Pattern Recognition 121 (2022) 108194

Contents lists available at ScienceDirect

Pattern Recognition

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

Towards robust explanations for deep neural networks

Ann-Kathrin Dombrowski[a], Christopher J. Anders[a,d], Klaus-Robert Müller[a,b,c,d,e,*], Pan Kessel[a,d,*]

[a] Machine Learning Group, Department of Electrical Engineering & Computer Science, Technische Universität Berlin, Marchstr. 23, Berlin 10587, Germany
[b] Department of Artificial Intelligence, Korea University, Anam-dong, Seongbuk-gu, Seoul 02841, Korea
[c] Max Planck Institute for Informatics, Stuhlsatzenhausweg 4, Saarbrücken 66123, Germany
[d] BIFOLD - Berlin Institute for the Foundations of Learning and Data, Technische Universität Berlin, Berlin, Germany
[e] Google Research, Brain team, Berlin, Germany

ARTICLE INFO

Article history:
Received 18 December 2020
Revised 10 June 2021
Accepted 20 July 2021
Available online 29 July 2021

Keywords:
Explanation method
Saliency map
Adversarial attacks
Manipulation
Neural networks,

ABSTRACT

Explanation methods shed light on the decision process of black-box classifiers such as deep neural networks. But their usefulness can be compromised because they are susceptible to manipulations. With this work, we aim to enhance the resilience of explanations. We develop a unified theoretical framework for deriving bounds on the maximal manipulability of a model. Based on these theoretical insights, we present three different techniques to boost robustness against manipulation: training with weight decay, smoothing activation functions, and minimizing the Hessian of the network. Our experimental results confirm the effectiveness of these approaches.

1. Introduction

In recent years, deep neural networks have revolutionized many different areas. Despite their impressive performance, the reasoning behind their decision processes remains difficult to grasp for humans. This can limit their usefulness in applications that require transparency. Explanation methods promise to make neural networks interpretable. In this work we consider explanations of individual predictions that can be given in terms of explanation maps [1–20] which visualize the importance of each input feature for the network's prediction. They give valuable information about relevant features [21–24], help us understand what a model has learned [3,25–27], and identify unwanted behavior or biases in the data [7,28–30].

While explanation methods show promising results in many areas, concerns regarding their reliability exist. Recent work has shown that explanations are sensitive to small perturbations of the input that do not change the classification result [31]. Furthermore, these perturbations can be constructed such that an arbitrary target explanation is closely reproduced and all class scores are approximately unchanged (as opposed to only the classification result), see [32]. An alternative approach leaves the input unchanged but manipulates the model such that it has the same output on the entire data manifold but reproduces an arbitrary target explanation map [33,34]. The former class of methods is often refered to as input manipulations and the latter as model manipulations.

Untrustworthy explanations are evidently problematic for various reasons. For a large number of applications, one is interested in the prediction as well as in the explanation of a phenomenon. Examples include medical and natural science applications. As some explanations are susceptible even to random input perturbations, it seems questionable if much insight can be derived from inspecting such explanations. In a setting where explanations are legally required [35], explanation manipulability obviously raises serious concerns as they cannot be considered trustworthy evidence. An example for this is credit risk assessment: The supplier can obfuscate that a decision was made based on racist, sexist or other discriminating features by manipulating the model [34]. Similarly, attacks from the user side are possible by manipulating the input as they can create the impression that the decision was based on unaccepted features and thus subvert the result.

In this paper, we develop methods to make explanations provably more robust against attacks that manipulate the *input*. To this end, we provide the following key contributions:

- We analyze the difference between the original and the manipulated explanation maps theoretically and provide a unified

[*] Corresponding author.
E-mail addresses: a.dombrowski@tu-berlin.de (A.-K. Dombrowski), anders@tu-berlin.de (C.J. Anders), klaus-robert.mueller@tu-berlin.de (K.-R. Müller), pan.kessel@tu-berlin.de (P. Kessel).

https://doi.org/10.1016/j.patcog.2021.108194
0031-3203/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/4.0/)

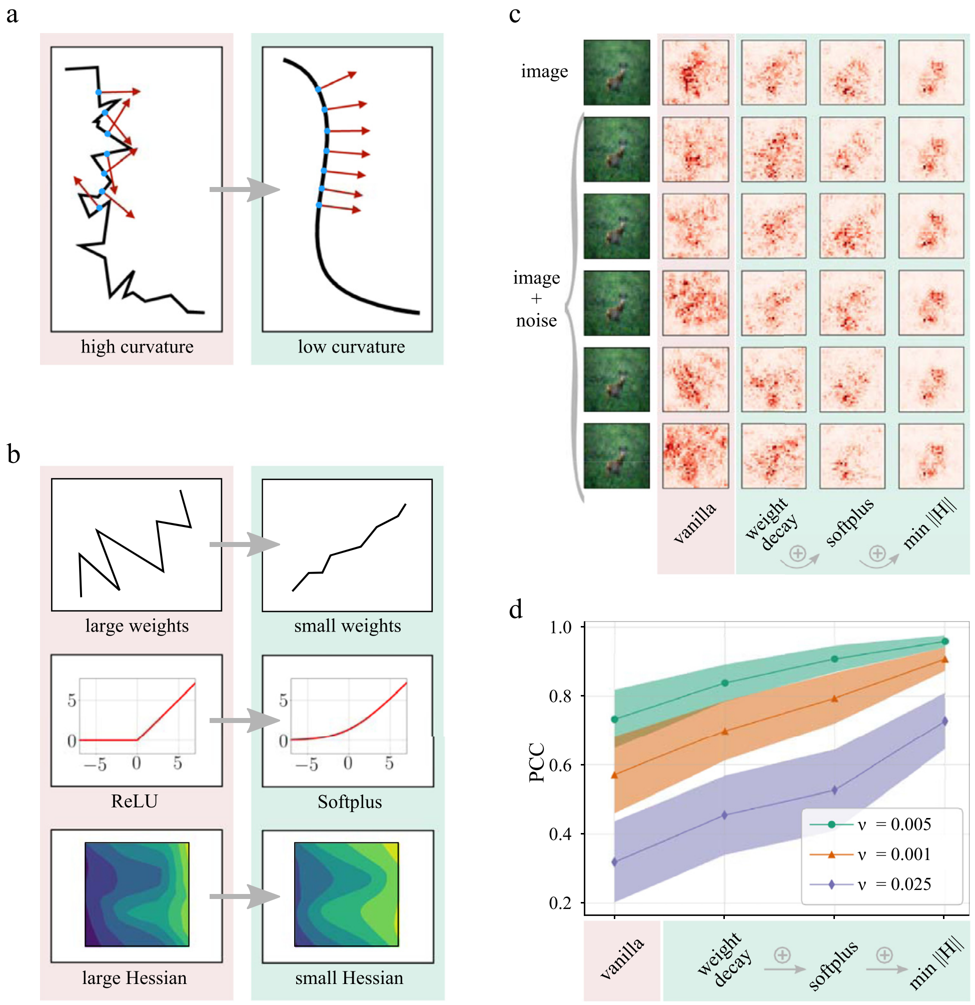

Fig. 1. Intuition for our approach and results. **a** The gradient (red arrows) changes drastically when moving along a line with high curvature but changes only gradually when the curvature is low. We get a similar effect for networks with high and low curvature (see c). **b** We propose several techniques that reduce curvature when incorporated in the training procedure. Weight decay flattens the angles between piece-wise linear functions, softplus smooths out the kinks of the ReLU function, Hessian minimization reduces curvature locally at the data points. **c** We show how the Gradient explanation maps change when adding small perturbations to the input. For the vanilla net the explanation maps differ strongly. For networks trained with a combination of our proposed methods the explanation maps become robust to the input perturbations. For a network trained with weight decay, softplus activations and Hessian minimization (last column) the explanation stays approximately constant. **d** A quantitative analysis on the complete test set confirms our theoretical findings. The similarity—measured as Pearson correlation coefficient (PCC)—between original explanation and explanation of a perturbed input is significantly higher for networks trained with our methods. We get most benefits when combining our methods (last network). We show results for three different noise levels ν. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

theoretical framework which allows us to derive bounds on the maximal change.

- Based on this theoretical framework, we derive several techniques to make neural networks more resilient against attacks on the explanation, namely:
  - regularizing: here, training with weight decay,
  - training with smoothed activation functions,
  - training while minimizing the Hessian of the network with respect to the input.
- We demonstrate the effectiveness of the above methods experimentally for several different explanation methods and architectures on the CIFAR-10 and ImageNet data set.

Fig. 1 provides an intuition for why explanations are susceptible to manipulation and how our methods lead to more robust explanations.

1.1. Related work

In some works, changing the input is part of the explanation process itself: [12] averages gradients over noisy inputs, [7] trains an interpretable (linear) classifier on perturbed samples, [13] and [15] use gradual interpolation between an input and a root point to create the explanation and [3] occludes part of the input with a gray square and then tracks the correct class probability, as a function of the position of the occluded area. Other studies have used input or model manipulation to compare different explanation methods: In [5,36] the pixel flipping method is introduced. Based on the relevance score of the explanation, pixel values are sequentially replaced and the effect on the network output is measured. The idea is to verify that the explanation method accurately identified the relevant pixels—those that the network needs to correctly classify an image. Variants of this method are also used in [37] and [38]. [39] takes a similar approach but retrains on the modified data to avoid effects from artifacts outside of the original data distribution. Reference [40] proposes "sanity checks" for explanations by either perturbing the model weights or the labels of the data set and measuring the resulting change in the explanation. Some explanations turn out to be visually appealing but insensitive to the model or the data generating process. In [41], a more theoretical analysis of why many propagation-based explanations fail the sanity checks introduced in [40] is given. Reference [42] modifies the input by adding a constant shift, which is then subtracted in the first layer by changing the bias. The modified network and data set produce the exact same outputs as the original but some explanation methods attribute relevance to the shifted pixels.

On the other hand, attention towards more malignant manipulation of explanations has developed relatively recently. The authors in [33] explicitly change model weights to manipulate the explanations while keeping the model output approximately unchanged and [34] expands on this, by analyzing these experimental results with theoretical insights and providing a method to make explanation more robust against model manipulations. In [43], a facade model is added to the original network in the context of Natural Language Processing. The combined network has similar predictions but the Gradient Explanation is dominated by the customized facade model. Furthermore, [44] proposes a similar approach treating the classifier as a blackbox. Their scaffolding technique can change an explanation of a biased classifiers to look completely innocuous.

In [31], an approach similar to conventional adversarial attacks on the model output is presented. The difference to adversarial attacks on the model output is that attacks on the explanation aim to keep the classification unchanged while the explanation shows major modifications. The work [32] expands on this and shows that explanations can be changed to arbitrary target maps by adding imperceptible perturbations to the input. The authors explain this susceptibility to manipulations with tools from differential geometry. Our theoretical analysis extends these results significantly in that it also holds for a small (but non-vanishing) change in output of the network, for more general network architectures and various attack methods.

Explanations that include averaging over multiple inputs [12] are naturally more resilient to input perturbations, but are not completely save from manipulation [32]. Several works propose to counteract susceptibility of explanations by changing the explanation process. In [32] ReLU activations are changed to softplus activations. This is done for the explanation process only and not part of the training process, in contrast to the approach taken in the present work. Reference [45] proposes to include a penalty on the largest principle curvature in the loss function to train networks that are more resilient to attacks on the explanation. This is different to our Hessian norm training which can be roughly understood as a penalty on all principle curvatures. Furthermore, [34] proposes a projection of the explanation onto the previously estimated data manifold, [46] shows that combining several explanation methods can often improve robustness to manipulated inputs, [45] averages over several examples from a uniform distribution around the input, and [47] proposes adversarial training to construct black box explanations like [7,10] that are robust to input perturbations and distribution shifts.

2. Theoretical considerations

In the following, we formally introduce the basic underlying idea of a theoretical analysis of explanation manipulability. Let us consider gradient explanations for concreteness. We restrict to the output of the winning class, i.e. $g(x) := g(x)_k$ with $k = \arg\max_i g(x)_i$, since the gradient method only depends on this component of the output. To manipulate the explanation of an input $x \in \mathbb{R}^N$ of a classifier $g : \mathbb{R}^N \to \mathbb{R}$, we construct an adversarially perturbed input $x_{\text{adv}} = x + \delta x$ such that the output of the network is (approximately) unchanged, i.e.

$$g(x) \approx g(x_{\text{adv}}) \tag{1}$$

but the corresponding (gradient) explanations $h = \nabla g$ are drastically different, i.e.

$$||h(x) - h(x_{\text{adv}})|| \gg 1\,. \tag{2}$$

Typically, the perturbation is assumed to be small, $||\delta x|| \ll 1$, such that it is imperceptible. For theoretical analysis, one would like to derive upper bounds on the change of saliency map $||h(x) - h(x + \delta x)||$ by any such perturbation δx. To this end, one considers a curve $\gamma : \mathbb{R} \to \mathbb{R}^N$ with affine parameter t connecting the unperturbed data point x with its adversarially perturbed counterpart x_{adv}, i.e.

$$\gamma(t = -\infty) = x\,, \quad \gamma(t = +\infty) = x_{\text{adv}}\,. \tag{3}$$

In practice, intermediate points on the curve may correspond to iterations of an optimization procedure which adversarially perturbs the input in an iterative manner (although this interpretation is not needed for any of the theoretical considerations). One can then use the gradient theorem to rewrite the change in j-th component of the explanation h as[1]

$$\begin{aligned} h_j(x) - h_j(x_{\text{adv}}) = \partial_j g(x) - \partial_j g(x_{\text{adv}}) &= \int_\gamma \sum_i \partial_i \partial_j g(x)\, \mathrm{d}x_i \\ &= \int_{-\infty}^{\infty} \sum_i \partial_i \partial_j g(\gamma(t))\, \dot{\gamma}_i(t)\, \mathrm{d}t\,, \end{aligned} \tag{4}$$

Let the Frobenius norm of the Hessian $H_{ij}(g) = \partial_i \partial_j g$ be bounded, i.e.

$$||H(g)(x)|| \leq H^* \in \mathbb{R}_+\,, \qquad \forall x \in \mathbb{R}^N\,.$$

It then follows immediately that the maximal change in explanation is also bounded:

$$\begin{aligned} ||h(x) - h(x_{\text{adv}})|| &\leq \int_{-\infty}^{+\infty} ||H(g)\, \gamma(t)||\, \mathrm{d}t \\ &\leq H^* \int_{-\infty}^{+\infty} ||\gamma(t)||\, \mathrm{d}t = H^* L(\gamma)\,, \end{aligned} \tag{5}$$

where $L(\gamma) = \int_{-\infty}^{+\infty} ||\gamma(t)||\, \mathrm{d}t$ is the length of the curve γ. We have therefore deduced that bounding the Frobenius norm of the Hessian implies a bound on the maximal possible change in explanation by input manipulation.

3. Methods for robuster explanations

Based on the theoretical analysis in the last section, we propose three approaches to reduce the Frobenius norm of the Hessian and thereby increase the robustness with respect to explanation manipulation.

3.1. Curvature minimization

As a first approach, we propose to modify the training procedure such that a small value of the Frobenius norm of the Hessian is part of the objective. To this end, we add an additional term to the loss function which penalizes the Frobenius norm, i.e.

$$\mathcal{L} = \mathcal{L}_0 + \zeta \sum_{x \in \mathcal{T}} ||H||_F^2(x)\,, \tag{6}$$

where ζ is a hyperparameter regulating how strongly the Hessian norm is minimized. Futhermore, $\mathcal{T}$ denotes the training set and $\mathcal{L}_0$ is the unregularized loss function. A related approach has been previously proposed in [48] in the context of conventional adversarial attacks.

