x + \mu_{w_b} \tag{6}$$

$$\sigma_z^2 = x^2[\sigma_w^2]^T + \sigma_{w_b}^2 \tag{7}$$

To propagate the moments through an arbitrary element-wise non-linear function, f (e.g., ReLU, SELU), we use a first order Taylor-series approximation, to get an output, a:

$$a = f(z) \tag{8}$$

$$\mu_a \approx f(\mu_z) \tag{9}$$

$$\sigma_a^2 = \sigma_z^2 \odot (f'(\mu_z))^2 \tag{10}$$

If the incoming input, a is a random variable (for the intermediate layers), then the first two moments, $\tilde{y}$ are:

$$\mu_{\tilde{y}} = \mu_v^T\mu_a + \mu_{v_b} \tag{11}$$

$$\sigma_{\hat{y}}^2 = \sigma_v^2[\sigma_a^2]^T + \mu_v^2[\sigma_a^2]^T + \mu_a^2[\sigma_v^2]^T + \sigma_{v_b}^2 \tag{12}$$

To propagate through a non-linear function that is not element-wise, e.g. softmax, we use a first order Taylor-series approximation (Simon, 2006). The output, $\hat{y}$ of the non-linear function, g is:

$$\mu_{\hat{y}} \approx g(\mu_{\tilde{y}}) \tag{13}$$

$$\begin{aligned}\sigma_{\hat{y}}^2 &\approx \mathbf{J}_g^2 \odot \sigma_{\tilde{y}}^2 \\ &\approx (\mu_{\hat{y}}(1-\mu_{\hat{y}}))^2 \odot \sigma_{\tilde{y}}^2\end{aligned} \tag{14}$$

where $\mathbf{J}_g$ is the Jacobian of g.

3. Experiments

3.1. Iris dataset

To study the effect of random initialization on influence function estimates, we reproduced an experiment from Basu, Pope, and Feizi (2020) using the Iris dataset. The Iris dataset consists of 150 instances with 4 features and 3 classes. The decision to use this dataset as a benchmark is due to its simplicity. To make our models more robust to random initialization, we considered weight decay as well as Stochastic Weight Averaging (SWA) and Bayesian Neural Networks (BNNs) as novel additions to this experiment (Dera et al., 2019; Izmailov et al., 2018; Madhyastha & Jain, 2019).

This experiment was repeated for two types of DNNs: (1) DNNs with constant width (number of nodes in a hidden layer) and variable depth (number of hidden layers), and (2) DNNs with constant depth and variable width. In the experiments with variable depth, the number of nodes per hidden layer was held constant at 5 as in Basu, Pope, and Feizi (2020). In the variable width experiments, the depth of the network was held constant at 1, i.e., one hidden layer only. We used the Adam optimizer with an initial learning rate of 0.001 as in Basu, Pope, and Feizi (2020). A learning rate scheduler was used to decrease the learning rate by a factor of 10 if the loss did not decrease for 100 epochs. For the experiments with weight decay, we used a constant value of 0.005 as in Basu, Pope, and Feizi (2020). Each experiment was repeated 50 times.

Koh and Liang (2017) showed that fine-tuning a trained DNN from the optimal parameters is approximately equal to retraining the same network with a training instance removed. Therefore, to obtain the true differences in loss when removing a test point, we replicate the training procedure outlined by Basu, Pope, and Feizi (2020). The models are initially trained for 60k epochs of full-batch gradient descent instead of SGD. The training instances are then sorted by their loss and the 40 training instances with the maximal loss are identified. We then fine-tune only the top layer for 7.5k epochs when individually removing each of the training points with the highest loss. Finally, we compute the influence function estimates for those training instances with respect to the test instance with the highest loss. The Spearman correlation between the true and approximate differences in loss are then computed. The eigenvalues of the Hessian for each network were computed via power iteration using the PyHessian Python package (Yao et al., 2020).