[1] Here we assume that the classifier g is twice differentiable. However, this assumption can, under certain circumstances, be relaxed as discussed in Section 3.3.

Calculating the Frobenius norm of the Hessian is expensive, i.e. to obtain the second derivative we would have to backpropagate through the network once per input pixel. For larger images, this becomes unfeasible especially when we want to include the norm minimization in the training procedure.

We therefore propose to estimate the Frobenius norm stochastically. Let $v \sim \mathcal{N}(0,1)$, which implies that $\mathbb{E}[v_i]=0$ and $\mathbb{E}[v_i v_j]=\delta_{ij}$.[2] We can then rewrite the Frobenius norm of the Hessian as follows

$$\begin{aligned}||H||_F^2 &= \sum_i \left(\frac{\partial^2 g}{\partial x_i \partial x_i}\right)^2 \\ &= \sum_{i,j} \mathbb{E}\left[v_i v_j\right]\left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2 \\ &= \mathbb{E}\left[\sum_i \left(\frac{\partial}{\partial x_i}\sum_j v_j \frac{\partial g}{\partial x_j}\right)^2\right].\end{aligned}$$

We can estimate the final expectation value by Monte-Carlo, i.e. we draw a random vector v, and compute $v^T \nabla g(x)$ at the usual cost of a single backward pass. Since the resulting expression is a scalar, we can calculate its derivative at the cost of another single backward pass [49]. Multiple samples can be combined in mini-batches. The average over the mini-batch is then an unbiased estimator for the expectation value.

3.2. Weight decay

The second approach starts from the observation that the Frobenius norm of the Hessian depends on the weights of the neural network. More precisely, in Appendix A we show that

Theorem 1. *Let $g:\mathbb{R}^N \to \mathbb{R}$ be a fully-connected neural network with L layers. The weights of the l-th layer are denoted by $W^{(l)}$ and its activation functions σ are twice-differentiable and bounded*

$$|\sigma'(x)| \leq \Sigma_1\,, \quad |\sigma''(x)| \leq \Sigma_2\,. \tag{7}$$

The Hessian of the network is then bounded by

$$||H(g)||_F \leq \sum_{m=1}^{L}\left(\prod_{l=1}^{m}||W^{(l)}||_F^2 \prod_{l=m+1}^{L}||W^{(l)}||_F\right)\Sigma_1^{L+m-2}\,\Sigma_2\,. \tag{8}$$

As a practical consequence of the theorem, we can reduce the maximal possible change in explanation by decreasing the Frobenius norms of the weights. Motivated by this theoretical insight, we propose to use weight decay for training neural networks such that their explanations are more robust to manipulation. Note while it is well-known that weight decay can improve generalization of neural networks [50–52], its effect on the manipulability of explanations has not previously been established. Other regularizations that reduce the weight norms (L^1-regularization, variants of L^2-regularization, etc [53]) may have a similar effect.

3.3. Smoothing activation functions

As a third approach, we note that the bound of the network's Hessian (8) also depends on the maximal values of the activation function's first and second derivatives (7). Choosing activations with smaller values for these maximal values therefore will lead to robuster explanations.

As a concrete example, consider the softplus activation function

$$\sigma(x) = \frac{1}{\beta}\ln(1+e^{\beta x})\,, \tag{9}$$

where $\beta \in \mathbb{R}_+$ is a hyperparameter. Its first and second derivative are bounded by

$$|\sigma'(x)| \leq 1\,, \qquad |\sigma''(x)| \leq \frac{1}{4}\beta\,, \tag{10}$$

and thus $\Sigma_1 = 1$ and $\Sigma_2 = \frac{1}{4}\beta$, see (7). From the bound (8), it then follows that networks with softplus non-linearities with smaller β value have robuster explanations compared to networks with larger values of β (provided that the Frobenius norms of the weights is the same).

We therefore propose to use smoother non-linearities, i.e. functions with small Σ_1 and Σ_2, to make explanations more robust.

Note on ReLU non-linearites The popular ReLU non-linearity can be recovered from softplus in the limit $\beta \to \infty$. Note however that the bound (8) diverges in this limit since $\Sigma_2 \to \infty$, see (10). The fundamental underlying difficulty is that the second derivative $\mathrm{relu}''(x)$ is ill-defined at $x=0$. In Appendix B, we, however, generalize the bound (8) to the case of ReLU non-linearities. For this, we use the fact that a distributional generalization of the second derivative of the ReLU non-linearity can be defined, i.e. $\mathrm{relu}''(x)=\delta(x)$ where δ denotes the Dirac distribution. The corresponding right-hand-side of this generalized bound only depends on the weights of the neural network. Thus, this result establishes theoretically that weight decay also certifiably improves robustness for ReLU non-linearities.

4. Experimental analysis

4.1. Overview

In this section, we compare the performance of the proposed methods experimentally.

Briefly summarized, we measure the degree of robustness as follows: we perturb an input sample x by Gaussian noise $\delta x \sim \mathcal{N}(0,\sigma^2)$ (for a discussion of other noise distributions, we refer to D.5). For the resulting adversarially perturbed input $x_{\mathrm{adv}} = x + \delta x$, we then calculate the explanation $h(x_{\mathrm{adv}})$ and measure its similarity to the original explanation $h(x)$. The standard deviation σ is chosen such that the output of the neural network is approximately unchanged, i.e. $g(x) \approx g(x_{\mathrm{adv}})$. We repeat this analysis for various explanation methods.

In more detail, our experiments use the following setup:

Similarity Scores for Explanations In order to quantify the visual similarity of the explanations, we use three different measures following [40]: Pearson correlation coefficient (PCC), structural similarity index measure (SSIM) and mean squared error (MSE). PCC and SSIM are relative error measures where values close to 1 indicate high similarity and small values indicate low similarity. MSE is an absolute error measure where values close to 0 indicate high similarity and large values indicate low similarity.

Model and Data set To demonstrate the proposed robustness effects generically, we use the same convolutional neural network (CNN) architecture for all our models and train on the CIFAR10 data set [54]. The models achieve up to 88% test set accuracy. For more details on the network architecture and training, we refer to D.2.

Noise Level We choose the level of noise such that it does not significantly change the network's output. To this end, we perturb all 10k images of the test set of CIFAR10 with Gaussian noise of a given standard deviation σ. It is convenient to express the standard deviation σ in terms of the noise level ν by

$$\sigma = (x_{max} - x_{min})\,\nu\,, \tag{11}$$

[2] Here, we use the Kronecker delta symbol with $\delta_{ij} = \begin{cases} 0 & i \neq j\,, \\ 1 & i = j\,. \end{cases}$

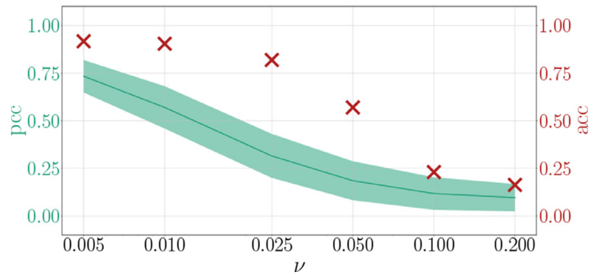

Fig. 2. PCC between explanations drops more rapidly than accuracy when adding noise with small ν to the original image. We show mean +/- std for PCC.

where $x_{\max}$ and $x_{\min}$ denote the maximum and the minimum values of the input domain. For other types of noise we refer to D.5.

Fig. 2 shows the classification accuracy and the PCC similarity score between the original and adversarially perturbed explanations for different noise levels ν. Smaller noise levels (between 0.005 and 0.025) lead to a comparatively mild drop in accuracy but result in a significant reduction in the similarity of the explanations. We therefore restrict the noise levels to this interval for our experiments.

Explanation Methods We apply our approach to the explanation methods following [32] :

- **Gradient**: The map $h(x) = \frac{\partial g}{\partial x}(x)$ quantifies change of the scores $g(x)$ due to infinitesimal perturbations in each pixel [1,2].
- **Gradient**×**Input**: This method uses the map $h(x) = x \odot \frac{\partial g}{\partial x}(x)$ [9] which, for linear models, gives the exact contribution of each pixel to the prediction.
- **Integrated Gradients**: This method defines the explanation map $h(x) = (x - \bar{x}) \odot \int_0^1 \frac{\partial g(\bar{x}+t(x-\bar{x}))}{\partial x} dt$, where $\bar{x}$ is a suitable baseline [13].
- **Guided Backpropagation (GBP)**: This method is a modification of the Gradient explanation which blocks negative components of the gradient when backpropagating through the nonlinearities [4].
- **Layer-wise Relevance Propagation (LRP)** is a framework [5,15] that applies specific rules at different layers to propagate relevance backwards through the network, see [55] for a complete overview over the possible choices for the propagation rules. We adopt the following conventions: for the output layer, the relevance is given by

$$R_i^L = \delta_{ik} \,, \tag{12}$$

where k is the index of the predicted class. This is then propagated backwards through all layers but the first using the z^+ rule

$$R_i^l = \sum_j \frac{x_i^l (W^l)^+_{ji}}{\sum_i x_i^l (W^l)^+_{ji}} R_j^{l+1} \,, \tag{13}$$

where $(W^l)^+$ denotes the positive weights of the l-th layer and x^l is the activation vector of the l-th layer. For the first layer, we use the $z^{\mathcal{B}}$ rule to account for the bounded input domain

$$R_i^0 = \sum_j \frac{x_j^0 W_{ji}^0 - l_j (W^0)^+_{ji} - h_j (W^0)^-_{ji}}{\sum_i (x_j^0 W_{ji}^0 - l_j (W^0)^+_{ji} - h_j (W^0)^-_{ji})} R_j^1 \,, \tag{14}$$

where l_i and h_i are the lower and upper bounds of the input domain respectively.

For an extensive overview of these methods see [19,20]. To obtain a pixel-wise relevance score, we sum over absolute values of the three colour channels and normalize the explanation to have $\sum_i |h(x)_i| = 1$.

4.2. Robustness from weight decay

Weight decay adds a regularizing term to the update rule of the network parameters w_i so that large values are penalized. The update is then given by

$$w_i \to w_i - \alpha \left(\frac{\partial \mathcal{L}_0}{\partial w_i} + \lambda w_i \right) \tag{15}$$

where α is the learning rate and $\mathcal{L}_0$ is the unregularized loss. The hyperparameter λ controls how strongly the network parameters are penalized. We choose five different values for λ and train the CNN for each. Fig. 3 shows higher PCC values for larger values of λ, i.e. weight decay increases the robustness of explanations with respect to input manipulation. As was to be expected, there is a trade-off between robustness and accuracy of the networks. For networks trained with strong weight decay ($\lambda >$ 1e-2), the accuracy decreases drastically. On the other hand, networks trained with 5e-5 $\leq \lambda \leq$ 5e-3 achieve comparable accuracy but are significantly more robust to manipulations than a network trained with $\lambda = 0$.

4.3. Robustness from softplus

To see how the β value of the softplus activations (9) affects the robustness, we train networks with four different β values. We do this for all but the largest value of the weight-decay hyperparameter λ from the previous section; in total $4 \cdot 5 = 20$ networks. With decreasing β values, the explanations become less prone to input manipulations. Fig. 4 shows the results for networks trained with $\lambda =$ 5e-4 and different values for β. For β values smaller than 5, the accuracy of the network decreases slightly. Crucially, comparable accuracy is achieved for β values of 5 and 10. Results for other choices of the weight decay parameter λ look qualitatively similar. We list results for all combinations in D.2.

4.4. Robustness from curvature minimization

To evaluate the effectiveness of Hessian norm minimization, we train networks with different values of the hyperparameter ζ which controls the degree of regularization in the modified loss in Eq. (6).

We approximate the Hessian norm only for softplus networks since we need to calculate second derivatives and[3]

$$\frac{\partial^2 g}{\partial x^2} \propto \mathrm{relu}'' = 0$$

for ReLU networks. We consider six different values for ζ for each of the networks from the previous section, i.e. we train $6 \cdot 20 = 120$ networks in total.

Fig. 5 shows how curvature minimization affects the robustness against random perturbations, when using weight decay with $\lambda =$ 5e-4 and softplus activations with $\beta = 10$. Even a small value for ζ results in significant improvement. For larger ζ values, the PCC value slowly converges to one. We list results for all combinations of the weight decay parameter λ and the softplus parameter β in D.2.

Fig. 6 shows a concrete example. In the top row, we show an image and several samples with added Gaussian noise (with noise level $\nu = 0.025$). Below we show the Gradient explanation maps of two different networks. For the first network (middle row) the explanations appear noisy and vary strongly. This net-

[3] More precisely, the second derivative $\mathrm{relu}''(x)$ is not defined for $x = 0$ and the relation only holds up to such root points of the non-linearity.

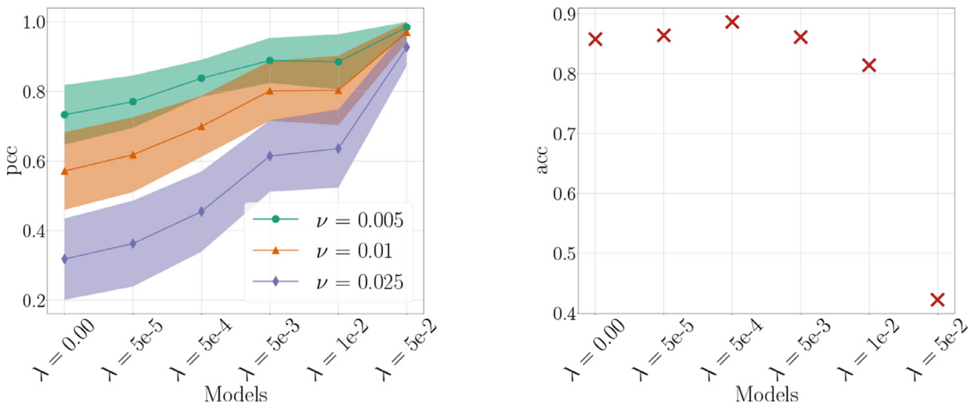

Fig. 3. Left: PCC increases with stronger weight decay (higher λ). Therefore, weight decay improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: For moderate weight decay ($\lambda \approx$ 5e-4) accuracy increases, while for strong weight decay ($\lambda \geq$ 1e-2) accuracy drops.

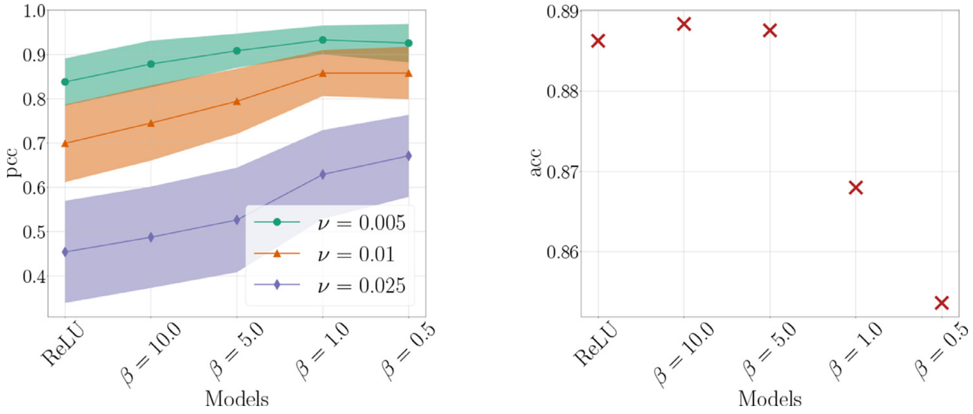

Fig. 4. Left: PCC is higher for networks trained with softplus activation that have small β value. Replacing ReLU with softplus activations improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: Accuracy decreases if β is very small. All networks were trained with weight decay (λ = 5e-4).

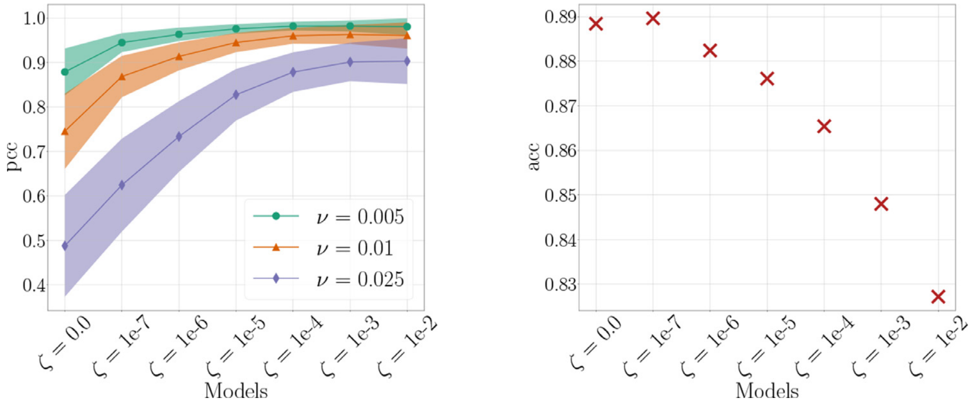

Fig. 5. Left: PCC is larger for networks trained with strong minimization of the Hessian norm $\|H\|$ (larger ζ values). Therefore, minimizing $\|H\|$ improves robustness of explanations. We show mean +/- std for three different noise levels ν. Right: accuracy decreases when ζ gets large. All networks were trained with weight decay (λ = 5e-4) and softplus activations ($\beta = 10$).

work was trained without any techniques to enhance robustness (no weight decay, ReLU activations, no Hessian minimization). For the second network (bottom row) the explanations stay relatively steady. This network was trained with measures that enhance robustness (weight decay with $\lambda =$ 5e-4, softplus activations with $\beta = 10$, Hessian minimization with $\zeta =$ 1e-7).

4.5. Additional architectures, data sets, and explanation methods

Explanation Methods: so far we have focused on Gradient explanation maps. But we can apply any other suitable explanation method to our networks. In Table 1, we show results for different explanation methods. Specifically, PCC values (averaged over

Table 1
PCC (mean ± std) between original explanations and explanations of perturbed inputs (noise level $\nu = 0.025$) for explanation maps: Gradient, Gradient×Input, Integrated Gradients, Guided Backpropagation (GBP), and Layerwise Relevance Propagation (LRP). High PCC values indicate high robustness.

Network	Gradient	Grad×Input	IntGrad	GBP	LRP
original	0.32±0.12	0.44±0.13	0.53±0.12	0.78±0.10	0.91±0.06
robust	0.73±0.08	0.76±0.08	0.82±0.06	0.94±0.03	0.98±0.01

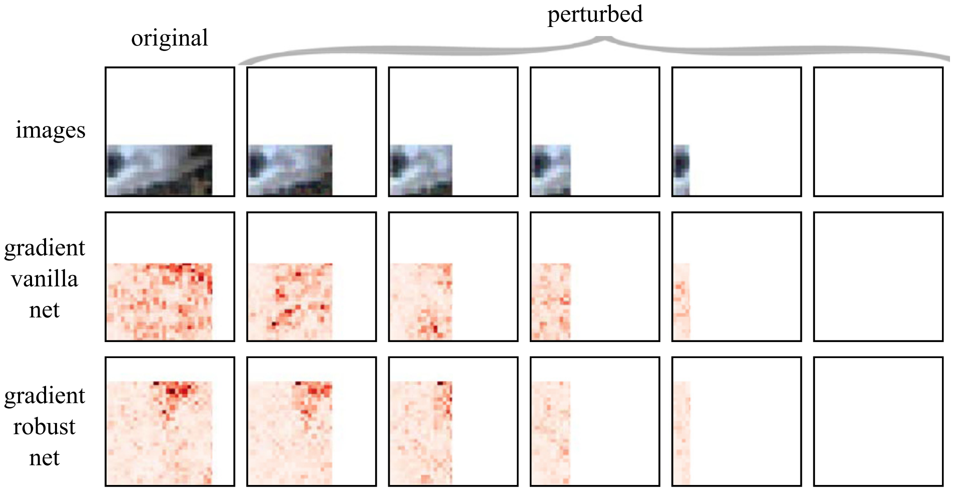

Fig. 6. Top row: original image and image with added noise ($\nu = 0.025$). Middle row: Gradient explanations for a network trained with $\lambda = 0$, ReLU activations and $\zeta = 0$. Bottom row: Gradient explanations for a network trained with weight decay ($\lambda =$ 5e-4), softplus activations ($\beta = 10$) and Hessian minimization ($\zeta =$ 1e-7). The explanations of the robust network in the bottom row are clearly more resilient to random input perturbations.

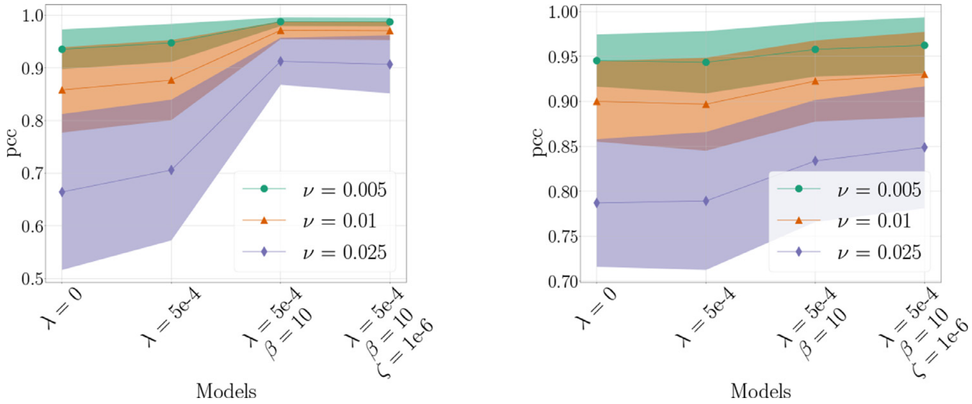

Fig. 7. VGG16 (left) and ResNet18 (right) trained on (a subset of) ImageNet. PCC is larger for networks trained with robustness methods..

the complete test set) between original and manipulated explanations when perturbing images with a noise level of $\nu = 0.025$ are listed. In the first row, we show how the respective explanations change when using the original network ($\lambda = 0$, ReLU activations, $\zeta = 0$) and in the second row we show the values for a network trained with all our robustness measures ($\lambda =$ 5e-4, $\beta = 10$, $\zeta =$ 1e-6). While the Gradient explanation is most vulnerable to random perturbations, the results for Gradient×Input, and Integrated Gradients look qualitatively similar to the Gradient explanation. When using all our robustifying measures the PCC similarity between these explanations improves by around 31 to 40 percentage points. Guided Backpropagation (GBP) and Layerwise Relevance Propagation (LRP) are noticeably more resilient to random perturbations. However, our robust network still achieves significantly higher PCC similarities, demonstrating that even more robust explanation methods can profit. We refer to D.4 for a more detailed discussion.

Data sets: in order to demonstrate that our results also hold for input data with higher dimensions, we consider the ImageNet data set. Due to the substantial computational costs, we restrict to a subset of the former following [56]. We train using the original train-test split of ImageNet, using 50 images per class for testing. *Architectures:* we also applied our methods to VGG16 [57] and ResNet18 [58]. The training was performed on the ImageNet data set as described above. The results are illustrated in Fig. 7. We observe similar behaviour as for the convolutional architecture used on the CIFAR10 data set. Notably, the largest increase in robustness is obtained by substituting ReLU with softplus activations while curvature minimization does not seem to have a comparative effect. For a more detailed discussion, we refer to Appendix F.

Targeted manipulation: in addition to perturbing the explanation by using unstructured random noise, we also considered a targeted manipulation method. For this, we use the CNN model on CIFAR10 as well as VGG16 and ResNet18 trained on the subset of ImageNet. We choose 100 randomly selected test samples. For each of them, the target explanations is chosen to be the explanation of another randomly selected test sample. As ReLU networks are not twice differentiable, we substitute the ReLU with softplus activations with small β value during the attack. For the final comparison, we restore the original ReLU activations. We stop an attack when the mean squared error exceeds a specified threshold. As shown in Fig. 8, our methods also significantly increase the robustness with respect to these targeted attacks. We refer to Appendix G for a more detailed discussion.

4.6. Comparison of proposed methods

All our proposed methods can improve robustness of explanations against input manipulations. We observe this trend for all considered explanation methods, similarity measures and noise levels.

We note that each method appears to improve robustness in a different manner. As evident from our theoretically-derived upper bound Eq. (8), both weight decay and small β values for the

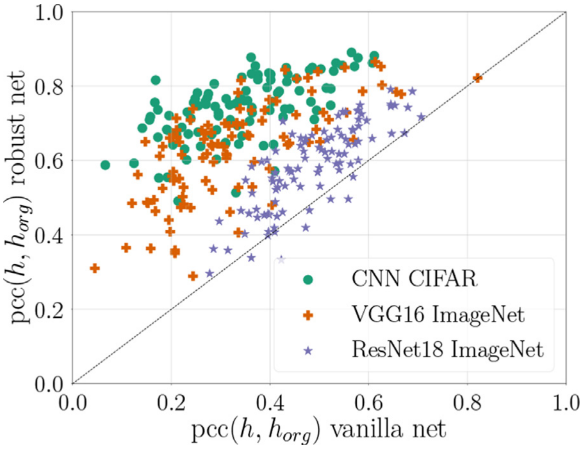

Fig. 8. Targeted attacks for various data sets and architectures. Models trained with robustness methods are significantly less vulnerable to targeted manipulations (Pearson correlation coefficient between manipulated explanation h and original explanation h_{org} is *higher* for the robust nets)..

softplus activations affect the Hessian norm. Weight decay leads to smaller Hessian norms by minimizing the weight norms. Replacing ReLU by softplus with comparatively small β parameter also leads to smaller Hessian norms but the weight norms stay approximately constant for different β values. When minimizing the Hessian norm directly during training, the Hessian norms decrease significantly while the weight norms decrease only minimally. This shows that Hessian norm minimization does not just improve robustness by indirectly minimizing the weight norms.

While we showed that each method separately improves robustness—we keep the weight-decay hyperparameter λ constant when evaluating different smoothing parameters β for the softplus activations and we keep the weight-decay hyperparameter λ and the smoothing parameter β constant when evaluating different values for hyperparameter ζ for the Hessian norm minimization—we get most benefits when combining them. Besides enhancing robustness, weight decay plays an essential role for the accuracy—as expected and well-known in the literature [50]; all networks trained without weight decay stay at an accuracy below 86.5%.

5. Conclusion

Explanation methods have gained significant popularity among practitioners in science and engineering recently. With increased attention to explainable AI, questions about manipulability and thus trustworthiness of explanations have been raised. In this contribution, we have addressed the need for robustness of explanation methods against manipulation of the input data. Rather than introducing a new explanation method, we focused on enhancing the robustness of the networks themselves and, as a result, any applied explanation method was shown to profit.

We could derive bounds for the maximal change in explanation. Based on this theoretical analysis, we proposed three approaches to increase the robustness of explanations. Specifically, we show that weight decay can efficiently boost robustness of explanations. We furthermore propose to use networks with smoothed activation functions and to include a regularizer for the network's curvature in the training process, which leads to significantly enhanced resilience against manipulated inputs.

An interesting direction for future research will be to relate the established limits of robust explanation methods to techniques for uncertainty quantification respectively in relation to methods studying the relevant structural parts in learning models [59,60]. Furthermore it will be helpful to discuss resilience to manipulation of explanation methods also in the context of unsupervised learning [61–63] and multi-modal data/similarity streams [64].

Declaration of Competing Interest

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

Acknowledgments

This work was supported in part by the German Ministry for Education and Research (BMBF) under Grants 01IS14013A-E, 01GQ1115, 01GQ0850, 031L0207D, 01IS18025A and 01IS18037A. This work was also partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korea Government (No. 2017-0-00451, Development of BCI based Brain and Cognitive Computing Technology for Recognizing User's Intentions using Deep Learning and No. 2019-0-00079, Artificial Intelligence Graduate School Program, Korea University), as well as by the Research Training Group "Differential Equation- and Data-driven Models in Life Sciences and Fluid Dynamics (DAEDALUS)" (GRK 2433) and Grant Math+, EXC 2046/1, Project ID 390685689 both funded by the German Research Foundation (DFG). We gratefully acknowledge helpful comments on the ms by Rodolphe Jenatton. Correspondence to KRM and PK.

Appendix A. Proof of Theorem 1

Let $\sigma(x)$ denote the non-linearity of the network. We also use the notation $\sigma^{(l)}(x) = \sigma(W^{(l)}x)$ where $W^{(l)}$ are the weights of layer l. By assumption, the activation functions σ are twice-differentiable and bounded

$$|\sigma'(x)| \leq \Sigma_1\,, \quad |\sigma''(x)| \leq \Sigma_2\,. \tag{A.1}$$

The activation at layer L is then given by

$$a^{(L)}(x) = (\sigma^{(L)} \circ \cdots \circ \sigma^{(1)})(x) \tag{A.2}$$

Its derivative $\partial_k a_i^{(l)}$ is equal to

$$\sum_{s_2 \ldots s_l} W_{is_l}^{(l)} \sigma'\left(\sum_j W_{ij}^{(l)} a_j^{(l-1)}\right) W_{s_l s_{l-1}}^{(l-1)} \sigma'\left(\sum_j W_{s_l j}^{(l-1)} a_j^{(l-2)}\right) \\ \ldots W_{s_2 k}^{(1)} \sigma'\left(\sum_j W_{s_2 j}^{(1)} x_j\right).$$

We therefore obtain

$$\left\| \nabla a^{(l)} \right\|_F \leq (\Sigma_1)^l \prod_{i=1}^{l} \left\| W^{(i)} \right\|_F \tag{A.3}$$

From the expression for $\partial_k a_i^{(l)}$, we can straightforwardly derive that

$$\partial_l \partial_k a_i^{(L)} = \sum_m \sum_{s_2 \ldots s_L} \Bigg\{ \\ W_{is_L}^{(L)} \sigma'\left(\sum_j W_{ij}^{(L)} a_j^{(L-1)}\right) W_{s_L s_{L-1}}^{(L-1)} \sigma'\left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)}\right) \\ \cdots \sum_p W_{s_{m+1} p}^{(m)} W_{s_{m+1} s_m}^{(m)} \sigma''\left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x)\right) \partial_l a_p^{(m-1)}(x) \\ \ldots W_{s_2 k}^{(1)} \sigma'\left(\sum_j W_{s_2 j}^{(1)} x_j\right) \Bigg\}.$$

Restrict to the case for which the index i only takes a single value, the Hessian $H_{ij}(g) = \partial_i \partial_j a^L(x)$ is then bounded by

$$\|H(g)\|_F \le \sum_{m=1}^{L} \left(\prod_{l=1}^{m} ||W^{(l)}||_F^2 \prod_{l=m+1}^{L} ||W^{(l)}||_F \right) \Sigma_1^{L+m-2}\, \Sigma_2 \,. \tag{A.4}$$

Appendix B. Relu networks

As was discussed in the main text, for softplus non-linearites the bound (8) diverges for ReLU non-linearities. This is because ReLU can be obtained from softplus by taking the limit $\beta \to \infty$ and the constant Σ_2 in (8) diverges in this limit, see (10). The underlying fundamental difficulty is that the Hessian of ReLU networks is not well-defined.