3.2. MNIST and CIFAR10

We drastically increase the model and dataset size to study the performance of influence functions in non-convex settings. Similar to the experiment described in Basu, Pope, and Feizi (2020), we chose to look at a small fully connected network, LeNet, and VGG13. Each model was trained in a similar manner as our previous experiment. The Adam optimizer was used with an initial learning rate of 0.001 and weight decay of 0.001. The learning rate was reduced by a factor of 10 if the loss did not decrease after 2 epochs. The test instance with the maximal loss was used to compute the influence functions and influence functions were computed for all training instances. We deviate from our previous experiment when choosing the training instances to remove and retrain. The true loss difference was computed for the top 40 most influential training points (highest absolute value) using the re-train from optimal approximation. The Spearman correlation between the true and estimated differences in loss was computed.

3.3. Statistical analysis

We use one-way analysis of variance (ANOVA) to compare various dependent variables and establish statistical significance in various experiments described above.

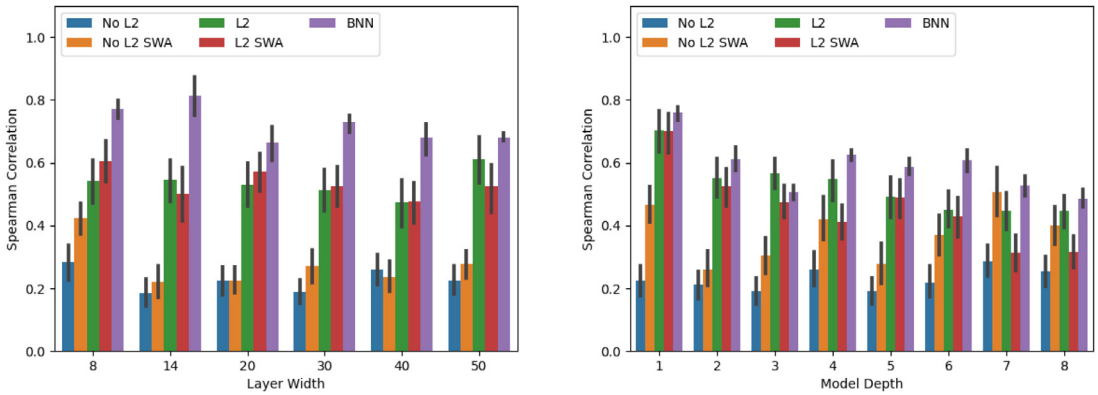

Fig. 1. Influence function performance evaluation on Iris dataset. **Left**: constant depth experiment. **Right**: constant width experiment. Spearman correlation between the true and approximate loss differences is on the y-axis (higher is better). The error bars represent the 95% intervals obtained by repeating the experiment 50 times. **Blue**: training without weight decay. **Orange**: training without weight decay but with Stochastic Weight Averaging (SWA). **Green**: training with weight decay. **Red**: training with weight decay and SWA. **Purple**: training with BNN. We observe that the influence functions that come from BNN are significantly better than the rest of the methods in almost all cases. SWA has significant performance increases without the presence of regularization but with regularization has little effect and was removed for clarity. Statistical testing using one-way ANOVA revealed no significant difference ($p > .05$) between correlation values for any of the model types.