In the following, we will discuss how to generalize the analysis to networks with ReLU activations. We will establish that a distributional generalization of the Hessian can be derived for ReLU networks. A distributional form of the Hessian is sufficient for our purposes because in deriving a bound for the maximal change in explanation we only need to consider the Hessian under an integral, see (4). Since integrals over distributions are well-defined, the resulting expression will be well-defined as well.

In the following, we will first illustrate this for a simple toy model before considering the general case.

B1. Toy example

Consider the network (depicted in Fig. B.9)

$$g(x) = \mathrm{relu}(w^{(1)T}x) + \mathrm{relu}(w^{(2)T}x) \tag{B.1}$$

with input vector $x \in \mathbb{R}^2$ and weight vectors $w^{(1)} = \frac{1}{\sqrt{2}}[1, 1]^T$ and $w^{(2)} = \frac{1}{\sqrt{2}}[1, -1]^T$.

The first derivative with respect to x_j is

$$\partial_j g(x) = w_j^{(1)}\theta(w^{(1)T}x) + w_j^{(2)}\theta(w^{(2)T}x)\,, \tag{B.2}$$

where we have defined the Heaviside step function

$$\theta(x) = \begin{cases} 1 & x \geq 0\,, \\ 0 & x < 0\,. \end{cases} \tag{B.3}$$

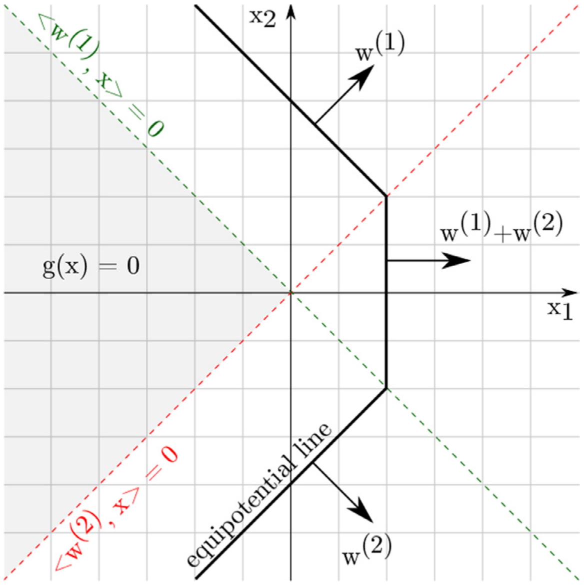

Fig. B.9. Toy function. Lines of rootpoints are marked in green and red. The grey area shows where $g(x) = 0$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

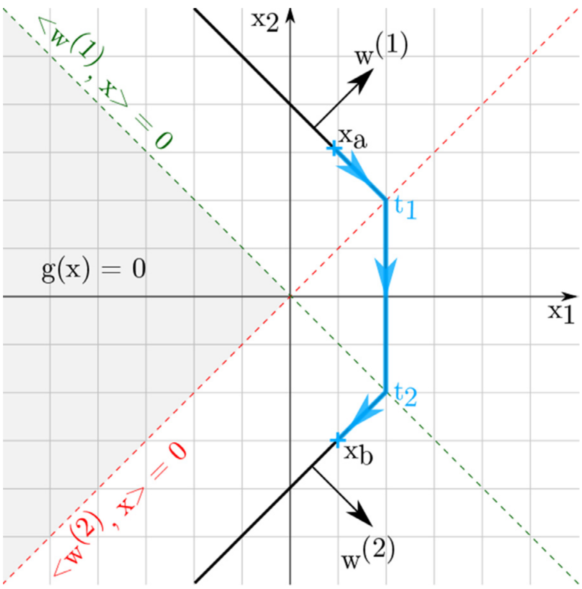

Fig. B.10. Path (in blue) along an equipotential line (constant network output $g(x) = \mathrm{const}$). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We note that the derivative of the step function is not well-defined at zero. However, a distributional generalization thereof can be defined, i.e.

$$\theta'(x) = \delta(x)\,, \tag{B.4}$$

where δ denotes the Dirac delta distribution.

With this definition, the ij-th entry of (the distributional generalization of) the Hessian matrix can formally be written as

$$\partial_i \partial_j g(x) = w_i^{(1)} w_j^{(1)} \delta(w^{(1)T}x) + w_i^{(2)} w_j^{(2)} \delta(w^{(2)T}x)\,. \tag{B.5}$$

By (4), the change in (gradient) explanation when moving from point x to x_{adv} is then given by

$$(h(x) - h(x_{\mathrm{adv}}))_j = \int_{-\infty}^{\infty} \sum_i \Big(w_i^{(1)} w_j^{(1)} \delta(w^{(1)T}x) + w_i^{(2)} w_j^{(2)} \delta(w^{(2)T}x) \Big) \dot{x}_i \,\mathrm{d}t\,, \tag{B.6}$$

where we have used the notation $x(t)$ for the curve connecting the unperturbed and perturbed data points.

For integrating over the delta distribution in composition with a (scalar-valued) function, we use

$$\int_{-\infty}^{\infty} f(t)\delta(y(t))\mathrm{d}t = \sum_{t_N} \frac{f(t_N)}{|y'(t_N)|} \tag{B.7}$$

with t_N being the roots of $y(t)$. Using this expression, we then obtain the following change in saliency map

$$\begin{aligned}(h(x) - h(x_{\mathrm{adv}}))_j &= \sum_{t_N} \frac{\sum_i w_i^{(1)} w_j^{(1)} \dot{x}_i}{\left|\sum_i w_i^{(1)} \dot{x}_i\right|} + \sum_{t_N} \frac{\sum_i w_i^{(2)} w_j^{(2)} \dot{x}_i}{\left|\sum_i w_i^{(2)} \dot{x}_i\right|} \\ &= \sum_{t_N} \mathrm{sgn}\left(\sum_i w_i^{(1)} \dot{x}_i \right) w_j^{(1)} \\ &\quad + \sum_{t_N} \mathrm{sgn}\left(\sum_i w_i^{(2)} \dot{x}_i \right) w_j^{(2)}\,.\end{aligned}$$

Consider the blue path in Fig. B.10 whose root points are denoted by t_1 and t_2. We note that these root points correspond to kinks in the curve $x(t)$ connecting the unperturbed and perturbed data point. Their corresponding normalized velocity vectors are given by $\dot{x}(t_1) = w^{(2)}$ and $\dot{x}(t_2) = (0, -1)^T$ respectively. We therefore obtain

$$\begin{aligned} h(x) - h(x_{\text{adv}}) &= \operatorname{sgn}\big(\langle w^{(1)}, \dot{x}(t_2)\rangle\big) w^{(1)} + \operatorname{sgn}\big(\langle w^{(2)}, \dot{x}(t_1)\rangle\big) w^{(2)} \\ &= \operatorname{sgn}\left(-\frac{1}{\sqrt{2}}\right) w^{(1)} + \operatorname{sgn}(1) w^{(2)} \\ &= w^{(2)} - w^{(1)} \end{aligned}$$

which is correct as $h(x) = w^{(2)}$ and $h(x_{\text{adv}}) = w^{(1)}$. It is important to stress that we have obtain this result despite the fact that the Hessian of the neural network g is only given in generalized distributional form.

B2. General case

The argument of the previous section can be generalized to arbitrary fully-connected networks with weights W^l of layer $l \in \{1, \dots, L\}$. The general logic follows closely the toy model discussed in the previous section, i.e. a distributional generalization of the Hessian is derived and since on the right-hand-side of (4) the Hessian only appears under an integral, this distributional form is sufficient to obtain a bound on the maximal change in explanation due to a perturbation of the input. Using this technique, we derive the following theorem:

Theorem 2. *Let x and $x_{\text{adv}} = x + \delta x$ denote the unperturbed and perturbed data points respectively. We denote by $x(t)$ the curve connecting the unperturbed and perturbed points, i.e. $x(t = -\infty) = x$ and $x(t = +\infty) = x_{\text{adv}}$. Furthermore, we assume that all points on the curve have the same network output, i.e. $g(x(t_1)) = g(x(t_2))$ for all $t_1, t_2 \in \mathbb{R}$. The maximal change of explanation is then given by*

$$\big|\big|h(x) - h(x_{\text{adv}})\big|\big|^2 \leq \sum_{\text{kinks}(x(t))} \left(\prod_{l=1}^{L} \big\| W^{(l)} \big\|_F^2 \right), \tag{B.8}$$

where the sum runs over all kinks of the curve $x(t)$.

We can give an intuition for the theorem by considering the blue curve of Fig. B.10 for the toy model of the previous section. In this case, the sum over the kinks would run over $x(t_1)$ and $x(t_2)$, see Fig. B.10. Only at these kinks, the gradient of the network will change. In the theorem, we then estimate this change by its maximal value, i.e. the change is equal to the product of all weights.

As a practical consequence of the theorem, we can make explanations more robust by weight decay also in the case of ReLU non-linearities.

Proof: Let $W^{(l)}$ be the weights of layer l. We denote the l-th layer by $\operatorname{relu}^{(l)}(x) = \operatorname{relu}(W^{(l)} x)$. It then follows that

$$\partial_k \operatorname{relu}\left(\sum_j W_{ij} x_j\right) = W_{ik}\, \theta\left(\sum_j W_{ij} x_j\right) \tag{B.9}$$

$$\partial_l\, \theta\left(\sum_j W_{ij} x_j\right) = W_{il}\, \delta\left(\sum_j W_{ij} x_j\right) \tag{B.10}$$

where θ and δ are the Heaviside step function and the delta distribution respectively. The activation at layer L is then given by

$$a^{(L)}(x) = (\operatorname{relu}^{(L)} \circ \dots \circ \operatorname{relu}^{(1)})(x) \tag{B.11}$$

Its derivative $\partial_k a_i^{(L)}$ is equal to

$$\sum_{s_2 \dots s_L} W_{i s_L}^{(L)} \theta\left(\sum_j W_{ij}^{(L)} a_j^{(L-1)}\right) W_{s_L s_{L-1}}^{(L-1)} \theta\left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)}\right) \dots W_{s_2 k}^{(1)} \theta\left(\sum_j W_{s_2 j}^{(1)} x_j\right)$$

Deriving this expression for $\partial_k a_i^{(L)}$ again, we obtain

$$\begin{aligned} \partial_l \partial_k a_i^{(L)} = \sum_m \sum_{s_2 \dots s_L} \Bigg\{ & W_{i s_L}^{(L)} \theta\left(\sum_j W_{ij}^{(L)} a_j^{(L-1)}\right) W_{s_L s_{L-1}}^{(L-1)} \theta\left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)}\right) \\ & \dots \sum_p W_{s_{m+1} p}^{(m)} W_{s_{m+1} s_m}^{(m)} \delta\left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x)\right) \partial_l a_p^{(m-1)}(x) \\ & \dots W_{s_2 k}^{(1)} \theta\left(\sum_j W_{s_2 j}^{(1)} x_j\right) \Bigg\} \end{aligned}$$

We now restrict to the case that $a^{(L)}$ has only a single output value. As a result, the index i in the expression above only takes one value, i.e. $i = 1$. We define $g(x) = a_1^{(L)}(x)$ to ease notation. We then substitute this expression for $\partial_l \partial_k g = \partial_l \partial_k a_1^{(L)}$ in (4) and obtain

$$\begin{aligned} \big(h(x) - h(x_{\text{adv}})\big)_k = \sum_m \sum_{s_2 \dots s_L} \int_{-\infty}^{\infty} \mathrm{d}t \\ & \Bigg\{ W_{1 s_L}^{(L)} \theta\left(\sum_j W_{ij}^{(L)} a_j^{(L-1)}\right) W_{s_L s_{L-1}}^{(L-1)} \theta\left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)}\right) \\ & \dots \sum_{\hat{s}_m} W_{s_{m+1} \hat{s}_m}^{(m)} W_{s_{m+1} s_m}^{(m)} \delta\left(\sum_j W_{s_{m+1} j}^{(m)} a_j^{(m-1)}(x)\right) \dot{a}_{\hat{s}_m}^{(m-1)}(x) \\ & \dots W_{s_2 k}^{(1)} \theta\left(\sum_j W_{s_2 j}^{(1)} x_j\right) \Bigg\}, \end{aligned}$$

where we have used the notation $\partial_t a^{(m-1)} = \dot{a}^{(m-1)}$ for notational simplicity. Using the identity (B.7), we then obtain

$$\begin{aligned} \big(h(x) - h(x_{\text{adv}})\big)_k = \sum_m \sum_{x_N^m} \sum_{s_2 \dots s_L} \\ & \Bigg\{ W_{1 s_L}^{(L)} \theta\left(\sum_j W_{ij}^{(L)} a_j^{(L-1)}\right) W_{s_L s_{L-1}}^{(L-1)} \theta\left(\sum_j W_{s_L j}^{(L-1)} a_j^{(L-2)}\right) \\ & \dots W_{s_{m+1} s_m}^{(m)} \operatorname{sgn}\left(\sum_j W_{s_{m+1} j}^{(m)} \dot{a}_j^{(m-1)}(x_N^m)\right) \\ & \dots W_{s_2 k}^{(1)} \theta\left(\sum_j W_{s_2 j}^{(1)} (x_N^m)_j\right) \Bigg\}, \end{aligned}$$

where the sum over x_N^m runs over all zeropoints of $\sum_j W_{s_{m+1} j}^{(m)} a^{(m-1)}$ along the trajectory connecting x with x_{adv}. Using the fact that $|\theta(\bullet)| \leq 1$ and $|\operatorname{sgn}(\bullet)| \leq 1$, we obtain

$$\big|\big|h(x) - h(x_{\text{adv}})\big|\big|^2 \leq \sum_m \sum_{x_N^m} \big\|W^{(L)}\big\|_F^2 \big\|W^{(L-1)}\big\|_F^2 \dots \big\|W^{(m)}\big\|_F^2 \dots \big\|W^{(1)}\big\|_F^2. \tag{B.12}$$

As in the case of the toymodel, the summands run over all kinks of the trajectory. This bound for ReLU networks depends purely on the network weights and the number of kinks passed when moving from x to x_{adv}. If we reduce the Frobenius norms of the weights, we also reduce the maximal possible change in explanation.

Table D.2
Statistics of all network configurations. Columns show weight decay, activation function (ReLU or β parameter for softplus), parameter for curvature minimization ζ, test accuracy (acc), mean Pearson correlation coefficient (pcc) for Gaussian noise with different noise levels ν, average weight norm ($\|W\|$) and average approximated Hessian norm ($\|H\|$).