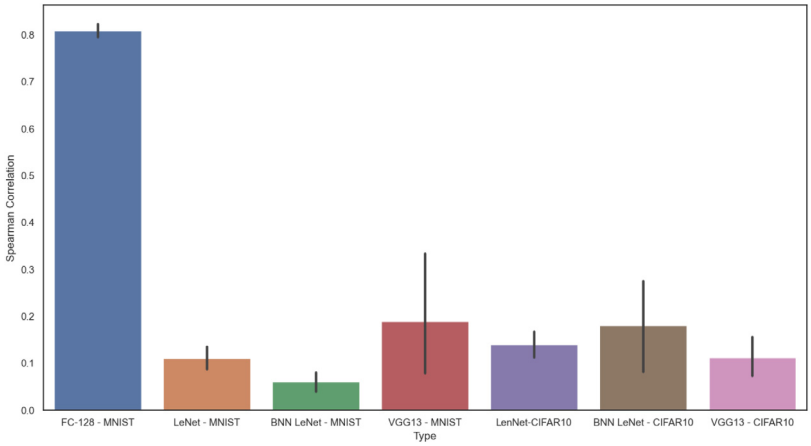

Fig. 2. Spearman correlation between the true and approximate loss differences is on the y-axis (higher is better). The error bars represent the 95% intervals obtained by repeating the experiment 10 times. We observe that the Spearman correlation is only significant in the small fully connected model on the MNIST dataset. As the number of parameters increases, the influence function performance falls off sharply, which was expected. There are no significant differences between VDP and the other large models.

4. Results and discussion

4.1. Effect of model size on the influence function estimates

In Fig. 1, we present the Spearman's rank correlation coefficient (ρ) between the true and estimated loss differences for the Iris dataset for a variety of model types and sizes. We present four different types of models, including a model with L2 regularization, a model without L2 regularization, a model with SWA, and a BNN. The figure presents models trained using an increasing number of neurons in one layer (Fig. 1-**Left**) and increasing number of layers with fixed number of neurons in each layer (Fig. 1-**Right**). The true loss difference is found using the re-training strategy and the estimated loss difference is found using Eq. (2). The error bars represent the 95% confidence intervals obtained by repeating the experiment 50 times. It is evident from both sub-figures that for any type of model (L2, No-L2, SWA, and BNN), there is a minimal effect of increasing number of neurons or number of layers on the quality of estimate (of the influence of a training point on the selected test data point) provided by influence functions (using Eq. (2)). A statistical analysis performed using ANOVA did not reveal any significant effect of number of neurons or layers on the Spearman correlation ($p > 0.5$ for all cases). Previously, Basu, Pope, and Feizi (2020) had reported increasing model size (depth and width) degrades influence function estimates. We believe that the discrepancy between the reported results is linked to statistical rigor as no statistical tests or analyses were reported by Basu, Pope, and Feizi (2020) to establish the effect of model size on the quality of estimates produced by influence functions.

We also observe that the estimates provided by influence function are more accurate for models with regularization, as shown in Fig. 1, in particular, the Bayesian models (BNNs) outperform all other methods. We consider that the observed behavior is linked to (1) the "ensemble" or "average" effect introduced by Bayesian approaches in the model training, and (2) the type of regularization present in the ELBO loss function which has been shown to give these models superior self-compression properties (Carannante, Dera, Rasool, & Bouaynaya, 2020). This performance increase, however does not seem to carry over to our experiments with larger datasets (Fig. 2), where all models were trained with regularization. This is congruent with the