λ	β	ζ	acc	pcc for different ν			$\|\|W\|\|$	$\|\|H\|\|$
				0.005	0.01	0.025		
0	ReLU	0.0	85.75	0.73	0.57	0.32	30.79	-
5e-5	ReLU	0.0	86.38	0.77	0.62	0.36	23.28	-
5e-4	ReLU	0.0	88.63	0.84	0.70	0.45	11.37	-
5e-3	ReLU	0.0	86.10	0.89	0.80	0.62	4.80	-
1e-2	ReLU	0.0	81.41	0.89	0.80	0.64	3.61	-
0	10	0.0	85.61	0.81	0.63	0.34	30.01	503.61
0	5	0.0	85.60	0.88	0.73	0.39	28.70	280.71
0	1	0.0	85.60	0.93	0.85	0.61	27.76	59.21
0	5e-1	0.0	84.51	0.94	0.88	0.67	28.84	39.06
5e-5	10	0.0	86.36	0.86	0.70	0.39	22.91	298.79
5e-5	5	0.0	86.33	0.91	0.78	0.46	22.97	154.94
5e-5	1	0.0	86.03	0.94	0.86	0.62	22.73	51.63
5e-5	5e-1	0.0	85.34	0.94	0.88	0.67	23.76	36.43
5e-4	10	0.0	88.84	0.88	0.75	0.49	11.24	91.68
5e-4	5	0.0	88.76	0.91	0.79	0.53	11.36	59.57
5e-4	1	0.0	86.80	0.93	0.86	0.63	11.93	28.45
5e-4	5e-1	0.0	85.36	0.93	0.86	0.67	10.62	16.78
5e-3	10	0.0	86.13	0.91	0.82	0.64	4.81	9.54
5e-3	5	0.0	85.44	0.91	0.83	0.64	4.76	7.49
5e-3	1	0.0	83.35	0.92	0.86	0.71	4.86	3.79
5e-3	5e-1	0.0	77.60	0.96	0.93	0.85	4.66	2.02
1e-2	10	0.0	80.44	0.90	0.82	0.66	3.48	5.42
1e-2	5	0.0	77.57	0.89	0.82	0.65	3.33	5.27
1e-2	1	0.0	71.74	0.97	0.94	0.86	3.21	1.26
1e-2	5e-1	0.0	72.03	0.98	0.95	0.89	3.35	0.97
0	10	1e-7	85.65	0.95	0.87	0.60	24.72	56.77
0	10	1e-6	85.74	0.97	0.92	0.73	22.45	21.47
0	10	1e-5	85.56	0.98	0.95	0.84	20.59	8.49
0	10	1e-4	84.12	0.99	0.97	0.90	19.14	3.23
0	10	1e-3	82.40	0.99	0.98	0.94	17.91	1.19
0	10	1e-2	80.07	0.99	0.98	0.94	17.08	0.43
0	5	1e-7	86.26	0.95	0.88	0.64	25.44	49.58
0	5	1e-6	85.94	0.97	0.92	0.74	23.41	20.85
0	5	1e-5	85.87	0.98	0.95	0.83	21.88	7.91
0	5	1e-4	84.81	0.98	0.97	0.90	20.74	2.96
0	5	1e-3	83.12	0.99	0.98	0.93	19.72	1.29
0	5	1e-2	80.95	0.99	0.98	0.94	19.02	0.41
0	1	1e-7	85.24	0.94	0.88	0.69	27.69	31.61
0	1	1e-6	85.17	0.95	0.90	0.75	26.29	17.67
0	1	1e-5	84.85	0.97	0.94	0.83	25.11	7.62
0	1	1e-4	84.70	0.98	0.95	0.87	24.25	3.03
0	1	1e-3	82.68	0.98	0.96	0.90	23.23	1.16
0	1	1e-2	81.57	0.98	0.95	0.89	22.20	0.42
0	5e-1	1e-7	81.90	0.95	0.90	0.77	12.05	9.28
0	5e-1	1e-6	85.46	0.96	0.91	0.77	28.07	15.20
0	5e-1	1e-5	84.41	0.97	0.93	0.82	26.81	6.79
0	5e-1	1e-4	84.08	0.98	0.96	0.89	25.59	2.86
0	5e-1	1e-3	82.84	0.98	0.96	0.89	22.21	1.12
0	5e-1	1e-2	81.04	0.98	0.96	0.90	23.36	0.41
5e-5	10	1e-7	86.68	0.95	0.87	0.62	20.24	52.22
5e-5	10	1e-6	86.47	0.97	0.92	0.74	18.75	21.36
5e-5	10	1e-5	85.87	0.98	0.95	0.84	17.28	8.18
5e-5	10	1e-4	84.55	0.99	0.97	0.90	16.05	3.29
5e-5	10	1e-3	83.03	0.99	0.98	0.93	15.01	1.14
5e-5	10	1e-2	80.47	0.99	0.98	0.94	13.88	0.43
5e-5	5	1e-7	86.76	0.95	0.87	0.62	21.05	45.57
5e-5	5	1e-6	86.41	0.97	0.92	0.74	19.57	19.16
5e-5	5	1e-5	86.16	0.98	0.95	0.84	18.29	8.00
5e-5	5	1e-4	85.21	0.99	0.97	0.90	17.20	3.24
5e-5	5	1e-3	82.69	0.99	0.98	0.93	16.41	1.17
5e-5	5	1e-2	80.91	0.99	0.97	0.93	15.35	0.45
5e-5	1	1e-7	85.78	0.95	0.89	0.70	22.42	29.93
5e-5	1	1e-6	86.31	0.96	0.92	0.77	21.76	16.98
5e-5	1	1e-5	85.54	0.97	0.93	0.82	20.53	7.38
5e-5	1	1e-4	84.62	0.98	0.95	0.88	18.94	2.99
5e-5	1	1e-3	83.82	0.98	0.97	0.91	18.00	1.17
5e-5	1	1e-2	80.49	0.98	0.97	0.92	17.10	0.43
5e-5	5e-1	1e-7	84.84	0.95	0.89	0.70	23.02	26.39
5e-5	5e-1	1e-6	85.21	0.96	0.91	0.75	20.38	16.81
5e-5	5e-1	1e-5	82.98	0.96	0.92	0.81	12.51	6.07

(*continued on next page*)

Table D.2 (*continued*)

λ	β	ζ	acc	pcc for different ν			$\|\|W\|\|$	$\|\|H\|\|$
5e-5	5e-1	1e-4	84.29	0.97	0.95	0.87	15.54	3.02
5e-5	5e-1	1e-3	82.67	0.98	0.96	0.89	14.95	1.12
5e-5	5e-1	1e-2	80.35	0.98	0.96	0.89	11.56	0.42
5e-4	10	1e-7	88.96	0.94	0.87	0.62	10.95	35.47
5e-4	10	1e-6	88.24	0.96	0.91	0.73	10.76	17.68
5e-4	10	1e-5	87.61	0.98	0.94	0.83	9.98	7.27
5e-4	10	1e-4	86.54	0.98	0.96	0.88	9.20	3.08
5e-4	10	1e-3	84.80	0.98	0.96	0.90	8.42	1.10
5e-4	10	1e-2	82.72	0.98	0.96	0.90	7.67	0.41
5e-4	5	1e-7	88.68	0.94	0.87	0.63	11.07	32.80
5e-4	5	1e-6	88.31	0.96	0.91	0.73	10.78	16.68
5e-4	5	1e-5	87.75	0.97	0.94	0.83	10.03	6.97
5e-4	5	1e-4	86.35	0.98	0.96	0.88	9.70	2.96
5e-4	5	1e-3	84.74	0.98	0.96	0.91	8.97	1.16
5e-4	5	1e-2	82.12	0.98	0.96	0.91	8.04	0.44
5e-4	1	1e-7	87.11	0.94	0.86	0.64	11.67	25.09
5e-4	1	1e-6	87.08	0.96	0.91	0.75	11.34	14.06
5e-4	1	1e-5	86.72	0.97	0.94	0.82	11.13	7.31
5e-4	1	1e-4	85.64	0.98	0.96	0.88	9.81	2.92
5e-4	1	1e-3	83.48	0.98	0.97	0.91	9.49	1.20
5e-4	1	1e-2	81.87	0.98	0.96	0.91	8.11	0.44
5e-4	5e-1	1e-7	78.10	0.93	0.87	0.73	7.07	4.57
5e-4	5e-1	1e-6	85.76	0.95	0.90	0.75	11.03	12.23
5e-4	5e-1	1e-5	85.56	0.97	0.93	0.82	10.82	6.20
5e-4	5e-1	1e-4	85.21	0.98	0.96	0.88	10.21	2.86
5e-4	5e-1	1e-3	81.05	0.96	0.93	0.84	6.99	0.94
5e-4	5e-1	1e-2	81.98	0.98	0.97	0.91	8.12	0.46
5e-3	10	1e-7	86.09	0.91	0.83	0.65	4.79	8.23
5e-3	10	1e-6	85.92	0.91	0.83	0.66	4.74	6.31
5e-3	10	1e-5	85.59	0.93	0.87	0.72	4.66	3.73
5e-3	10	1e-4	84.53	0.95	0.91	0.79	4.50	1.88
5e-3	10	1e-3	83.11	0.96	0.93	0.84	4.30	0.86
5e-3	10	1e-2	80.16	0.97	0.95	0.88	4.00	0.33
5e-3	5	1e-7	85.90	0.90	0.82	0.64	4.78	7.09
5e-3	5	1e-6	85.43	0.91	0.84	0.67	4.79	5.57
5e-3	5	1e-5	85.26	0.92	0.86	0.71	4.67	3.48
5e-3	5	1e-4	84.51	0.95	0.90	0.78	4.52	1.76
5e-3	5	1e-3	83.07	0.96	0.93	0.84	4.32	0.82
5e-3	5	1e-2	80.53	0.97	0.94	0.87	4.06	0.32
5e-3	1	1e-7	82.84	0.91	0.85	0.70	4.76	4.11
5e-3	1	1e-6	83.40	0.92	0.86	0.72	4.85	3.37
5e-3	1	1e-5	81.72	0.93	0.88	0.75	4.65	2.62
5e-3	1	1e-4	82.19	0.95	0.91	0.81	4.68	1.52
5e-3	1	1e-3	81.14	0.96	0.93	0.86	4.43	0.74
5e-3	1	1e-2	79.13	0.96	0.94	0.86	4.16	0.32
5e-3	5e-1	1e-7	77.41	0.96	0.93	0.85	4.59	2.03
5e-3	5e-1	1e-6	77.42	0.97	0.94	0.85	4.57	1.91
5e-3	5e-1	1e-5	76.88	0.97	0.94	0.86	4.47	1.57
5e-3	5e-1	1e-4	77.17	0.97	0.94	0.86	4.49	1.24
5e-3	5e-1	1e-3	76.68	0.97	0.95	0.89	4.29	0.64
5e-3	5e-1	1e-2	75.01	0.98	0.97	0.92	3.89	0.27
1e-2	10	1e-7	64.43	0.94	0.87	0.65	3.37	7.11
1e-2	10	1e-6	79.77	0.90	0.83	0.66	3.46	4.06
1e-2	10	1e-5	79.87	0.91	0.84	0.69	3.46	2.47
1e-2	10	1e-4	78.86	0.94	0.88	0.75	3.37	1.34
1e-2	10	1e-3	77.68	0.95	0.92	0.82	3.25	0.63
1e-2	10	1e-2	75.75	0.97	0.94	0.87	3.12	0.25
1e-2	5	1e-7	78.63	0.89	0.81	0.65	3.40	4.42
1e-2	5	1e-6	77.74	0.90	0.83	0.66	3.34	4.21
1e-2	5	1e-5	78.16	0.90	0.83	0.68	3.35	2.34
1e-2	5	1e-4	78.18	0.92	0.86	0.73	3.31	1.20
1e-2	5	1e-3	77.43	0.96	0.92	0.82	3.25	0.59
1e-2	5	1e-2	74.45	0.96	0.93	0.85	3.02	0.25
1e-2	1	1e-7	71.74	0.97	0.94	0.86	3.21	1.26
1e-2	1	1e-6	71.92	0.97	0.95	0.88	3.24	1.18
1e-2	1	1e-5	73.02	0.97	0.94	0.86	3.30	0.98
1e-2	1	1e-4	72.02	0.97	0.95	0.88	3.16	0.77
1e-2	1	1e-3	71.16	0.98	0.95	0.89	3.08	0.43
1e-2	1	1e-2	69.64	0.98	0.97	0.92	2.90	0.19
1e-2	5e-1	1e-7	70.53	0.98	0.96	0.90	3.20	0.87
1e-2	5e-1	1e-6	70.07	0.98	0.96	0.90	3.21	0.89
1e-2	5e-1	1e-5	70.46	0.98	0.96	0.89	3.20	0.79
1e-2	5e-1	1e-4	71.49	0.98	0.96	0.91	3.31	0.72
1e-2	5e-1	1e-3	70.06	0.99	0.97	0.93	3.11	0.41
1e-2	5e-1	1e-2	67.76	0.99	0.98	0.94	2.84	0.19

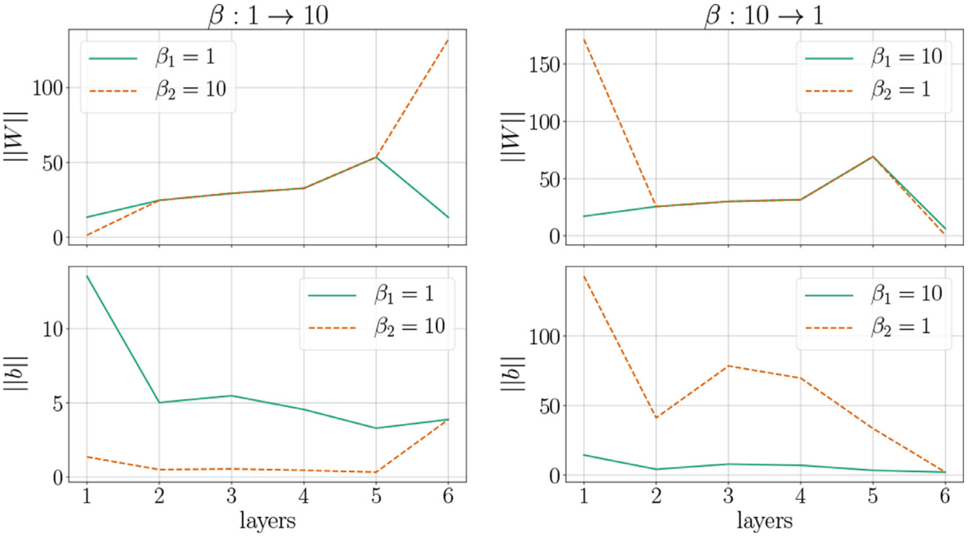

Fig. C.11. Weights and biases for networks with identical outputs but different β value for the softplus activation. Left: β was changed from 1 to 10. Right: β was changed from 10 to 1.