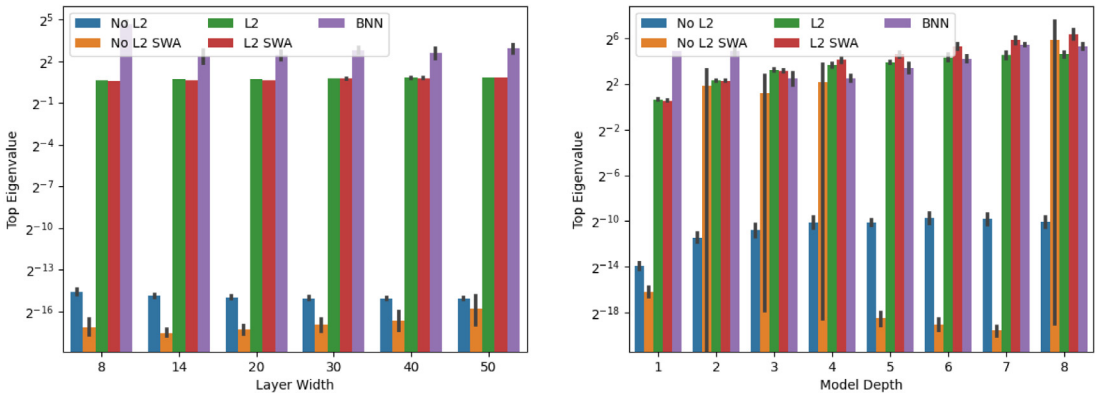

Fig. 3. Influence Function performance evaluation on Iris dataset. **Left**: constant depth experiment. **Right**: constant width experiment. Largest Eigenvalue is on the y-axis. The error bars represent the 95% intervals obtained by repeating the experiment 50 times. **Blue**: training without weight decay. **Orange**: training without weight decay but with Stochastic Weight Averaging (SWA). **Green**: training with weight decay. **Red**: training with weight decay and SWA. **Purple**: training with BNN. This figure shows little correlation between curvature of the loss function. Statistical testing using one-way ANOVA showed no difference in the top eigenvalue for any model in the width or depth experiments.

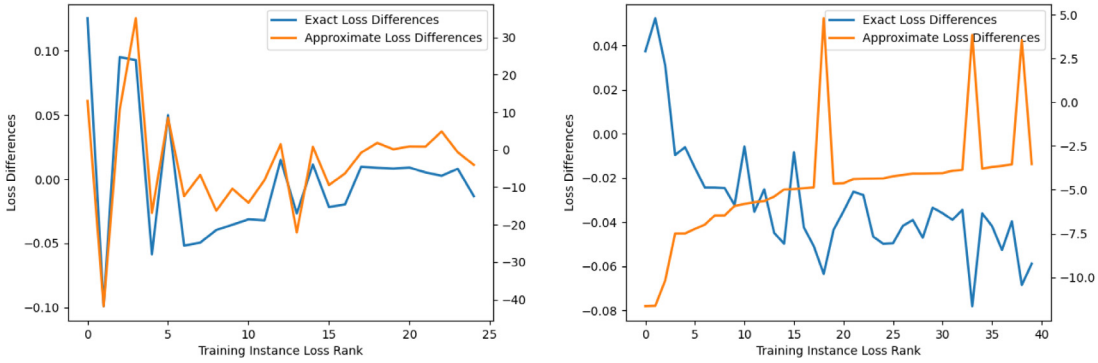

Fig. 4. Example of miss-relation. **Left:** Depth 1 width 5 network with weight decay on Iris dataset. **Right:** Small FC network with 128 nodes on MNIST dataset. We show that when the loss function is convex, our estimates match the true loss differences (Left, scale does not matter). When the loss function is non-convex there is significant deviation from the true loss differences. Both left and right receive an absolute Spearman correlation of 0.85, which results in a miss-relation for the right graph.

results obtained by Basu, Pope, and Feizi (2020) on the same datasets.

4.1.1. The largest eigen value

In Fig. 3 we observe the same trend that Basu, Pope, and Feizi (2020) found in the Iris experiment, e.g., the eigenvalues of the Hessian increase with model width and depth (ANOVA $p < 0.05$). We do not however, relate the supposed decrease in influence function estimates to the increasing top eigenvalue as a proxy for curvature of the loss function given that our statistical results from Fig. 1 show that there are no significant differences between model sizes and influence function performance. Given that Koh and Liang (2017) have shown that even when most assumptions about convexity of the loss function have been broken, i.e., the optimal parameters have not been obtained ($\tilde{\theta} \neq \hat{\theta}$) and the Hessian has negative eigenvalues (Hessian not PD), we can still obtain "good" influence estimates. We postulate that the problem lies with the methods that have been used to evaluate influence functions.

4.2. The (In-)validity of Spearman's rank correlation coefficient

Spearman's rank correlation coefficient is an established metric for determining the accuracy of influence function estimates (Basu, Pope, & Feizi, 2020; Basu, You, & Feizi, 2020; Koh & Liang, 2017). We note that the output of Eq. (2) is the difference in the loss function value for the test instance if the training instance is removed. This loss difference can be positive or negative. For a training instance to be influential, it needs to have a large magnitude.

In Fig. 4, we provide an example where Spearman's correlation coefficient is unable to capture the underlying relationship between the true loss difference and estimated loss difference, where the estimate is being calculated using Eq. (2). The horizontal axis in both sub-figures (Fig. 4 **Left** and **Right**) corresponds to the rank of the training point, where the rank is determined by the approximate loss difference. Thus, we should expect to see the exact loss differences (blue points) move from a large magnitude towards zero as we move from left to right on the horizontal axis. In Fig. 4(**Left**), the estimated and true values (after ignoring the scale) are close to each other. In Fig. 4(**Right**), the values of true and estimated loss differences are significantly different from each other. However, the value of Spearman's correlation coefficient for both cases is approximately 0.85.

We consider that since the relationship between the estimated and true loss function difference values may not always be **monotonically** decreasing or increasing, the Spearman's correlation can lead to misleading results.

4.3. Re-training for optimal parameters

To compute the Spearman's correlation coefficient, we need to know the true difference in the loss function value. This requires retraining models for every training instance that we want to

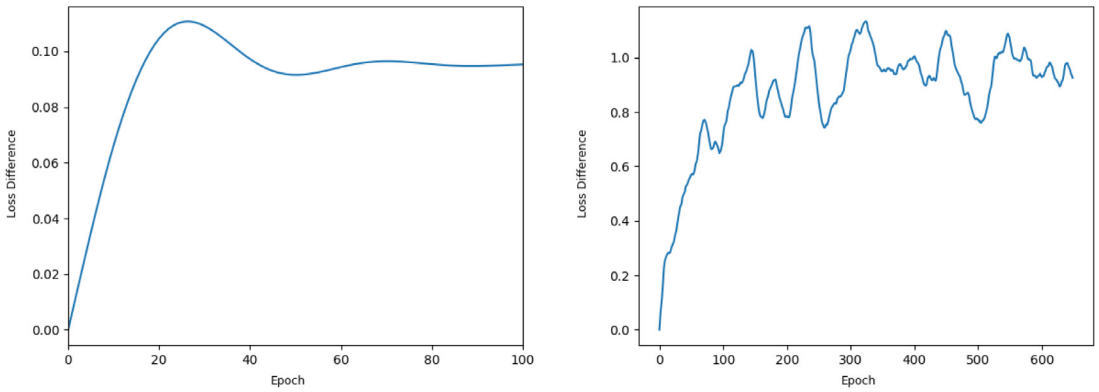

Fig. 5. The loss trajectories followed during re-raining loss. **Left** Depth 1 width 5 network with weight decay on Iris dataset. **Right** Small FC network with 128 nodes on MNIST dataset. Here we show the test loss as a function of re-training in convex and non-convex settings. The sharp jumps of the right plot indicate that the model leaves the minima that it settled in previously, which breaks the assumptions of the influence functions.

analyze. This is a very costly operation in time. The re-training from optimal parameters has been shown to be an approximately equivalent alternative to retraining from scratch (Koh & Liang, 2017). Previous works have not proven that this approximation is valid for large datasets (Basu, Pope, & Feizi, 2020; Koh & Liang, 2017). It has been well established that increasing model and dataset complexity increases the largest eigenvalue of the Hessian of the loss function (Ghorbani, Krishnan, & Xiao, 2019; Sagun et al., 2017). While we have demonstrated that the increasing curvature does not affect estimates made by influence functions with small datasets and models, with large datasets and models the extreme curvature of the loss function makes us question the validity of the re-training approximation (Zhang, Wang, Xu, & Grosse, 2018). To study this, we looked at the loss of the test instance at each epoch during re-training in both the Iris and MNIST experiments. In Fig. 5, the test loss difference is plotted against epochs on the horizontal axis. We note significant differences in the trajectories followed by the gradient descent algorithm for two cases (Iris - Fig. 5 left and MNIST 5 right). The Iris model has a well damped convergence whereas the MNIST model is underdamped and does not seem to converge as smoothly as did Iris.