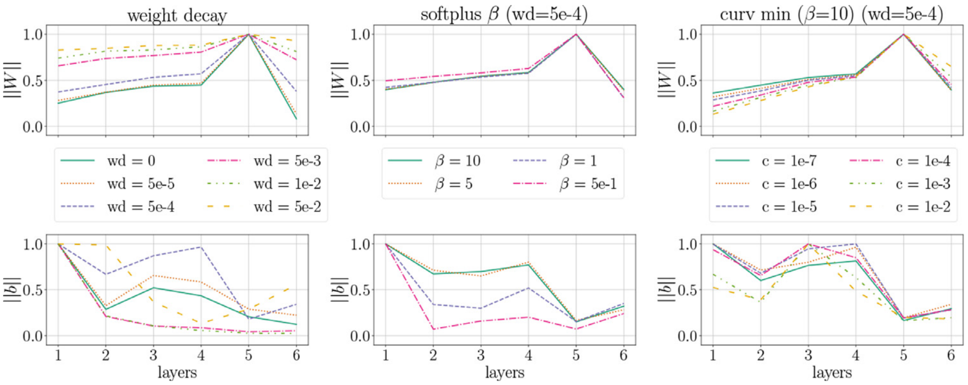

Fig. C.12. Weights and biases plotted over layers for different networks (left column: networks with weight decay, middle column: networks with different β values, right column: networks trained with curvature minimization).

Appendix C. Interchangeability of softplus β

C1. Interchangeability of softplus β

When training softplus networks with different β values it is interesting to consider how they differ as the beta values partially cancel out or can be absorbed into the weights and biases.

The softplus function is defined as:

$$\mathrm{sp}_\beta(x) = \frac{1}{\beta}\ln(1 + \mathrm{e}^{\beta x}) \tag{C.1}$$

Therefore, we can relate two softplus functions with different β values β_1 and β_2 as follows:

$$\mathrm{sp}_{\beta_1}(x) = \frac{\beta_2}{\beta_1}\mathrm{sp}_{\beta_2}(\frac{\beta_1}{\beta_2}x) \tag{C.2}$$

A network consisting of linear layers and softplus activations with $\beta = \beta_1$ has weights $W^{(i)}$ and biases $b^{(i)}$. We can define a network with the same structure but a different softplus $\beta = \beta_2$ and weights $\tilde{W}^{(i)}$ and biases $\tilde{b}^{(i)}$. The networks give identical outputs for all inputs if we define the weights and biases of the second network in the following way:

$$\tilde{W}^{(1)} = \frac{\beta_1}{\beta_2}W^{(1)}$$
$$\tilde{W}^{(i)} = W^{(i)}, \quad \forall i : 1 < i < n$$
$$\tilde{W}^{(n)} = \frac{\beta_2}{\beta_1}W^{(n)}$$
$$\tilde{b}^{(i)} = \frac{\beta_1}{\beta_2}b^{(i)}, \quad \forall i : i < n$$
$$\tilde{b}^{(n)} = b^{(n)}$$

However, this mapping is not learned when training networks with different β values from scratch as the distribution over weight norms stays very similar while the distribution changes drastically when artificially changing the β value as demonstrated above. We show this effect for a few examples in C.2.

C2. Examples

In Section C.1, we show that, by adjusting weights and biases of a softplus network, it can be functionally equivalent to a network with the same structure but different β value for the softplus activation. Artificially constructing networks in this way leads to a larger variance in the weight norms. Even when no weight decay is used during training the weight norms of the different network layers in one network tend to vary within one order of magnitude.

Fig. C.11 shows the weights and biases for two networks from Table D.2 with $\beta = 1$ and $\beta = 10$ and the respective weights and biases for two networks that produce identical output but have changed β values. In both cases the average weight norm of the constructed network is higher than of the original.

Fig. C.12 shows weights and biases for some of our networks from Table D.2. The weight and bias norms for each network are normalized with the respective maximum value over all layers. Without exception the highest weight norm is found in layer 5 in contrast to the maximum weight norm when we do the artificial β value switch. Thus training softplus networks from scratch does produce fundamentally different networks that cannot be obtained with a mere rescaling of weights and biases.

Appendix D. Experimental analysis

D1. Error measures

In this section we define the error measures we use to quantify our results. To ease notation we refer to the explanation of our original image as $h(x) = u$ and to the explanation of our perturbed image as $h(x_{\text{adv}}) = v$. both explanations can be expressed as a vector of length n.

- The **Mean squared error (MSE)** between two explanations is defined as

  $$\text{MSE}(u, v) = \frac{1}{n}\sum_{i}^{n}(u_i - v_i)^2$$

- The **Pearson correlation coefficient (PCC)** between two explanations is given by

  $$\text{PCC}(u, v) = \frac{\sum_i^n (u_i - \mu_u)(v_i - \mu_v)}{\sqrt{\sum_i^n (u_i - \mu_u)^2}\sqrt{\sum_i^n (v_i - \mu_v)^2}}$$

  where $\mu_u = \frac{1}{n}\sum_i^n u_i$ is the mean value of explanation u. Analogously for μ_v.
- The **Structural similarity index (SSIM)** between two explanations is given by calculating

  $$\text{SSIM}(u, v) = \frac{(2\mu_u\mu_v + (k_1L)^2)(2\sigma_{uv} + (k_2L)^2)}{(\mu_u^2 + \mu_v^2 + (k1L)^2)(\sigma_u^2 + \sigma_v^2 + (k_2L)^2)}$$

  for a 7 × 7 window centered at each pixel and then averaging over all windows. Mean values μ_u and μ_v and variances σ_{uv}, σ_u and σ_v are calculated separately for each 7 × 7 window. L is the range between the largest and smallest value that a pixel in the explanation can have and $k_1 = 0.01$ and $k_2 = 0.03$ are constants to stabilize the division.
- The **accuracy (acc)** of a network is the ratio of correctly classified images of the test set:

  $$\text{acc} = \frac{\textit{correct predictions}}{\textit{all predictions}}$$

D2. Network structure

The structure of all networks trained within the scope of this work is depicted in Fig. D.13. The activation function is either ReLU or softplus (for the networks trained with β smoothing or Hessian minimization). In order to focus on the robustness we aimed to train the different networks to similar accuracy (albeit no longer than 200 epochs). We use Stochastic Gradient Descent with momentum and learning rate decay. We do not perform any further hyperparameter optimization. Statistics for all trained networks are summarized in Table D.2.

D3. Gradient explanation

In Figs. D.14, D.15, and D.16, we show additional error measures for the Gradient explanation. PCC and SSIM increase with robustness while MSE decreases.

```
CNN_CIFAR(
  (features): Sequential(
    (conv0): Conv2d(3, 32, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti0): ActivationFunction()
    (conv1): Conv2d(32, 32, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti1): ActivationFunction()
    (pool2): MaxPool2d(kernel_size=2, stride=2,
                    padding=0, dilation=1,
                    ceil_mode=False)
    (conv3): Conv2d(32, 64, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti3): ActivationFunction()
    (conv4): Conv2d(64, 64, kernel_size=(3, 3),
                    stride=(1, 1), padding=(1, 1))
    (acti4): ActivationFunction()
    (pool5): MaxPool2d(kernel_size=2, stride=2,
                    padding=0, dilation=1,
                    ceil_mode=False)
  )
  (classifier): Sequential(
    (view0): Reshape()
    (dens0): Linear(in_features=4096,
                    out_features=256,
                    bias=True)
    (acti0): ActivationFunction()
    (dens1): Linear(in_features=256,
                    out_features=10,
                    bias=True)
  )
)
```

Fig. D.13. Setup of simple CNN for CIFAR-10.

D4. Other explanation methods

In Figs. D.17, D.18, D.19, and D.20, we show how our proposed measures effect other explanation methods. The trend towards increased robustness is clearly visible for all considered explanation methods. We note that the explanations start from different levels of robustness but can still profit from our methods. The most resilient method against random input perturbations is Layerwise Relevance Propagation, followed by Guided Backpropagation, Integrated Gradients, Gradient×Input and Gradient in descending order.

D5. Other types of noise

In the main text we only consider Gaussian noise. We repeat our experiments from 4 for the Gradient explanation when we perturb the input images with Laplacian noise and salt-pepper noise.

D5.1. Laplace noise

We sample random noise from the Laplace distribution

$$f(x|\mu, b) = \frac{1}{2b}\exp -\frac{|x - \mu|}{b} \tag{D.1}$$

where μ is the data mean and b is a scale parameter which we define as $b = (x_{\max} - x_{\min})\nu$, depending on the noise level ν. Fig. D.21 shows effects on the Gradient explanation when adding Laplace noise to the input images. We see that the results look statistically very similar to the results for Gaussian noise.

D5.2. Salt-pepper noise

To perturb an image with salt-pepper noise we randomly select $100 * \frac{\nu}{2}$ % of the pixel in the image and switch them to $x_{\max}$ (white) or $x_{\min}$ (black) at random. We select a very small amount

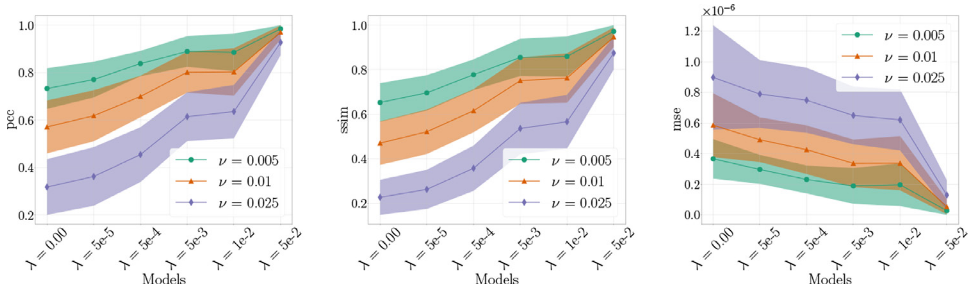

Fig. D.14. PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for networks trained with weight decay. That means weight decay improves robustness of explanations. We show mean +/- std.

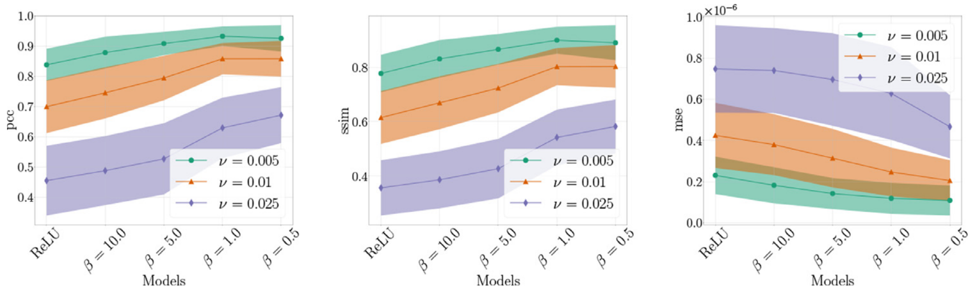

Fig. D.15. PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for softplus networks trained with a small β value. That means softplus activations improve robustness of explanations. All nets were trained with weight decay (λ=5e-4). We show mean +/- std.

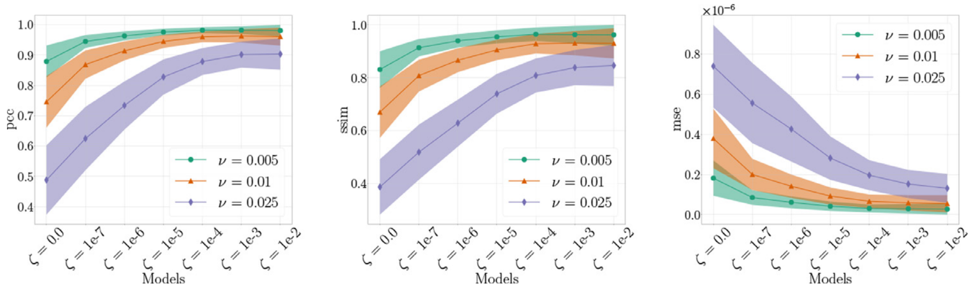

Fig. D.16. PCC, SSIM and MSE between original Gradient explanation map and explanation after adding random noise to the image. PCC and SSIM are higher and MSE is lower for networks trained with strong curvature minimization. That means minimizing the curvature improves robustness of explanations. All nets were trained with softplus activations ($\beta = 10$) and weight decay (λ=5e-4). We show mean +/- std.

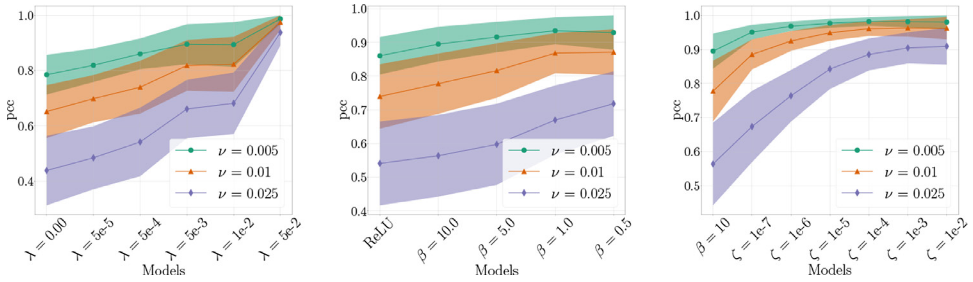

Fig. D.17. PCCs (mean +/- std) between original Gradient×Input explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e-4$), right: effect of curvature minimization (for $\lambda = 5e-4$, β=10).

of pixel (for noise level $\nu = 0.005$ only 3 pixels) to be perturbed as salt-pepper noise has a very strong effect on the classification accuracy which we aim to keep approximately constant.

Fig. D.22 shows effects on the Gradient explanation when adding salt-pepper noise to the input images. We can still see a significant improvement in robustness when training networks with our proposed methods, however the effect for softplus activations and Hessian minimization is less pronounced than for Laplace or Gaussian noise.

D6. Connection between Hessian norms and weight norms

In the main text, Section 4.6, we mentioned that weight decay leads to decreased weight norms and thus also to decreased Hessian norms. However when we minimize the Hessian norm directly, the weight norms only change minimally. Fig. D.23 shows how weight norms and Hessian norms change when applying weight decay (varying λ), substituting ReLU with softplus (varying β) and minimizing the Hessian norm directly (varying ζ). We

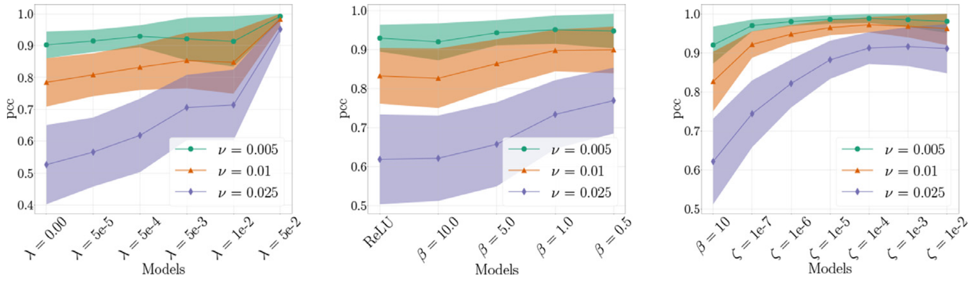

Fig. D.18. PCCs (mean +/- std) between original Integrated Gradients explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

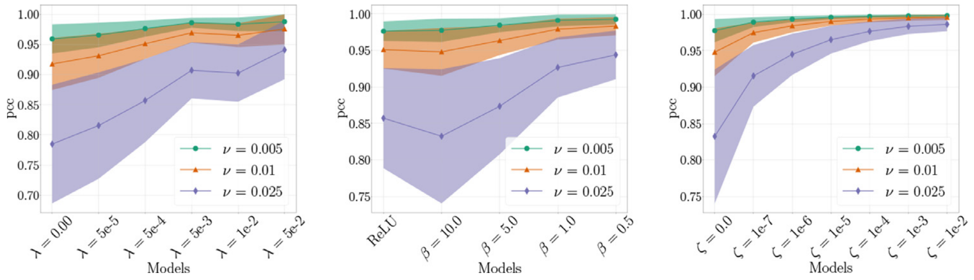

Fig. D.19. PCCs (mean +/- std) between original Guided Backpropagation explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

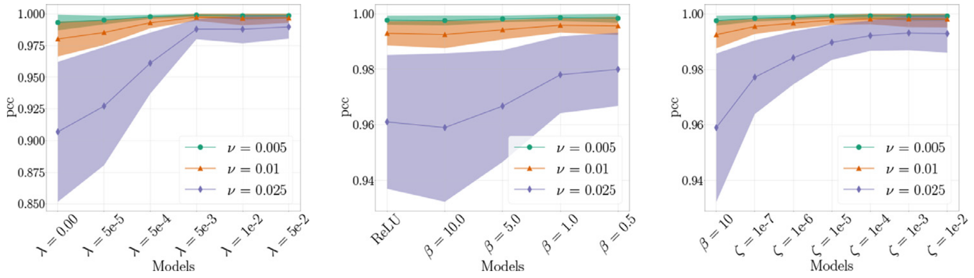

Fig. D.20. PCCs (mean +/- std) between original Layerwise Relevance Propagation explanation map and explanation after adding random noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

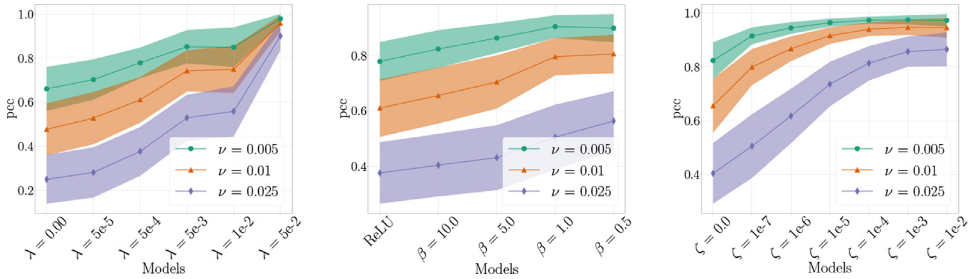

Fig. D.21. PCCs (mean +/- std) between original Gradient explanation map and explanation after adding Laplace noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

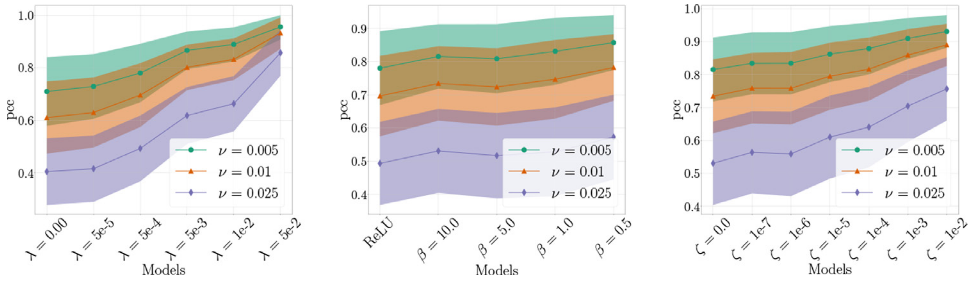

Fig. D.22. PCCs (mean +/- std) between original Gradient explanation map and explanation after adding salt-pepper noise to the image. left: effect of weight decay, middle: effect of softplus β (for $\lambda = 5e\text{-}4$), right: effect of curvature minimization (for $\lambda = 5e\text{-}4$, β=10).

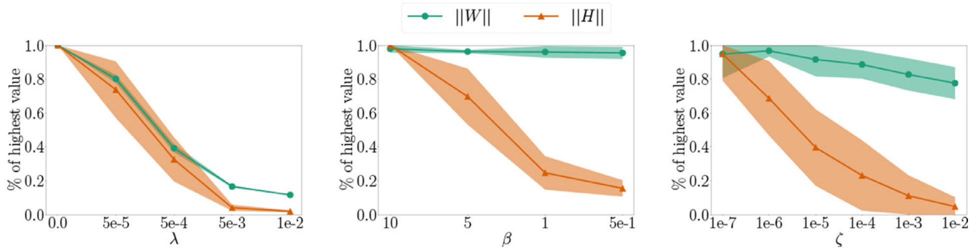

Fig. D.23. The connection between weight norms $\|W\|$ and Hessian norms $\|H\|$ is different for our three methods. Left: $\|W\|$ and $\|H\|$ both decrease in a similar manner when applying stronger weight decay (increasing λ). Middle: $\|W\|$ stays relatively constant while $\|H\|$ decreases with increasing softplus β. Right: $\|W\|$ decrease slightly while $\|H\|$ decrease strongly with stronger Hessian minimization (increasing ζ). We show mean±std.

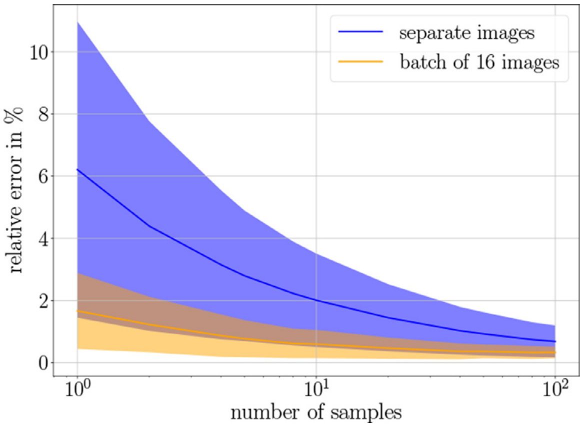

Fig. E.24. Relative error (mean and standard deviation) between True Hessian norm and approximation via sampling.

average over all softplus networks trained with $\zeta = 0$ for the first two plots and we average over all networks trained with Hessian minimization for the last plot.

Appendix E. Hessian norm approximation

In Section 3.1, we showed that for the number of samples $N \to \infty$ the sampling approximation approaches the true Hessian norm. To include the Hessian approximation in our training procedure, we need to fix a certain number of samples and to perform a Monte-Carlo estimate of the expectation value. Fig. E.24 shows the relative error between sampled Hessian norm and true Hessian norm. Increasing the sample size noticeably reduces the error, but a sample size of 1 already has a relative error of only 6%. If we average over a batch of images, the error reduces further. This means that for training and validation sampling once per image is in practice sufficient.

Appendix F. Additional network structures and data sets

Data set: We train on ImageNette [56], a data set that contains 10 sub classes of the ImageNet data set (tench, English springer, cassette player, chain saw, church, French horn, garbage truck, gas pump, golf ball, parachute). We train on the original train-test split of ImageNet, using 50 images per class for testing. *Network architectures:* We use two different architectures: VGG16 [57] without batch normalization and ResNet18 [58]. *Configuration:* We select the most promising parameters for each of our proposed methods and train with four different parameter configurations for each network architecture. We train a vanilla network without weight decay ($\lambda = 0$), a network with moderate weight decay ($\lambda =$ 5e-4), a network with softplus activations ($\lambda =$ 5e-4, $\beta = 10$) and a network with curvature minimization ($\lambda =$ 5e-4, $\beta = 10$, $\zeta =$ 1e-6). All networks were trained for 60 to 70 epochs to make them more comparable and selected based on their validation set accuracy. We list test accuracies for the different networks in Table F.3.

Table F.3
Accuracies on ImageNette test set for VGG16 and ResNet18 with different parameter configurations. .

architecture	λ	β	ζ	test acc
VGG16	0.0	ReLU	0.0	84.0
VGG16	5e-4	ReLU	0.0	84.2
VGG16	5e-4	10	0.0	79.6
VGG16	5e-4	10	1e-6	80.8
ResNet18	0.0	ReLU	0.0	89.0
ResNet18	5e-4	ReLU	0.0	89.8
ResNet18	5e-4	10	0.0	88.6
ResNet18	5e-4	10	1e-6	89.2

We examine the robustness against random perturbations with different noise levels and show the results in Fig. F.25 and F.26. For both architectures we observe a clear trend towards increased robustness when applying our methods. Interestingly we get the largest increase in robustness when substituting ReLU activations with softplus activations while curvature minimization does not seem to have a large effect. In Fig. F.27 we compare weight norms and Hessian norms between the three network architectures we analyse. Compared to the CNN trained on CIFAR, approximated Hessian norms for VGG16 are an order of magnitude smaller even without curvature minimization. For ResNet18 they are two orders of magnitude smaller.

Appendix G. Targeted adversarial attacks

We perform targeted adversarial attacks on the CNN model that was trained on CIFAR10 as well as on VGG16 and ResNet18 trained on ImageNette for 100 test images. We choose the target explanations from another 100 randomly selected test images. We stop an attack when the mean squared error between original and manipulated input exceeds 0.0015. We use the Adam optimiser with standard parameters and adapt the learning rate between configurations so that the attacks need roughly the same number of iterations until the stopping criterion is reached. As ReLU networks are not twice differentiable we substitute the ReLU activations with softplus activations with small β value during the attack. For the final comparison we re-substitute the ReLU activations.

We always compare a vanilla net ($\lambda = 0$, ReLU activations, $\zeta = 0$) with a robust net (weight decay with $\lambda =$ 5e-4, softplus activations with $\beta = 10$, curvature minimization with $\zeta =$ 1e-6). As our attacks are targeted we compare the manipulated explanation h to the original explanation h_{org} and the target explanation h^t in Fig. G.28. For the CNNs trained on CIFAR and VGG16 trained on ImageNette we see a large positive effect on the robustness when using our methods during training. For ResNet18 we see a smaller effect.

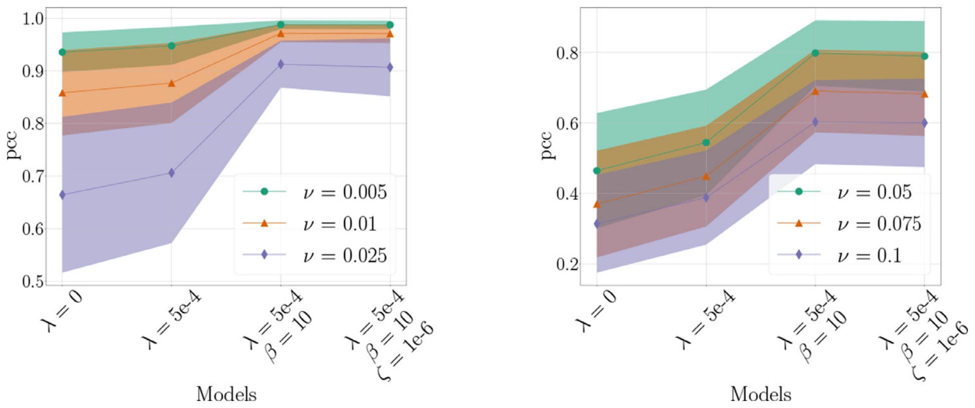

Fig. F.25. Experiments on VGG16 trained on the ImageNette data set. PCC increases with weight decay, softplus activations and curvature minimization. We show mean +/- std for six different noise levels ν..

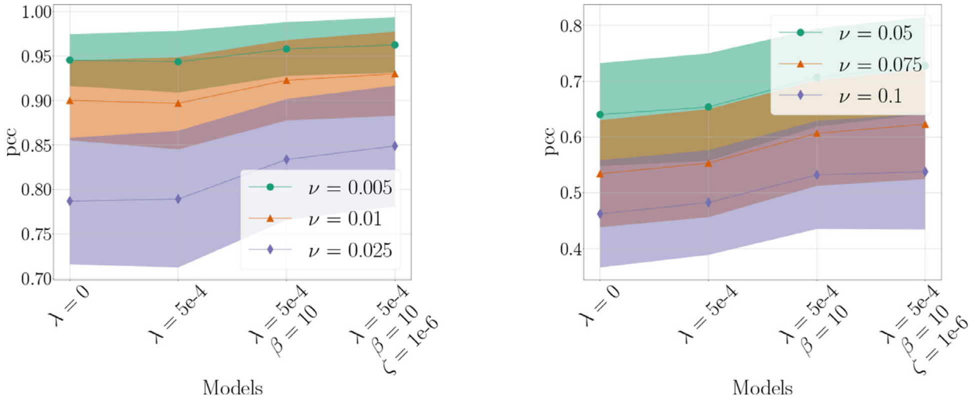

Fig. F.26. Experiments on ResNet18 trained on the ImageNette data set. PCC increases with weight decay, softplus activations and curvature minimization. We show mean +/- std for six different noise levels ν..

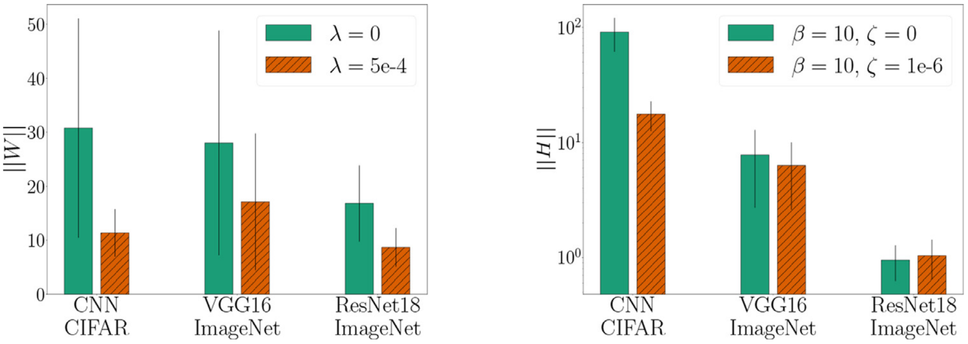

Fig. F.27. Left: effect of weight decay on different architectures. Right: effect of curvature minimization on different architectures (a weight decay of $\lambda =$ 5e-4 was used).

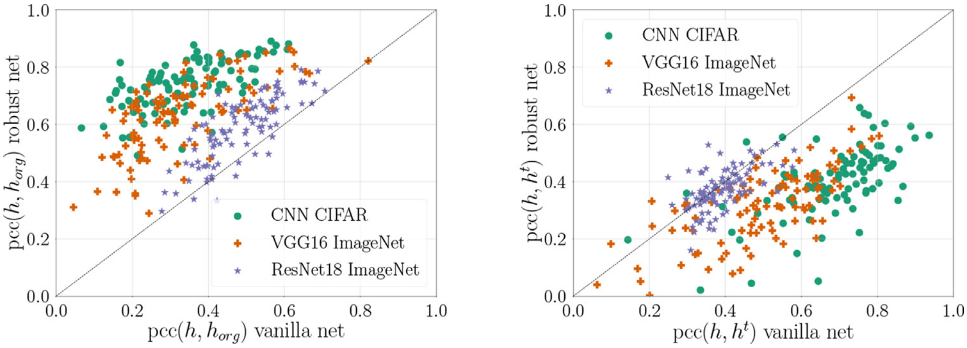

Fig. G.28. Targeted adversarial attacks on different models. Left: similarity between manipulated explanation h and original explanation h_{org} is *higher* for the robust nets trained with our methods. Right: similarity between manipulated explanation h and target explanation h^t is *lower* for the robust nets trained with our methods. This shows that our methods also improve robustness against targeted adversarial attacks..