4.4. The effect of large networks

We consider large neural networks as having more parameters, more non-linear operations owing to their depth, and reluctantly requiring large datasets for training. We note that originally Cook and Weisberg derived influence functions for regression models, which can be considered as neural networks with one layer and mean-square error loss function (Cook & Weisberg, 1982). Recently, Koh and Liang (2017) extended the idea of using influence functions in deep neural networks by treating all but the last layer of the deep neural network as a feature extractor. The influence functions were computed with respect to only the last layer. This practice seems to work in some cases and produce promising results (Koh & Liang, 2017). However, it does not account for the large dimensionality of the final layer of modern neural networks. This proves to be a problem when these large parameter matrices become ill-conditioned (Belsley, Kuh, & Welsch, 2005). This problem was captured by Basu, Pope, and Feizi (2020) in their analysis of large datasets like CIFAR-100 and ImageNet where true differences in losses from removing training instances resulted in very noisy results.

Large neural networks may have more layers (depth) and/or more operations per layer (width). This results in increasing the number of non-linear operations which are performed on the data for calculating the loss function. The most popular implementation of influence functions, as defined by Koh and Liang (2017), relies on only a first-order Taylor series approximation to efficiently compute influence (Eq. (A.6)). We argue that the increasing number of non-linear operations strongly affects the convexity assumption of loss function $R(\theta)$ (Eq. (A.1)) as used in the mathematical relationships derived for influence functions (Eq. (A.2)). There is evidence suggesting that adding the second term of the Taylor series in the influence function approximation improves the estimates (Basu, You, & Feizi, 2020; Koh et al., 2019).

Finally, large networks typically go hand-in-hand with large datasets. From Eq. (1), it is evident that removing a training instance is equivalent to up-weighting it by $\epsilon = -\frac{1}{n}$. In the Iris dataset, $|\epsilon| \approx 6.6e-3$ compared to MNIST where $|\epsilon| \approx 1.6e-5$. Any larger datasets will lead to smaller epsilon, that is:

$$\hat{\boldsymbol{\theta}}_{-z} - \hat{\boldsymbol{\theta}} \approx 0 \text{ or } \hat{\boldsymbol{\theta}}_{-z} \approx \hat{\boldsymbol{\theta}}. \tag{15}$$

In other words, owing to the large dataset, the influence of a single training point on a test sample is asymptotically approaching zero. Perhaps the first-order Taylor series approximation of the influence functions does not provide enough resolution to predict loss differences when predicting on only one training instance. If one wanted to use influence functions in large datasets like CIFAR-100 and ImageNet, one would have to turn to higher-order approximations of influence functions as well as group influences. In Basu, You, and Feizi (2020), promising results were obtained by examining the effect of a second-order approximation as well as group influence. These solutions of course have an associated cost. Adding a second-order term increases the cost and complexity of the analysis. Optimal group selection is also a non-trivial and expensive operation.

While there is theoretical evidence to suggest that the first-order implementation of influence functions is fragile, due to the difficulty of finding robust ways to empirically evaluate them in difficult settings, their supposed fragility remains unclear.

5. Limitations

While we have established that the procedures used to measure the accuracy of influence functions are flawed in multiple ways, we have not been able to ascertain exactly where or why these procedures break down. It appears that the answer lies with increasing model and dataset size. To precisely define the boundaries on where violating the approximations that Koh and Liang (2017) have established is valid, we would need to exhaustively search the space of increasing complexity of the model and dataset.

6. Conclusion

Validating the performance of explanation methods is a key area of deep learning that has not been well studied. In particular, the validation of influence functions in deep learning has been an area of interest. In this work, we analyzed several experiments from the recent literature in order to understand the fragility of influence functions. We obtained results that conflict with those of our peers, which we attribute to the repetition in our experimental design as well as the misuse of the Spearman correlation metric and retraining procedure.

While we have demonstrated that the methods we use to measure the accuracy of influence functions are flawed, we must not conclude that influence functions are uninformative. Due to the flaw in validation methodology, we do not have any evidence to support the claim that the explanations provided by influence functions are not accurate or faithful to the original model. Future efforts should be focused on developing robust validation frameworks for explainable methods in order to foster user-model trust.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code and Raw output files are available at the following url: https://github.com/jrepifano/xai_is_fragile.