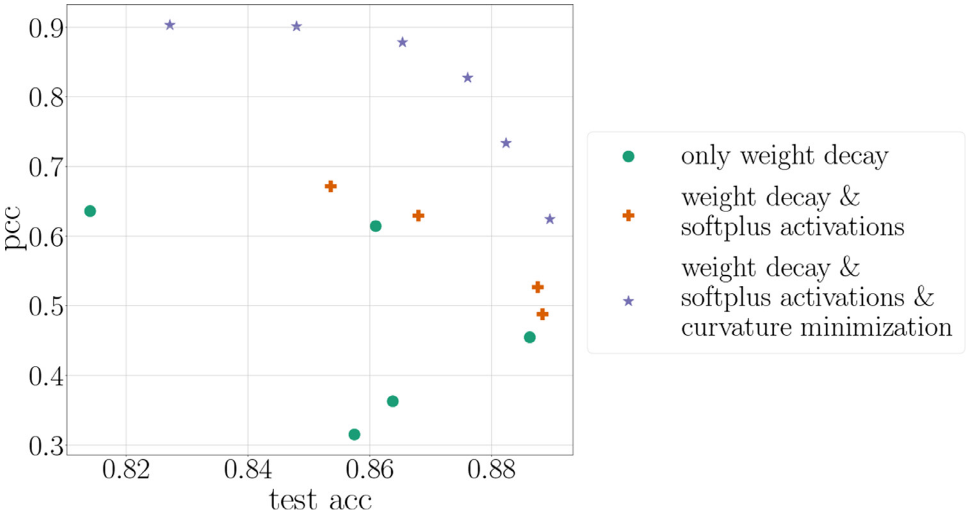

Fig. H.29. Tradeoff between robustness (high PCC between explanations) and accuracies for selected CNNs trained on CIFAR10..

Appendix H. Accuracy-robustnes tradeoff

As discussed in the main text, increasing the robustness of networks comes at the price of decreased accuracy. This is illustrated in Figure H.29. The best trade-off is obtained by combining all proposed methods.

References

[1] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, Journal of Machine Learning Research 11 (61) (2010) 1803–1831. http://jmlr.org/papers/v11/baehrens10a.html

[2] K. Simonyan, A. Vedaldi, A. Zisserman, Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, in: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Workshop Track Proceedings, 2014. http://arxiv.org/abs/1312.6034

[3] M.D. Zeiler, R. Fergus, Visualizing and Understanding Convolutional Networks, in: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I, 2014, pp. 818–833, doi:10.1007/978-3-319-10590-1_53.

[4] J.T. Springenberg, A. Dosovitskiy, T. Brox, M.A. Riedmiller, Striving for Simplicity: The All Convolutional Net, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Workshop Track Proceedings, 2015. http://arxiv.org/abs/1412.6806

[5] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, W. Samek, On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation, PLoS ONE 10 (7) (2015) 1–46, doi:10.1371/journal.pone.0130140.

[6] R.R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, D. Batra, Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 618–626, doi:10.1109/ICCV.2017.74.

[7] M.T. Ribeiro, S. Singh, C. Guestrin, “Why Should I Trust You?”: Explaining the Predictions of Any Classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, 2016, pp. 1135–1144, doi:10.1145/2939672.2939778.

[8] L.M. Zintgraf, T.S. Cohen, T. Adel, M. Welling, Visualizing Deep Neural Network Decisions: Prediction Difference Analysis, in: 5th International Conference on Learning Representations (ICLR), Toulon, France, April 24–26, 2017, Conference Track Proceedings, 2017. https://openreview.net/forum?id=BJ5UeU9xx

[9] A. Shrikumar, P. Greenside, A. Kundaje, Learning Important Features Through Propagating Activation Differences, in: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, 2017, pp. 3145–3153. http://proceedings.mlr.press/v70/shrikumar17a.html

[10] S.M. Lundberg, S.-I. Lee, A Unified Approach to Interpreting Model Predictions, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[11] P. Dabkowski, Y. Gal, Real Time Image Saliency for Black Box Classifiers, in: Advances in Neural Information Processing Systems 30, Curran Associates, Inc., 2017, pp. 6967–6976. http://papers.nips.cc/paper/7272-real-time-image-saliency-for-black-box-classifiers.pdf

[12] D. Smilkov, N. Thorat, B. Kim, F.B. Viégas, M. Wattenberg, SmoothGrad: removing noise by adding noise, in: Workshop on Visualization for Deep Learning, International Conference on Machine Learning (ICML) 2017, Sydney, Australia, Aug 10, 2017.

[13] M. Sundararajan, A. Taly, Q. Yan, Axiomatic Attribution for Deep Networks, in: Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017, 2017, pp. 3319–3328. http://proceedings.mlr.press/v70/sundararajan17a.html

[14] R.C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 3449–3457.

[15] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, K.-R. Müller, Explaining nonlinear classification decisions with deep Taylor decomposition, Pattern Recognition 65 (2017) 211–222.

[16] P.-J. Kindermans, K.T. Schütt, M. Alber, K.-R. Müller, D. Erhan, B. Kim, S. Dähne, Learning how to explain neural networks: PatternNet and PatternAttribution, in: International Conference on Learning Representations (ICLR), Vancouver Convention Center, Vancouver, BC, Canada April 30, – May 3, 2018. https://openreview.net/forum?id=Hkn7CBaTW

[17] B. Kim, M. Wattenberg, J. Gilmer, C.J. Cai, J. Wexler, F.B. Viégas, R. Sayres, Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV), in: Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, pp. 2673–2682.

[18] G. Montavon, W. Samek, K.-R. Müller, Methods for interpreting and understanding deep neural networks, Digital Signal Processing 73 (2018) 1–15, doi:10.1016/j.dsp.2017.10.011.

[19] W. Samek, G. Montavon, A. Vedaldi, L.K. Hansen, K.-R. Müller, Explainable AI: Interpreting, explaining and visualizing deep learning, volume 11700, Springer Nature, 2019.

[20] W. Samek, G. Montavon, S. Lapuschkin, C.J. Anders, K.-R. Müller, Explaining deep neural networks and beyond: A review of methods and applications., Proceedings of the IEEE 109 (2021) 247–278.

[21] I. Sturm, S. Lapuschkin, W. Samek, K.-R. Müller, Interpretable deep neural networks for single-trial EEG classification, Journal of Neuroscience Methods 274 (2016) 141–145.

[22] F. Arbabzadah, G. Montavon, K.-R. Müller, W. Samek, Identifying Individual Facial Expressions by Deconstructing a Neural Network, in: German Conference on Pattern Recognition, Springer, 2016, pp. 344–354.

[23] K.T. Schütt, F. Arbabzadah, S. Chmiela, K.R. Müller, A. Tkatchenko, Quantum–chemical insights from deep tensor neural networks, Nature Communications 8 (1) (2017) 13890.

[24] M. Hägele, P. Seegerer, S. Lapuschkin, M. Bockmayr, W. Samek, F. Klauschen, K.-R. Müller, A. Binder, Resolving challenges in deep learning-based analyses of histopathological images using explanation methods, Scientific Reports 10 (1) (2020) 6423.

[25] T. Zahavy, N.B. Zrihem, S. Mannor, Graying the Black Box: Understanding DQNs, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML) - Volume 48, 2016, pp. 1899–1908.

[26] L. Arras, F. Horn, G. Montavon, K.-R. Müller, W. Samek, “What is relevant in a text document?”: An interpretable machine learning approach, PLoS ONE 12 (8) (2017) e0181142.

[27] S. Greydanus, A. Koul, J. Dodge, A. Fern, Visualizing and Understanding Atari Agents, in: International Conference on Machine Learning, PMLR, 2018, pp. 1792–1801.

[28] S. Lapuschkin, A. Binder, G. Montavon, K.-R. Muller, W. Samek, Analyzing classifiers: Fisher vectors and deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2912–2920.

[29] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn, Nature Communications 10 (1) (2019) 1096.

[30] C.J. Anders, L. Weber, D. Neumann, W. Samek, K.-R. Müller, S. Lapuschkin, Finding and removing Clever Hans: Using explanation methods to debug and improve deep models, Information Fusion (2021), doi:10.1016/j.inffus.2021.07.015.

[31] A. Ghorbani, A. Abid, J.Y. Zou, Interpretation of Neural Networks Is Fragile, in: The 33rd Conference on Artificial Intelligence, AAAI, 2019, pp. 3681–3688.

[32] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, P. Kessel, Explanations can be manipulated and geometry is to blame, in: Advances in Neural Information Processing Systems, 2019, pp. 13567–13578.

[33] J. Heo, S. Joo, T. Moon, Fooling Neural Network Interpretations via Adversarial Model Manipulation, in: Advances in Neural Information Processing Systems, 2019, pp. 2921–2932.

[34] C. Anders, P. Pasliev, A.-K. Dombrowski, K.-R. Müller, P. Kessel, Fairwashing Explanations with Off-Manifold Detergent, in: Proceedings of the 37th International Conference on Machine Learning, ICML, Vienna, Austria, PMLR 119, 2020.

[35] Parliament and Council of the European Union, . automated individual decision making, including profiling, Official Journal of the European Union, 2016. https://eur-lex.europa.eu/

[36] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, K.-R. Müller, Evaluating the visualization of what a deep neural network has learned, IEEE Transactions on Neural Network and Learning Systems 28 (2017) 2660–2673, doi:10.1109/TNNLS.2016.2599820.

[37] L. Arras, G. Montavon, K.-R. Müller, W. Samek, Explaining recurrent neural network predictions in sentiment analysis, in: Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 159–168, doi:10.18653/v1/W17-5221.

[38] R. Tomsett, D. Harborne, S. Chakraborty, P. Gurram, A. Preece, Sanity checks for saliency metrics, Proceedings of the AAAI Conference on Artificial Intelligence (04) (2020) 6021–6029, doi:10.1609/aaai.v34i04.6064. https://ojs.aaai.org/index.php/AAAI/article/view/6064

[39] S. Hooker, D. Erhan, P.-J. Kindermans, B. Kim, A Benchmark for Interpretability Methods in Deep Neural Networks, in: Advances in Neural Information Processing Systems, 2019, pp. 9734–9745.

[40] J. Adebayo, J. Gilmer, M. Muelly, I.J. Goodfellow, M. Hardt, B. Kim, Sanity Checks for Saliency Maps, in: Advances in Neural Information Processing Systems, 31, NeurIPS, Montréal, Canada, 2018, pp. 9525–9536.

[41] L. Sixt, M. Granz, T. Landgraf, When Explanations Lie: Why Many Modified BP Attributions Fail, in: Proceedings of the 37th International Conference on Machine Learning, ICML, Vienna, Austria, PMLR 119, 2020.

[42] P. Kindermans, S. Hooker, J. Adebayo, M. Alber, K.T. Schütt, S. Dähne, D. Erhan, B. Kim, The (Un)reliability of Saliency Methods, in: Explainable AI: Interpreting, Explaining and Visualizing Deep Learning, Springer, 2019, pp. 267–280.

[43] J. Wang, J. Tuyls, E. Wallace, S. Singh, Gradient-based analysis of NLP models is manipulable, arXiv preprint arXiv:2010.05419 (2020).

[44] D. Slack, S. Hilgard, E. Jia, S. Singh, H. Lakkaraju, Fooling LIME and SHAP: Adversarial Attacks on Post hoc Explanation Methods, in: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, 2020, pp. 180–186.

[45] Z. Wang, H. Wang, S. Ramkumar, P. Mardziel, M. Fredrikson, A. Datta, Smoothed geometry for robust attribution, Advances in Neural Information Processing Systems 33 (2020).

[46] L. Rieger, L.K. Hansen, A simple defense against adversarial attacks on heatmap explanations, 2020 Workshop on Human Interpretability in Machine Learning (WHI) (2020).

[47] H. Lakkaraju, N. Arsov, O. Bastani, Robust and stable black box explanations, in: Proceedings of the 37th International Conference on Machine Learning, ICML, Vienna, Austria, PMLR 119, 2020.

[48] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, P. Frossard, Robustness via curvature regularization, and vice versa, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 9070–9078.

[49] B.A. Pearlmutter, Fast exact multiplication by the hessian, Neural Comput 6 (1) (1994) 147–160, doi:10.1162/neco.1994.6.1.147.

[50] A. Krogh, J.A. Hertz, A Simple Weight Decay Can Improve Generalization, in: Advances in neural information processing systems, 1992, pp. 950–957.

[51] S. Hanson, L. Pratt, Comparing biases for minimal network construction with back-propagation, Advances in Neural Information Processing Systems 1 (1988) 177–185.

[52] A.S. Weigend, D.E. Rumelhart, B.A. Huberman, Generalization by Weight-Elimination with Application to Forecasting, in: Advances in neural information processing systems, 1991, pp. 875–882.

[53] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, 2016. http://www.deeplearningbook.org

[54] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, CIFAR10 dataset, Technical Report, 2009. https://www.cs.toronto.edu/~kriz/cifar.html

[55] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, K.-R. Müller, Layer-Wise Relevance Rropagation: An Overview, in: Explainable AI: interpreting, explaining and visualizing deep learning, Springer, 2019, pp. 193–209.

[56] J. Howard, Imagenette, https://github.com/fastai/imagenette/.

[57] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings, 2015. http://arxiv.org/abs/1409.1556

[58] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016, 2016, pp. 770–778.

[59] M.L. Braun, J.M. Buhmann, K.-R. Müller, On relevant dimensions in kernel feature spaces, Journal of Machine Learning Research 9 (62) (2008) 1875–1908. http://jmlr.org/papers/v9/braun08a.html

[60] G. Montavon, M.L. Braun, K.-R. Müller, Kernel analysis of deep networks, Journal of Machine Learning Research 12 (78) (2011) 2563–2581. http://jmlr.org/papers/v12/montavon11a.html

[61] J. Kauffmann, M. Esders, G. Montavon, W. Samek, K.-R. Müller, From clustering to cluster explanations via neural networks, arXiv preprint arXiv:1906.07633 (2019).

[62] J. Kauffmann, K.-R. Müller, G. Montavon, Towards explaining anomalies: a deep taylor decomposition of one-class models, Pattern Recognition 101 (2020) 107198.

[63] L. Ruff, J.R. Kauffmann, R.A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T.G. Dietterich, K.-R. Müller, A unifying review of deep and shallow anomaly detection, Proceedings of the IEEE 109 (2021) 756–795.

[64] O. Eberle, J. Büttner, F. Kräutli, K.-R. Müller, M. Valleriani, G. Montavon, Building and interpreting deep similarity models, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021), doi:10.1109/TPAMI.2020.3020738.

Ann-Kathrin Dombrowski received a Bachelor's Degree in 2014 and a Master's Degree in 2017 in Computational Engineering Science from RWTH Aachen. She is currently a Ph. D. student in the Machine Learning Group at TU Berlin.

Christopher J. Anders received a Bachelor's Degree in 2016 and a Master's Degree in 2018 both in Computer Science from Technische Universität Berlin. He is currently pursuing his Ph.D. at the Machine Learning Group at TU Berlin.

Klaus-Robert Müller(Ph.D. 92) has been a Professor of computer science at TU Berlin since 2006; director Berlin Institute for the Foundations of Learning and Data. In 2012, he was elected member of the German National Academy of Sciences–Leopoldina, in 2017 of the Berlin Brandenburg Academy of Sciences and also in 2017 external scientific member of the Max Planck Society. In 2019 and 2020 he became ISI Highly Cited researcher in the cross-disciplinary area.

Pan Kessel obtained his PhD in String Theory in 2017 at the Max Planck Institute for Gravitational Physics. He is currently a postdoctoral researcher in the Machine Learning Group of TU Berlin and at the Berlin Institute for the Foundations of Learning and Data (BIFOLD).