Acknowledgments

Jacob R. Epifano is supported by US Department of Education GAANN award P200A180055. Ghulam Rasool was partly supported by National Science Foundation OAC-2008690.

Appendix. Influence function derivation

This derivation was taken directly from Koh and Liang (2017) and has been reproduced below:

Influence functions are considered one of the classic technique from robust statistics that can quantify the change in model parameters attributed to up-weighting a training point by an infinitesimal amount. In the following, we derive mathematical relationships for influence functions, examine their underlying assumptions, and attempt to explain these in the context of large neural networks. We start by defining an optimization problem where $\hat{\theta}$ minimizes the empirical risk following Koh and Liang (2017):

$$R(\theta) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} L(z_i, \theta). \tag{A.1}$$

A fundamental assumption for influence functions is that R is twice-differentiable and **strongly** convex in θ. That is, the Hessian, as defined by:

$$H_{\hat{\theta}} \stackrel{\text{def}}{=} \nabla^2 R(\hat{\theta}) = \frac{1}{n}\nabla^2_{\theta} L(z_i, \hat{\theta}), \tag{A.2}$$

exists and is positive definite. The convexity assumption guarantees the existence of $H^{-1}_{\hat{\theta}}$. Recall, that we approximate the removal of a training point by up-weighting the parameters by a small quantity $\epsilon \approx -\frac{1}{n}$, where n is the total number of training data points. The perturbed parameters, $\hat{\theta}_{\epsilon,z}$ can be written as (Koh & Liang, 2017):

$$\hat{\theta}_{\epsilon,z} = \arg\min_{\theta\in\Theta} \left[R(\theta) + \epsilon L(z,\theta)\right]. \tag{A.3}$$

The total parameter change by up-weighting a training example can be defined as $\Delta_\epsilon = \hat{\theta}_{\epsilon,z} - \hat{\theta}$. Differentiating with respect to ϵ and noting that $\hat{\theta}$ does not depend on ϵ, we can write:

$$\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon} = \frac{d\Delta_\epsilon}{d\epsilon}. \tag{A.4}$$

We note that for the optimal parameters $\hat{\theta}_{\epsilon,z}$, we can rewrite Eq. (A.3) as:

$$0 = \nabla R(\hat{\theta}_{\epsilon,z}) + \epsilon \nabla L(z, \hat{\theta}_{\epsilon,z}). \tag{A.5}$$

Since $\hat{\theta}_{\epsilon,z} \to \hat{\theta}$ as $\epsilon \to 0$, the **first-order** Taylor expansion of the right-hand side produces:

$$0 \approx \left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z,\hat{\theta})\right] + \left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z,\hat{\theta})\right]\Delta_\epsilon. \tag{A.6}$$

Solving for Δ_ϵ,

$$\Delta_\epsilon = -\left[\nabla^2 R(\hat{\theta}) + \epsilon \nabla^2 L(z,\hat{\theta})\right]^{-1}\left[\nabla R(\hat{\theta}) + \epsilon \nabla L(z,\hat{\theta})\right] \tag{A.7}$$

Since $\hat{\theta}$ minimizes R, then $\nabla R(\hat{\theta}) = 0$. Neglecting higher order ϵ terms,

$$\Delta_\epsilon \approx -\nabla^2 R(\hat{\theta})^{-1} \nabla L(z,\hat{\theta})\epsilon. \tag{A.8}$$

When we substitute Eqs. (A.2) and (A.4), we have:

$$\left.\frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\right|_{\epsilon=0} = -H^{-1}_{\hat{\theta}} \nabla L(z,\hat{\theta}) \stackrel{\text{def}}{=} \mathcal{I}_{\text{up,params}}(z). \tag{A.9}$$

References

Agarwal, N., Bullins, B., & Hazan, E. (2017). Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research, 18*(1), 4148–4187.

Alain, G., Roux, N. L., & Manzagol, P.-A. (2019). Negative eigenvalues of the hessian in deep neural networks. arXiv preprint arXiv:1902.02366.

Basu, S., Pope, P., & Feizi, S. (2020). Influence functions in deep learning are fragile. In *International conference on learning representations*.

Basu, S., You, X., & Feizi, S. (2020). On second-order group influence functions for black-box predictions. In *International conference on machine learning* (pp. 715–724). PMLR.

Belsley, D. A., Kuh, E., & Welsch, R. E. (2005). *Regression diagnostics: Identifying influential data and sources of collinearity*. John Wiley & Sons.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning* (pp. 1613–1622). PMLR.

Carannante, G., Dera, D., Rasool, G., & Bouaynaya, N. C. (2020). Self-compression in Bayesian neural networks. In *2020 IEEE 30th international workshop on machine learning for signal processing* (pp. 1–6). IEEE.

Carter, S., Armstrong, Z., Schubert, L., Johnson, I., & Olah, C. (2019). Activation atlas. *Distill, 4*(3), Article e15.

Cohen, G., Sapiro, G., & Giryes, R. (2020). Detecting adversarial samples using influence functions and nearest neighbors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14453–14462).

Cook, R. D., & Weisberg, S. (1982). *Residuals and influence in regression*. New York: Chapman and Hall.

Dera, D., Rasool, G., & Bouaynaya, N. (2019). Extended variational inference for propagating uncertainty in convolutional neural networks. In *2019 IEEE 29th international workshop on machine learning for signal processing* (pp. 1–6). IEEE.

Epifano, J. R., Ramachandran, R. P., Patel, S., & Rasool, G. (2020). Towards an explainable mortality prediction model. In *2020 IEEE 30th international workshop on machine learning for signal processing* (pp. 1–6). IEEE.

Ghorbani, A., Abid, A., & Zou, J. (2019). Interpretation of neural networks is fragile. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33* (pp. 3681–3688).

Ghorbani, B., Krishnan, S., & Xiao, Y. (2019). An investigation into neural net optimization via hessian eigenvalue density. In *International conference on machine learning* (pp. 2232–2241). PMLR.

Guo, H., Rajani, N. F., Hase, P., Bansal, M., & Xiong, C. (2020). Fastif: Scalable influence functions for efficient model interpretation and debugging. arXiv preprint arXiv:2012.15781.

Han, X., Wallace, B. C., & Tsvetkov, Y. (2020). Explaining black box predictions and unveiling data artifacts through influence functions. arXiv preprint arXiv:2005.06676.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018). Averaging weights leads to wider optima and better generalization. arXiv preprint arXiv:1803.05407.

Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al. (2018). Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning* (pp. 2668–2677). PMLR.

Koh, P. W., Ang, K.-S., Teo, H. H., & Liang, P. (2019). On the accuracy of influence functions for measuring group effects. arXiv preprint arXiv:1905.13289.

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

Lee, D., Park, H., Pham, T., & Yoo, C. D. (2020). Learning augmentation network via influence functions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10961–10970).

Madhyastha, P., & Jain, R. (2019). On model stability as a function of random seed. arXiv preprint arXiv:1909.10447.

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural Computation*, *6*(1), 147–160.

Sagun, L., Bottou, L., & LeCun, Y. (2016). Eigenvalues of the hessian in deep learning: Singularity and beyond. arXiv preprint arXiv:1611.07476.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., & Bottou, L. (2017). Empirical analysis of the hessian of over-parametrized neural networks. arXiv preprint arXiv:1706.04454.

Simon, D. (2006). *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *In workshop at international conference on learning representations*. Citeseer.

Smilkov, D., Thorat, N., Kim, B., Viégas, F., & Wattenberg, M. (2017). Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825.

Yao, Z., Gholami, A., Keutzer, K., & Mahoney, M. W. (2020). Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data* (pp. 581–590). IEEE.

Zhang, G., Wang, C., Xu, B., & Grosse, R. (2018). Three mechanisms of weight decay regularization. arXiv preprint arXiv:1810.12281.